



**FAKULTA
STROJNÍ
ČVUT V PRAZE**

Department of Instrumentation and Control Engineering

**Monitorování průmyslového HMI pomocí
strojového vidění**

**Machine vision based monitoring of
industrial HMI**

Master Thesis

2023

Bc. Jakub HORÁK

Study program: Automation and Instrumentation Engineering
Field of study: Automation and industrial informatics
Supervisor: Ing. Cyril Oswald, Ph.D.

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Horák** Jméno: **Jakub** Osobní číslo: **482439**
Fakulta/ústav: **Fakulta strojní**
Zadávací katedra/ústav: **Ústav přístrojové a řídicí techniky**
Studijní program: **Automatizační a přístrojová technika**
Specializace: **Automatizace a průmyslová informatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Monitorování průmyslového HMI pomocí strojového vidění

Název diplomové práce anglicky:

Machine vision based monitoring of industrial HMI

Pokyny pro vypracování:

- Navrhněte systém pro monitorování průmyslového HMI využívající strojového vidění.
- Proveďte rešerši standardních prvků průmyslových HMI s cílem vytípnout skupinu typických základních prvků pro monitorování
 - Proveďte rešerši metod strojového vidění vhodných pro využití pro monitorování průmyslových HMI
 - Navrhněte systém využívající strojové vidění pro monitorování průmyslových HMI
 - Navržený systém implementujte a otestujte

Seznam doporučené literatury:

M. Sonka, V. Hlavac, a R. Boyle, Image processing, analysis, and machine vision, Fourth edition. Stamford, CT, USA: Cengage Learning, 2015
R. Szeliski, Computer vision: algorithms and applications. London ; New York: Springer, 2011

Jméno a pracoviště vedoucí(ho) diplomové práce:

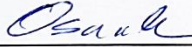
Ing. Cyril Oswald, Ph.D. U12110.3


Jméno a pracoviště druhého(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

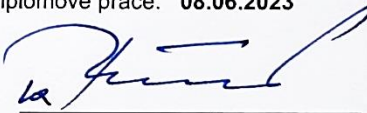
Datum zadání diplomové práce: **28.04.2023**

Termín odevzdání diplomové práce: **08.06.2023**

Platnost zadání diplomové práce: _____


Ing. Cyril Oswald, Ph.D.
podpis vedoucí(ho) práce


podpis vedoucí(ho) ústavu/katedry

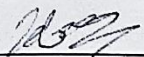

doc. Ing. Miroslav Španiel, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

3.5.2023

Datum převzetí zadání


Podpis studenta

DECLARATION

I hereby declare that I completed this thesis independently with help of my supervisor. I used only resources and literature listed in the table of bibliography.

In Prague

.....

Bc. Jakub Horák



ACKNOWLEDGMENTS

I would like to thank my thesis supervisor Ing. Cyril Oswald, Ph.D. for consultations. Also, I would like to thank the company Inovec Technology s.r.o. and especially Boris Fačkovec, Ph.D. for the interesting task and professional insight.

ANNOTATION LIST

Author name:	Bc. Jakub Horák
Název práce:	Monitorování průmyslového HMI pomocí strojového vidění
Thesis name:	Machine vision based monitoring of industrial HMI
Year:	2023
Study program:	Automation and Instrumentation Engineering
Field of study:	Automation and industrial informatics
Department:	Department of Instrumentation and Control Engineering
Supervisor:	Ing. Cyril Oswald, Ph.D.

Bibliographical information:	Number of pages	78
	Number of figures	60
	Number of tables	3
	Number of appendices	0

Klíčová slova: Digitalizace, Průmysl, Strojové vidění, Rozhraní člověk-stroj, Strojové učení, YOLOv8, Detekce objektů, Signální sloupek, Andon, Segmentový displej

Keywords: Digitalization, Industry, Machine Vision, Human-Machine Interface, Machine Learning, YOLOv8, Object Detection, Stack Light, Andon, Segment Display

Anotace: Tato práce navrhuje rámec pro sledování prvků rozhraní člověk-stroj (HMI), který řeší omezenou digitalizaci ve výrobě. Práce zavádí sjednocující přístup kombinující metody strojového učení a klasického počítačového vidění. Navržený rámec je částečně implementován a demonstruje svůj potenciál pro zvýšení efektivity výroby a slibuje zvýšení bezpečnosti na pracovišti.

Abstract: This thesis proposes a framework for monitoring human-machine interface (HMI) elements addressing the limited digitalization in manufacturing. The work introduces a unifying approach combining machine learning and classical computer vision methods. The proposed framework is partially implemented, demonstrating its potential to raise the manufacturing efficiency and promises increased workplace safety.

Table of Contents

1. Motivation.....	1
1.1. Problem formulation.....	1
1.2. Proposed solution	1
1.3. HMI description	2
1.4. Subject of the thesis.....	2
1.4.2. Localization task.....	2
1.5. Conclusion	3
2. HMI components	4
2.1. Signal lights	4
2.2. Digital displays	6
2.3. Dial meters	7
2.4. Operator panels	9
2.5. HMI components as 1D array.....	9
2.6. Conclusion	10
3. Physical constraints and influences.....	11
3.1. Positioning.....	11
3.2. Vibrations	12
3.3. Ambient lighting	12
3.4. Liquid and solid particles	13
3.5. Conclusion	14
4. Computer vision tasks and tools	15
4.1. Image capturing	15
4.2. Pre-processing	15
4.3. Object detection	17
4.4. Image understanding	18
4.5. Feature extraction.....	19
5. Methods proposed by previous work on partial and closely related problems	22
5.1. Comparison of traffic light and VSE recognition problem	22
5.2. Simple signal light and andon recognition	26
5.3. Segment display recognition.....	30
6. Approach planning.....	31
6.1. Research based analysis	31
6.2. Conceptual design.....	33
6.3. Implementation plan.....	35
6.4. Conclusion	36
7. Dataset preparation	37
7.1. Image gathering	37
7.2. Image annotation.....	38

7.3. Data augmentation	39
7.4. Dataset optimisation through selective augmentation	40
7.5. Conclusion	46
8. Detector training and performance	47
8.1. Training platform	47
8.2. Training	47
8.3. Performance	49
8.4. Conclusion	51
9. Implementation	52
9.1. Objects and instances.....	52
9.2. Packages and modules in general.....	52
9.3. Spatial object organiser (SOO)	54
9.4. YOLO as a detector and a low-level classifier	58
9.5. Camera	60
9.6. Rectangular perspective transformer	61
9.7. Linear rotator.....	66
9.8. Andon reader.....	69
9.9. Segment display reader	74
10. Conclusion	78
BIBLIOGRAPHY	79
Table of Figures	81
Table of tables	82
APPENDIX	83
The Albumentation pipeline definition	83

1. Motivation

1.1. Problem formulation

According to European Association of the Machine Tool Industries and related Manufacturing Technologies, "despite the opportunities that digitalisation can bring to manufacturing, by increasing its sustainability and competitiveness, only 5% of manufacturing small and medium-sized enterprises have networked their machinery, plants and systems across the board and only a third of them are taking the first steps in that direction or at least have concrete plans to do so." [3]

This could be due to the age of the machines: "the average age of the total installed machines in 2014 was 12.8 years and, in Germany as of 2015, the average age of computer-numerically controlled (CNC) and non-CNC machines was 10.5 and 19.7 years, respectively. Data provided to CECIMO by some machine tool manufacturers also shows that, on average, 80% of machines are still in service ten years after installation, while 65% are still in service after 20 years." [3]

Newer machines do often have the ability of machine to machine communication (M2M). This enables the utilization of SCADA (supervisory control and data acquisition) systems. Supervisory control and data acquisition are key components to efficiency and safety increase of the machine, the workers and the whole plant.

The M2M communication, however, is often restricted by the manufacturer of the machine. Not all data may be accessible, the accessible data may be unreliable. Also, the manufacturer charges extra money if the machine has M2M. For older machines without M2M, it is hard to gather any useful data. For any machine, there is always an additional cost of integration the M2M.

1.2. Proposed solution

In this thesis, a solution is proposed, using computer vision to monitor a human-machine interface (HMI). Using data from a HMI should be convenient, as all the signals are validated. Also, all the signals are simple and human-readable, so a computer should be able to read them, too. Computer vision was chosen for its space, cost, and resource efficiency.

This thesis focuses on three main points:

1. Providing a solid research-based knowledge fundament
2. Proposing a complete system design plan and a structure for implementation
3. Offering implemented solutions for the most general tasks defined in the design plan

1.3. HMI description

An integral part of every machine are its control elements and signal elements, which serve as an interface between the operator and the control devices - the human-machine interface (HMI). The control elements serve as an input to the machine accessible directly by humans, signal elements serve as an output from the machine to the human. [2]

In this thesis, only visual signal elements are in focus. Namely signal lights, digital displays and dials.

On machines with a high number of control and signal elements, an "operator panel" (OP) is present, with all the signals and displays on it. Andon light is a very common type of signal light.

1.4. Subject of the thesis

The goal of this thesis is detection, localization, classification and state recognition of specific visual signal elements (VSE), namely signal lights (with special focus on andon lights), digital displays and dials from a video. Furthermore, understanding of the whole image should be provided.

1.4.1. Detection task

Answers the question, if and which VSEs are there in a provided image or video. Additionally, whole operator panels (OPs) are detected. An accuracy estimation should be a part of the result.

1.4.2. Localization task

Output of this task is the position for every detected VSE and OP at any given time.

1.4.3. Classification task

Output of this task is a class of every detected VSE. For signal lights, the classes may be the shape of the light and the colour. In addition, if any operator panel is detected, membership of each VSE is evaluated.

1.4.4. State recognition task

Output of this task is the state of every detected VSE for any given time. For signal lights, the states might be ON, OFF, blinking (with certain frequency). For displays and dials, the state is the value displayed. Also, additional information should be recognised, such as "temporarily invisible".

1.4.5. Understanding

Data from all previous tasks are gathered and analysed to provide useful information. The analysis is highly dependent on the end application of the system proposed in this thesis. Some examples are given below:

- Machine downtime measuring
- Error logging
- Detecting predefined error states
- Production times measuring

Ideally, if the gathered data is sufficient, this would enable connecting the machine to SCADA.

1.5. Conclusion

The subject of this thesis is machine digitalisation using computer vision. VSEs are observed, performing following tasks:

1. Detecting and localising individual VSEs and OPs
2. Localising individual VSEs and OPs
3. Classifying individual VSEs
4. State recognition of each VSE
5. Building an understanding of the observed area

2. HMI components

As stated before, there are 3 main types of VSEs: signal lights, digital displays and dials. Signal lights can be further divided into 2 groups - simple signal lights and andon lights (or stack lights). Digital displays can be divided into 3 groups, segment displays (especially the 7 segment displays), dot-matrix displays and a general 2D displays. Dials can be divided to 2 groups, cyclonic dial meters and clock dial meters. [4]

2.1. Signal lights

Signal lights use light to transfer information.

There are 2 basic properties of the signal light:

- Colour
- Shape

Colour can either belong to a group of colours with universal meaning (GCwUM), or not belong to it and have an arbitrary or no special meaning. If possible, colours should be ordered from top: red, yellow, blue, green, white. The GCwUM is:

- red - emergency, danger
- yellow - abnormal state, critical state is imminent
- blue - obligation, the need for operator intervention
- green – normal
- white - neutral, can be used for any other purpose [2]

Except for red, which is used exclusively for emergency or danger indication, or used on power-off switches, these are recommendations, not obligations. So the final meaning of each signal light must be assigned in the Understanding step. [2]

In industrial applications, basic colours with universal meaning (listed above) are used. [7]

However, the colours do not have a specified wavelength, also the cameras used do not always have the same colour representation. Previous tests have shown, that for example the green light can range from truly green to cyan, yellow is often replaced with orange and a red light can actually appear orange when being lit on. This has to be taken into account, when classifying colours.

There are 3 basic states of the light:

- On - confirmation
- Off - inactive
- Blinking (with a certain frequency) - indication, warning

We can define two extra states - turning on and turning off, which are a consequence of transient effects taking place in the light. If the frame is taken during the transient process, the intensity appears to lie between the on and off states and might cause errors, especially for monitoring blinking state with low framerate. [5]

2.1.1. Simple signal light

Simple signal lights can vary greatly. Mostly, these are point or surface sources of light (simple LED) or a button with backlight. They are mostly round. Due to the high variability and nearly no standardization, it must be assumed that a simple signal light has an arbitrary shape.

Shape of the light can be basic like round, square, triangular, etc., in which case shape does not transfer any information. The shape of the light, however, can be a symbol with a special meaning.

In this thesis, three types of shapes are recognised - round, rectangular and other. Concrete meaning of each light can be assigned in the Understanding step.

2.1.2. Andon light

Andon lights, or stack lights, are specialised indicators of the machine state. The colours used are predominantly part of the GCwUM. The order is mostly red, yellow, green, blue, white. Not all the colours must be present, but the stack lights are mostly a combination of the first n elements of the list above.



Figure 1 - Andons with 1 to 5 segments [29]

The shape of an andon light is mostly a tall cylinder divided into segments with a given colour. Given the 3D nature of andon lights, the shape of the projection to the 2D camera plane can range from rectangular over elliptical to a general symmetrical shape.

2.2. Digital displays

Displays are widely used for their variability and the amount of information they are able to transfer. They can be active (the symbols are formed by lighting up) or passive (the segments are formed by blocking light - mostly LCD with or without backlight). Technically speaking, digital displays are a subset of general signal lights.

Speaking of **active segment displays**: in this thesis, each detected signal light will be examined to determine, if it is a simple signal light or a digital display. When this is decided, the two groups are not connected in any sense.

On the other hand, **passive segment displays** are also classified as rectangular signal lights, as the colour and intensity of the backlight might transfer some information.

Another consequence is, that low resolution images of digital displays could be wrongly classified as a simple signal light.

This thesis focuses on digitization of machines without M2M. Presence of a screen, especially one with colour and high resolution, implies a machine with sophisticated control unit, probably capable of M2M. Therefore, the main focus in this thesis is given to segment displays, as they are simple enough to be used on machines with simple control units.

2.2.1. Segment display

Segment displays are widely used for their simplicity. They can be active (the segments are lit up to form a symbol, mostly LED) or passive (the segments block light to form a symbol, mostly LCD with backlight).

Mostly used segment displays are the 7-segment displays. However, displays with 9, 14, 16 also are used. Moreover, there are custom segment displays with an arbitrary shape and number of segments. But for this thesis, only 7 segment displays will be taken into account.

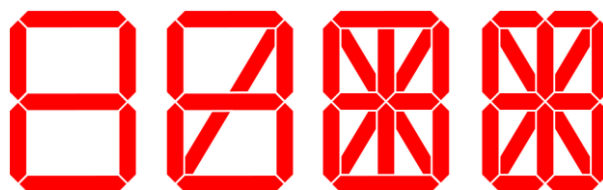


Figure 2 - Possibilities of segment arrangement in a segment display [30]

7 segment displays are able to produce simple symbols. The theoretical maximum of combinations is 128, but only 10 (11, if no segment lit up is also considered as a symbol) are mostly used in technical practice. The symbols are numbers from 0 to 9. In this thesis, the focus will be on reading numbers from 7 segment displays.

2.2.2. Dot-matrix display

The content of these displays is highly variable. Symbols of medium complexity can be displayed thus no specific techniques will be proposed to gather data from them. OCR will be used in this thesis, as a general solution for text recognition.



Figure 3 - Dot matrix display [31]

2.2.3. General 2D display

The content of a general 2D display is way more variable than other displays. It allows complex symbols to be displayed with an arbitrary structure. If the symbols displayed look like their real template, the general 2D display can be regarded as an OP and the symbols displayed as physical VSEs.

2.3. Dial meters

Dial meters are mostly mechanical, transforming some physical quantity into movement of some visible component. Based on the visible component, two groups are considered - cyclonic dial meters and clock dial meters. Both are in the focus of this thesis.

2.3.1. Cyclonic dial meter

Special case of mechanical dial meters, where there is a rotating cylinder with numbers from 0 to 9 for every number place. These meters are mostly used to measure a sum of some quantity. These displays are rarely backlit or glowing.

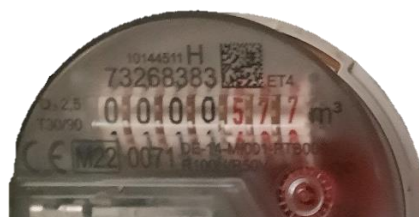


Figure 4 - A water flow meter with a cyclonic dial

2.3.2. Clock dial meter

Special case of dial meters, where there is a round dial face with a scale covering the entire range of measurement. The moving visible element is a pointer. The relative angle of the scale and the pointer determines the quantity.



Figure 5 - A manometer with a round clock dial

The pointer colour is mostly black, but can have an arbitrary colour, which is contrasting with the dial face. The dial faces are mostly white, but can have an arbitrary colour. They are commonly not backlit.

A simple output from reading such a meter would be a percentage of the range, where the pointer points. This does not transfer the information about the units, nor the exact number. On the other hand, this technique does not require any other information about the meter, therefore can be robust. This might be, however, sufficient for some cases, where only a few states are important. Those states could be 0% (turned off), 100% (overload) and 50-80% (normal). Another states might be "rising", "falling" and "steady", which are measures of change, for which the unit and concrete quantity is also irrelevant.

A comprehensive output would be an exact quantity with a given unit. This requires a high resolution image, which is not always available. Also, it might be easier to take the simple approach and to map the output to a manually defined range in the Understanding step.

2.4. Operator panels

Often, VSEs are concentrated in one place, so that the operator has everything at hand. That one place is an operator panel (OP). OPs are often visually or physically separated from the rest of the machine. They contain all VSEs of the machine, except andon lights, which are meant to be seen from all directions and thus are located on top of the machine or in a place that's higher than the machine. Nevertheless, all other VSEs of a single machine can be expected to be a part of an operator panel.

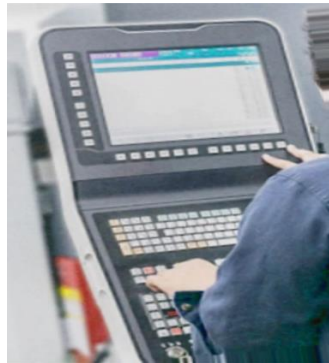


Figure 6 - An example of an operator panel [35]

The VSEs are in an arbitrary pattern, but it is often a rectangular grid pattern. This fact is important in detecting individual VSEs and whole OPs. In this thesis, an operator panel is a visually or physically separated region containing either at least one VSE and one input element or at least two VSEs.

During the classification process, individual VSEs are assigned to an OP if found. This makes the Understanding step easier, as it provides additional information about the relationship between the VSEs.

2.5. HMI components as 1D array

When a closer look is taken at the HMI components listed above, one important commonality appears among andons (also called stack lights), segment displays and cyclonic dials. They have a certain state expressed by a recurring feature.



Figure 7 - A cyclonic dial representing an 1D array of features [36]

Generally, these VSEs are one dimensional arrays of features. In this thesis, it makes sense to call those features "segments". The number of segments is prior unknown, but lies within boundaries. Every segment has a set of parameters. Those parameters can be either entirely constant or depend on the particular VSE, on the position within the VSE or the state of the VSE. These four groups of parameters are discussed below in greater detail.

- **Constant parameters** - same for any instance of the given class of VSE in a general way. Those are e.g. ratio of width and height of a segment, possible colours or the representations of a certain state. These parameters are stored in the settings files and will not be modified frequently. They might be adjusted only when fine tuning.

- **Particular VSE dependent parameters** - dependent on the particular VSE, but are the same for all segments. Those are e.g. width, height, spacing of segments, the colour used. Those parameters must be passed as an argument to the function reading the particular VSE.

- **Position dependent parameters** - important especially for segment displays and cyclonic dials, where position expresses the order of magnitude of the number in the segment.

- **State dependent parameters** - for andons, the on/off state is represented by the brightness of the segment. For segment displays and cyclonic dials, the state is the digit displayed.

2.6. Conclusion

HMI components may vary greatly, but there are still some standards. In this thesis, the aim is to monitor the following HMI components and their special properties:

1. Simple signal lights - colour, state, shape
2. Segment displays - colour, number
3. Dot-matrix and general 2D displays - displayed text
4. Cyclonic dial meters - number
5. Clock dial meters - percentage of the whole range
6. OPs - membership of other VSEs to this panel
7. Andon lights - colour, state and shape for each segment

In addition, following properties of every HMI components are monitored:

1. Position
2. Visibility
3. OP membership (for 1. - 5.)
4. Accuracy estimation for all the outputs

3. Physical constraints and influences

In this chapter, the problems of the physics of image capturing are described. The focus will be on practical problems and proposed solutions related to the subject of this thesis. Below, there is a diagram of a machine vision (MV) system describing important influences.

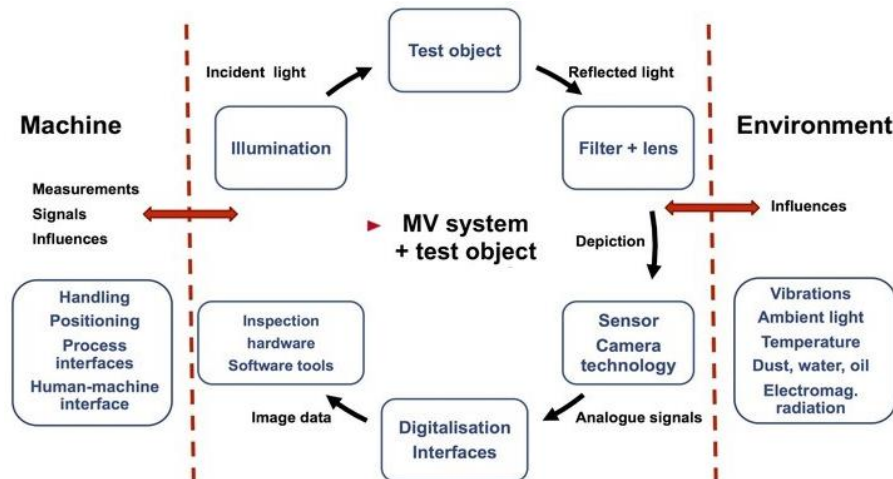


Figure 8 - Diagram of influences affecting a machine vision system [8]

On the side of the machine (the image-capturing device), the important influences are handling, positioning and process interface. The image-capturing device has optical defects caused by lenses and can be equipped with certain filters. The environment influences the scene and the image-capturing device itself by vibrations, ambient lighting, temperature, electromagnetic radiation. Additionally, there may be liquid or solid particles like dust, water or oil in the environment. [8]

Below, all influences relevant to the subject of this thesis are discussed in greater detail.

3.1. Positioning

The positioning of the camera is fairly arbitrary. However, a few rules are established for simpler object recognition.

1. The VSEs must be well visible by the camera.
2. The scene must be in an upright position with an angular tolerance of $\pm 45^\circ$
3. The camera must be in a fixed position relative to the VSEs.
4. The camera must be in a position with a low chance of objects appearing between the camera and the monitored objects, thus causing occlusion and making it impossible to determine the VSE's state.

The bigger and clearer the VSEs appear in the camera frame (equivalent to better **visibility**), the better and more certain the reading will be. This is especially the case for segment displays, dot-matrix and general 2D displays, cyclonic dial meters and clock dial meters, which are hard to read when captured from an acute angle or a long distance.

The **angle** of the frame to the horizon should not exceed a certain amount, which was set to $\pm 45^\circ$. This assumption makes the recognition simpler and is actually simple to implement. Mostly, the camera will naturally be in the right orientation. In the case of not being in the right orientation, a simple digital rotation of the frame by 90° , -90° or 180° can be done without losing any data.

The verification step assumes **no relative motion** between the camera and the VSEs. If this is not the case, the VSEs might still be detected, but with very high uncertainty and the reading might not be continuous. The movement of the background, however, is not restricted. This is important especially in a case of monitoring a VSE on a crane or other moving structure.

Occlusion is another problem that can be minimised by correct placement of the camera and camera redundancy. Occlusion causes an unknown state for all occluded VSEs. This might not be a problem for a short period of time (the reasonable maximum being units of seconds), where the validation step should fill the unknown part of the data. For longer occlusion, it is not safe to estimate the state of the VSE, thus having a missing part of data. This might be the case when monitoring OPs operated by humans.

3.2. Vibrations

In industrial environments, vibrations are often present. Vibration of the scene is a minor problem, but the vibration of the surface to which the camera is mounted can have negative effects on mechanical and optical basis. The major mechanical problem is loosening of joints causing unwanted movement of parts in the optical system of the camera and movement of the camera as a whole. The main optical problem occurs when the vibration frequency is in the same order of magnitude with the sampling frequency of the camera, which is often the case. This causes linear or partially circular blur.

3.3. Ambient lighting

In industrial environments, the ambient lighting consists of flood lights, the Sun, individual lights and VSEs. In most cases, during working hours, the inside of an industrial plant is illuminated by **flood lights** with fixed intensity and position. If there is any variation, it is known and can be a part of the recognition model.

Other common source of light for industrial environments is the **Sun**. Outdoors, it is often the only source. Indoors, the sun can shine through windows, creating less predictable lighting conditions. However, some basic knowledge about the movement of the Sun can be provided to the recognition model.

Individual light sources may include vehicle headlights, personal light sources and light sources on individual machines. These are often human operated and therefore hard to predict. Their intensity, however, is mostly lower than the flood lights and the Sun.

VSEs are the sources with lowest intensity, but with a high variety of colour and other properties. These are the sources this thesis is trying to recognise.

3.4. Liquid and solid particles

Industrial processes are often based on processing a matter of a certain type. When transporting or processing a liquid or solid matter, small particles may be emitted forming mist or dust. If the camera is in contact with the mist or dust, particles remain on the transparent cover of the camera lens. This effect worsens with the time of being exposed to the mist or dust. Also, bigger particles and droplets may land on and attach to the camera. When using auto-focus on the camera, the scene might get out of focus. Below are some specific problems that can emerge in certain scenarios.

Assuming uniform coverage by particles of dust and liquids, in the early stage, the consequences are change in hue and blurring. In the mid stage, blurring is stronger and a notable fall of brightness is present. Additionally, in the late stage, parts of the image become occluded (brightness falls below the camera range).

Assuming a singular translucent particle (or droplet) of a relevant size, the effects are a change of optical geometry, change in brightness and hue of a certain region. This may result in recognition with high uncertainty, lost detections and misdetections.

Assuming a singular opaque particle (or droplet) of a relevant size, a part of the image is occluded, resulting in lost detections.

All of these problems are fixable after they emerge. The importance of these problems ranges from nearly zero to high.

3.5. Conclusion

In this chapter, physical constraints and influences were described. Moreover, some rules were established. Below is the list of these rules.

3.5.1. Physical constraints and influences

1. Vibration of camera with a frequency near to the camera's sampling frequency results in linear or circular blur.
2. Vibration of camera may result in change of its position and changes of optical geometry inside the camera.
3. The lighting in an industrial plant consists of highly predictable flood-lights, semi-predictable sunlight, unpredictable individual lighting and VSEs.
4. On a camera lens or shield, a continuous layer of fluid or dust can form, resulting in malfunctioning auto-focus, blur and change in hue and brightness including total occlusion.
5. On a camera lens or shield, a singular particle or droplet can result in malfunctioning auto-focus, change in optical geometry, change in hue and brightness including occlusion.

3.5.2. Rules and assumptions

1. Strong vibration of camera must be avoided.
2. The VSEs must be well visible by the camera.
3. The scene must be in an upright position with an angular tolerance of $\pm 45^\circ$.
4. The camera must be in a fixed position relative to the VSEs.
5. The camera must be in a position with a low chance of occlusion lasting more than a few seconds.
6. Dusty and misty environment is acceptable, as long as the particles are smaller than the resolution of the camera.
7. The camera must be placed far from sources of droplets and large particles enough to not be contaminated.

4. Computer vision tasks and tools

"Image understanding by a machine can be seen as an attempt to find a relation between input image(s) and previously established models of the observed world." [1]

Computer vision is a field of science combining multiple fields. Capturing the input image relies on engineering, optics, and other fields of physics. Defining and implementing the models is a task for computer science. Defining principles of the model relies on mathematics and statistics. Other fields like psychology and neuroscience bring knowledge about how human brain works. Models based on machine learning and artificial intelligence are also widely applied. Implementing those models relies on software and electrical engineering. [1]

This thesis is focused on development of a model for a problem defined in the chapter 1.5. **Conclusion.** Previous results on similar subtasks are combined to create a new model, that will be applied in reality. Because of that, the developed model must be robust, so light conditions and light properties must be taken into account. Thus, a little attention is also given to physics behind the image capturing. [1]

4.1. Image capturing

A photo is a 3D scene mapped to a 2D camera plane and sampled into a matrix and quantized. The matrix is 2D for grayscale image and 3D for colour image. [1]

In this thesis, only a little attention is given to this topic, as image capturing is well established and implemented.

4.2. Pre-processing

Pre-processing is a set of low-level image processing techniques, that use very little knowledge about the content of the image. It takes an image function (commonly brightness) as an input and performs pre-processing tasks like compression, reshaping, noise filtering, edge extraction and image sharpening. Low-level tasks also can take inputs from higher-level tasks as a feedback. [1]

Practically, it is a set of operations done at the lowest level of abstraction. Both input and output are intensity images. Pre-processing typically decreases the entropy of irrelevant features, but tries to maintain it for the relevant ones. It's goal is to suppress irrelevant features (typical example: noise filtering, downsizing), enhancing the relevant ones (typical example: edge sharpening) and preparing the image for upcoming tasks (typical example: rotating, colour-space transformation). [1]

Low-level processing techniques are almost the same as digital-image processing techniques, which are well established. The well established low-level techniques are discussed and used, but the details of their operation are not discussed in this thesis. [1]

Pre-processing techniques can be divided into four main categories:

1. pixel brightness transformation
2. geometric transformation
3. local pre-processing
4. image restoration [1]

Pre-processing methods can be divided into three categories based on the knowledge used:

1. methods using little to no knowledge
2. methods taking into account the properties of the device capturing the image
3. methods using knowledge of the objects being recognised [1]

Pixel brightness transformation

A pixel brightness transformation is a function applied to a pixel which returns a new brightness of the pixel. Based on the function inputs, there are two categories:

Brightness correction - assigns new brightness to a pixel based on it's original brightness and position in the image. Brightness correction can compensate for systematically uneven illumination and known device's optical defects such as vignetting. [1]

Gray-scale transformation - assigns new brightness to a pixel based only on it's original brightness. Brightness thresholding and histogram equalization are the most used transformations. [1]

Geometric transformation

Compensates for geometric distortion. It is a vector function mapping a pixel to a new position. As the new positions aren't likely to end at the same spots, interpolation of the neighbouring pixels is done to sample the image of the desired size. The 3 basic geometric transformations are rotation, scaling and skewing. [1]

Local pre-processing

It can be divided into two categories:

Smoothing suppresses noise and small details, blurs sharp edges. It is equivalent to a low-pass filter in frequency domain. Two typical algorithms are averaging and median filtering. [1]

Gradient operators use local derivatives. It highlights places with high change of the image function and also rectifies noise. In the case of the image function being brightness, first order derivative operator acts as an edge enhancer. It is equivalent to a high-pass filter in frequency domain. [1]

Image restoration

Methods aiming on compensating for image degradation using knowledge about the whole picture. There are many types of degradation and even more methods for repairing it. [1]

There are some typical examples this thesis will be considering: defects of optical lenses, non-linearity of the electrooptical sensor, relative motion between an object and camera, wrong focus. Another important type of degradation is a perspective warp. [1]

Based on the type of knowledge we have about the image and on the image quality, there are two categories:

1. Deterministic - applicable to images with little noise and known degradation function.
2. Stochastic - degradation function is estimated using statistical tools [1]

Knowledge needed for image restoration can be:

- a priori - knowledge about the device or gathered from comparison of previously captured images and sample images without degradation - knowledge obtained before the image to restore is captured
- a posteriori - knowledge is gathered from the degraded image - after capturing it. [1]

4.3. Object detection

Object detection is the primary form of high-level image processing used in this thesis. It is based on knowledge and goals. It tries to imitate how human vision works. Artificial intelligence based models are widely used. [1]

"High-level vision begins with some form of formal model of the world, and then the 'reality' perceived in the form of digitized images is compared to the model. A match is attempted, and when differences emerge, partial matches (or sub-goals) are sought that overcome them; the computer switches to low-level image processing to find information needed to update the model." [1]

Object detection is a key subject of this thesis, therefore it is discussed in great detail.

4.3.1. Semantic segmentation

It is a task of dividing an image into parts with strong connection to objects. **Complete segmentation** divides the whole image into disjoint parts, while **partial segmentation** may generate overlapping regions that do not cover the whole image. **Semantic segmentation** task is about dividing an image into semantically meaningful regions. A label is assigned to each pixel. Each label is an element of a predefined set of labels. As a simple example, the sets of labels can be background and candidate. [1]

4.3.2. YOLO

There is a large variety of object detection approaches. The leading solutions today are CNN and the YOLO architecture.

You only look once (YOLO) is a state-of-the-art AI model for object detection. YOLO was built with the aim to solve multiple vision tasks at once, notably object detection, image segmentation, and classification. Its end-to-end design for the processing of images made it stand out from the competition, offering both efficiency and effectiveness. It has the best combination of speed and accuracy. It is pretrained to detect objects of the COCO dataset, which includes vehicles, animals, accessories, sport- and kitchen-ware, food, furniture and other categories. The number of categories is 91.[9,10]

Some categories, which are interesting for our thesis: laptop, remote, keyboard and clock.

The latest iteration, YOLOv8, was launched in January 2023 by Ultralytics. This new version introduced an array of five scaled versions to cater to varying computational requirements, from YOLOv8n (nano) to YOLOv8x (extra large). In this thesis, the YOLOv8m (medium) is used. [10]

In addition, YOLOv8 offers practical implementations for end users. It can be run from a command-line interface, installed as a PIP package, and offers multiple integrations for labelling, training, and deploying. In an evaluation using the MS COCO dataset, the YOLOv8x version achieved an impressive AP of 53.9% with an image size of 640 pixels, outperforming the 50.7% of YOLOv5, while operating at a speed of 280 FPS on an NVIDIA A100 with TensorRT. [10]

4.4. Image understanding

Image understanding refers to the ability of a machine to interpret and extract meaningful information from an image. Interpretations are hypothetical, they may be ambiguous and there

can be a large number of them. "The main task is to define control strategies that ensure an appropriate sequence of processing steps." [1]

4.5. Feature extraction

Machine learning, being a robust and general solution for many problems, comes with the cost of a lot of necessary annotated data. Detection of different objects in scenes with varying influences is one of the tasks where machine learning outperforms other methods.

For simpler and better defined problems, however, machine learning could be an unnecessarily complex solution. Classical methods of computer vision can handle these tasks better - they are often easier to implement, are more reliable and robust. The main focus in this thesis, when it comes to classical methods of computer vision, is given to feature extraction.

Man-made objects, especially in industry, have simple shapes, due to their simplicity of manufacturing. Straight lines, rectangles and circles are the three main geometric forms used in industrial design.

4.5.1. Line detection

Straight line defines direction of an object. They also indicate presence of an edge. In this thesis, detecting straight lines is done exclusively for determining the orientation of an object. The tool used is an OpenCV implementation of **hough transform**: `cv2.HoughLines`. [11]

The input of the function `cv2.HoughLines` is a binary image. Best results are achieved when passing an edge map. The function transforms each point from image space to a line in hough space, where the x coordinate of the point becomes the intersection of the line with y axis and the y coordinate becomes the slope of the line. Then, intersection of the lines are calculated and grouped. Groups are then inverse-transformed from a point in hough space into a line in image space. The number of intersections in the group is the weight of the line. [11]

When good pre-processing techniques are accommodated, the `cv2.HoughLines` proved to be a robust line detection algorithm.

Therefore, the function can be used to rotate detected objects to remove the influence of rotation. This can be useful for objects with long straight lines and not influenced by perspective. It can be used for andons, segment displays, general displays, cyclonic dial meters and clock dial meters with rectangular frame.

4.5.2. Perspective detection

Rectangles, when projected on the camera plane, will generally form a convex quadrilateral. If the deformation is low, it can be neglected and a rotation might be enough. But generally, rectangular object like general displays, segment displays and especially operator panels will be deformed, making it hard to correctly read their state.

The Xiaohu Lu vanishing point detection

For the perspective warp detection, a vanishing-point algorithm is used. A python implementation of Xiaohu Lu vanishing point detection called **lu-vp-detect** is used. Below, the code snippet shows the use of that implementation to detect vanishing points in an image and to show the debug image generated by the `create_debug_VP_image()` method. [12,13]

```
# importing the VPDetection class from lu_vp_detect library
from lu_vp_detect import VPDetection

# creating the detection object
vpd = VPDetection(length_thresh, principal_point, focal_length, seed)

# trying to detect the vanishing point
try:
    vpd.find_vps(image)
except:
    print('lu_vp_detect algorithm failed while finding vanishing points.')
else:
    # showing the debug image
    vps_debug_image = vpd.create_debug_VP_image(show_image=True)
    # getting the vanishing points in 3D space
    vps_3D = vpd.vps
    # getting the vanishing points in the 2D space of the image
    vps_2D = vpd.vps_2D
```



Figure 9 - A flow monitor with a segment display with detected lines by the *lu_vp_detection* algorithm [32]

The debug image shows lines that were used for vanishing point detection. They are divided by colour indicating which vanishing point were they used to detect. In this case, the green lines were used to detect the left vanishing point and the red ones to detect the vertical vanishing point. [12]

In this case, the detected vanishing points in the 2D space of the image are:

$vertical = [106302, 8604349]$

$R = [196, 196]$

$L = [-43874, 740]$

5. Methods proposed by previous work on partial and closely related problems

Surprisingly, no other work focusing on the comprehensive monitoring of an OP using CV was found during the first search. The search query was defined as follows: {"operator panel" OR "control panel" OR "HMI"} AND ("computer vision" OR "reading" OR "camera"). The search was done on Google, Google Scholar and IEEE Xplore search engines.

As a paper from 2020 says, "Despite being a relevant issue, the literature on industrial signal light detection or classification is scarce. A search on IEEE Xplore, ScienceDirect and SpringerLink (the major sources of publications in engineering) only produces one result". [5]

Although there is not much available on the exact same topic this thesis has, a lot of work was done on partial and closely related problems. These are described in detail below.

5.1. Comparison of traffic light and VSE recognition problem

Because of the huge interest in driving safety and automation, especially self-driving, there is a huge amount of work available on traffic light recognition. A traffic light is technically a set of (mostly 3) mostly round simple signal lights with red, yellow and green colour. A traffic light might also be regarded as an andon light with a non-standard shape. Traffic light recognition is a sub-problem of the problem defined in 2.1.1. Simple signal light

One highly usable dataset was found. It contains traffic light images annotated with colours and bounding boxes around individual lights, which makes it different and way more usable than other datasets, where whole traffic light cases are annotated. It is also free, even for commercial use. [14]

Given the level of coverage of this topic, it is desirable to adapt it to cover most of the problem defined in 2.1.1. Simple signal light. Below, a few ways of doing so are analysed.

5.1.1. Colour

Traffic lights have three colours - red, yellow and green. The [14] dataset was analysed for representation of individual colours. For comparison, 50 photos of industrial plants and operator panels with 171 occurrences of signal lights were also analysed. It should represent the reality this thesis is trying to understand.

colour	Traffic light detection dataset	Photos from industrial plants
red	57 %	44 %
yellow	2 %	14 %
blue	0 %	1 %
green	41 %	38 %
white	0 %	3 %

Table 1 - Comparison of colours occurring in the reality of industrial plants and in the "Traffic light detection dataset"

Based on the results in the table above, a following conclusions were drawn:

1. Red and green are most frequent colours and together form an overwhelming majority in both datasets.
2. Yellow is underrepresented in the Traffic light detection dataset compared to the reality of industrial plants.
3. Blue and white are the least frequent colours in the reality of industrial plants, but they are not represented in the Traffic light detection dataset at all.

5.1.2. Shape

Traffic lights are mostly round. Some traffic lights, like pedestrian-shaped and arrow-shaped lights, have an arbitrary shape. Rectangular lights, however, are not present in traffic lighting.

That seems to be a good representation of the reality this thesis is trying to understand. An algorithm capable of detecting arbitrary shaped lights should also be able to detect rectangular lights, which technically are a sub-category of arbitrary shaped lights.

5.1.3. Close neighbourhood

The close neighbourhood of an individual light is almost always the black casing of the whole traffic light. On the operator panel, simple signal lights often have a metal or black plastic rim. In the special case of andons, there is a slight similarity in the close neighbourhood. Otherwise, the close neighbourhood can't be regarded as similar for a traffic light problem and a general industrial light problem.

5.1.4. Background

The light with its close neighbourhood forms a hardware unit, which is placed on a background. Traffic lights, as they are often situated in high places, often have sky as a background, which is fairly uniform. Another large portion of traffic lights is situated in urban areas, which might mimic the industrial environment sufficiently. Industrial signal light hardware units are also situated either on a uniform background when being part of an OP, or on an arbitrary place with industrial plant environment as a background. OPs mostly have light grey colour. This, however, is not a rule.

5.1.5. Headlights, taillights and turn lights

As the pictures are from roads, there is a lot of cars. Cars also have signal lights - white or yellowish headlights, red taillights and orange turn lights. These are not a part of the annotations, but the detector described in this thesis should be able to detect them. This disparity will result in a lot of false positives when using the industrial signal light detector on the [14] dataset. The amount of false positives does not change the recall/sensitivity. Therefore, as long as recall/sensitivity is used as the metric, the "falsely" detecting headlights, taillights and turn lights doesn't play a major role.

5.1.6. Summary

1. Through colour transformation, it is possible to adjust existing traffic light models and datasets to detect all colours listed in 3.1. *Signal lights*. The colour change of the background should not be a problem, as the background colour is arbitrary. However, it is good to preserve shades of grey, as that is a colour often used in industrial applications. Also, it is important to keep black and white colours, which seem to play a key role in light detection and state recognition.
2. If rectangular shaped lights are not regarded as a disjunct set, but instead a subset of arbitrary shaped lights, traffic lights can be considered the same as industrial signal lights.
3. Close neighbourhood seems to be very different when comparing traffic lights and industrial signal lights. Therefore, if existing traffic light detection algorithms are used, they must not use the close neighbourhood of the light to detect the light. In case of neural networks, however, it is hard to determine features used for detection. Therefore, the above rule is relaxed to a degree where existing traffic light models and datasets can be used, but with following assumptions:
 - 3.1. The portion of traffic lights in a dataset used for learning/tuning an algorithm must be less than the maximum expected error. Otherwise, the learned/tuned algorithm must

be tested specifically whether close neighbourhood is used to detect the light. The outcome of the test is then used to determine the overall error of the algorithm.

- 3.2. The portion of traffic lights in a dataset used for testing any algorithm must be near 0.
4. The little differences in the background of the hardware units for both environments seem to not play an important role.
 5. An algorithm meeting the objectives of this thesis will generate a lot of false positives when used on data from traffic. Either labels should be added for head- , tail- and turn lights, or recall should be used as a metric. When applying the recall-as-a-metric solution when learning a neural network, the traffic light data should form only a minority of the dataset to minimize the effect of not rewarding the NN for detecting head- , tail- and turn lights.

The rule 3.2. is actually a general rule, that ensures that the assumptions and rules above are correct and complete. The amount of traffic lights in the testing dataset, however, should not be exactly 0, as there are traffic lights used in industrial plants.

5.1.7. Conclusion

Given the information above, the traffic light detection problem is similar to an industrial signal light detection problem with the following assumptions:

1. Colour transformation of input image is used to account for blue and white coloured lights.
2. Colour transformation of input image is used to set the proportions of individual colours to the reality of industrial plants.
3. The colour transformation should preserve shades of grey and must keep black and white colours.
4. Rectangular shaped lights are regarded as a subset of arbitrary shaped lights.
5. The portion of traffic lights in a dataset used for learning/tuning an algorithm must be less than the maximum expected error. Otherwise, the learned/tuned algorithm must be tested specifically whether close neighbourhood is used to detect the light. The outcome of the test is then used to determine the overall error of the algorithm.
6. The portion of traffic lights in a dataset used for testing must be near 0.
7. Either labels should be added for head- , tail- and turn lights, or recall should be used as a metric. When applying the recall-as-a-metric solution, the traffic light data should form only a minority of the dataset.

After utilizing all the above assumptions, the traffic light recognition problem becomes the same problem as industrial light detection problem. From here, the distinction of traffic lights and industrial signal lights will be regarded as irrelevant. Exception being fulfilment and verification of the assumptions above.

5.2. Simple signal light and andon recognition

After accepting the assumptions from the previous chapter 5.1.7. Conclusion, several suitable papers were found. A short overview is given below.

5.2.1. Overview

- [15] proposes a three phase detection algorithm - extraction, pruning and validating ROIs.
- [17] proposes a dynamic exposure optimisation algorithm. It uses a fuzzy-logic-based colour-clustering segmentation algorithm for traffic light detection.
- [16] proposes a three phase detection algorithm - filter processing, labelling process, rules. Those phases can be further divided into: ROI detection, erosion, dilation, segmentation, flood fill, bounding box localization and filtering based on rules.
- [18] proposes a 4-step algorithm - bounding box detection, classification, tracking, verification. YOLO is used as the detector. A small NN is used as a classifier. Object tracking across multiple frames ensures continuous detection and instance tracking.
- [19] proposes a two phase detection algorithm - pixel-wise semantic segmentation and candidate classification. Those phases can be further divided into: Colour based pre-segmentation, texture based segmentation, regions based segmentation, geometric and colour feature classification, tracking.
- [5] proposes a 3phase algorithm: pre-processing, detection and classification. The pre-processing steps are downsizing and splitting into sub images. Detection is done by YOLOv2. Classification is done by CNN.
- [20] a three phase algorithm - detection, classification, ROI assembly, tracking/filtering. Detection and classification are done in one step by colour segmentation of red and green. ROI assembly is done by applying multiple geometrical restrictions and generating additional ROI for yellow. Brightness classifier is used to determine activation of the light. Circular buffer of state is kept to detect blinking.
- [21] uses YOLOv5 for light detection and classification. This work focuses on round lights and combines datasets containing traffic lights and industrial signal lights.

5.2.2. Algorithm based on previous work

The algorithm this thesis is developing performs the following tasks, based on the chapter 1.5.

Conclusion:

1. Detecting and localising individual VSEs and OPs
2. Localising individual VSEs and OPs
3. Classifying individual VSEs
4. State recognition of each VSE
5. Building an understanding of the observed area

After analysis of the previous chapter 5.2.1 Overview, these are the steps most of the proposed algorithms share:

1. pre-processing
2. segmentation
3. detection
4. classification
5. verification

Additionally, a 6. step, Understanding, is added. As the researched work only dealt with a single problem at a time, no wider understanding of the image was proposed. However, for the case of this thesis, it is an important step. Below, each step will be discussed in detail.

Pre-processing

Input of this step is the captured image. The output is an image of a given size, orientation and colour model.

Based on the 2.2.1. Pre-processing chapter, pre-processing step makes 4 main steps - pixel brightness transformation, geometric transformation, local pre-processing and image restoration.

Geometric transformation is the first step. In this step, images should be scaled to a certain fixed size, rotated and compensated for known geometric distortion. The outcome of this step should be a normalised image, no matter the geometrical diversity of the input. Part of this step is also splitting into sub-images, suggested by [5].

Pixel brightness transforming is the second step. As this thesis deals with colours, and calculating differences between colours is an inevitable step, [1] suggests using LAB or LUV colour space. These are device-oriented colour spaces suitable for analysis and determining

colour differences. [6] leverages the usage of YUV colour space, which is fairly similar to LUV colour space. In this thesis, Luv is the colour space of choice. Another task is the brightness correction compensating for known defects of the image-capturing device.

Local pre-processing is the third step. The image is smoothed to suppress noise.

Image restoration is regarded as the fourth step. If there is a distortion caused by the optical lens, relative motion between the object and the camera or wrong focus, it is repaired in this step. Also, exposure optimisation is a part of this step. Based on the knowledge of the whole image, it is possible to adjust the exposure time of the camera to avoid over- and under-exposed regions of interest.

These results, however, are universal and can be applied when recognising any other class.

High-level image processing - segmentation, detection

The input of this step is a pre-processed image. The output are the ROIs with labels and certainty of detection.

Segmentation and detection are tasks of high-level image processing. Two main approaches are proposed by researched work: YOLO architecture and colour segmentation.

A pretrained YOLO model will be tuned for categories defined in 2.6. **Conclusion.** The input is an image, the output is a label, bounding box and confidence for each object detected. When YOLO is used, the objects are detected first and the image is segmented into rectangular regions given by the bounding boxes.

Detection by colour segmentation first utilises colour segmentation to find regions of a certain colour. These are considered to be the detected objects. One big disadvantage compared to a YOLO model is the dependence of a certain colour to be present. It works good for lights captured using correct exposure time. It is not able to detect any other category. Therefore, it is not an algorithm preferred for universal detection. However, it is used for partial detection tasks.

When using YOLO, high-level image processing described in this chapter will actually be the same for recognition of all classes described in 2.6. **Conclusion.**

Classification

The input of this step are detected ROIs with labels and certainty of detection. The output is a list of objects with their specific properties.

In this step there are two main objectives. First is to filter out false positives, the second is to determine the value of each property from 2.6. Conclusion. This is applicable for all classes. However, the way of doing so will be unique for each class.

For simple signal lights, the properties are colour, state and shape. One solution of a classifier is a small NN. Other proposed methods are averaging of pixel values to get colour and state, and template matching to get the shape and filter out false positives.

Verification

The input of this step is a list of classified objects with their specific properties. Other inputs for enhanced verification can be passed - time, data from another camera, data from other sensors. The output is a consistent validated information about each detected object.

This thesis will focus on a single camera setup with no additional sensors.

The verification step assures consistent outcomes across time. Also, additional properties of each object are determined as described in 2.6. Conclusion. These properties are position, visibility, OP membership and accuracy estimation.

It is assumed that the observed VSEs are static in the scene and the camera also is static. However, it is expected that the scene is changing dynamically - moving vehicles and people, differences in lighting, occasional movement of the VSEs or the camera itself.

Position is greatly dependent on the detected bounding box. However, the bounding box might contain pixels not belonging to the detected object. Also, certain objects are better represented by a bounding circle or another shape.

Position consistency across time is assured by **tracking** the VSEs. Tracking requires storing a certain number of records. Together with the information about the current time, the tracking algorithm is able to filter outliers and adapt to slow changes in the scene. Tracking allows to define specific object instances observed across a long period of time. This is useful for consistently outputting the parameters. Also, it enables to determine the "Visibility" of an object instead of creating a new instance every time a vehicle crosses the field of view.

Analysing the records also allows for **blinking detection**, which is important for signal lights.

The tracking algorithm is described in [6]: The records are stored in a circular buffer. The size of the buffer is determined by minimum frequency to be captured and also the desired inertia of the tracking. A distance based tracking is performed across multiple frames. In this step, an accuracy estimation is generated for each object instance.

Understanding

The input of this step are verified and consistent information about all detected object instances. The basic output of this step is a database containing historical data. Furthermore, user defined metrics and functions can be an output of this step.

This step is highly application-dependent. In this thesis, generation of a database containing historical data and several example applications are covered.

5.3. Segment display recognition

In [22], a seven segment display reading technique based on HSV colour slicing is proposed. It is stated, that due to background clutter, OCR techniques are not suitable. This work focuses on a lightweight approach for real time monitoring.

The [22] focuses on red LED displays, which are the most common. Other common technologies are LCD displays, sometimes backlit by a certain colour, and white LED displays. LED displays with other colours are rare. The characters of a seven segment display have a fixed font. The only variation is the angle of the vertical axis of the segments, which can approximately range from 0 to -15° .

The algorithm proposed is divided into two stages: detection and recognition. Both will be useful in the context of this thesis. For rough detection, YOLO is used. However, for fine detection, which is crucial for reading, the detection proposed in this work will be used.

The **detection stage** consists of four sub-steps: HSV conversion, predefined HSV colour slicing, object detection, preliminary filtering. Also, each foreground pixel must also have a certain value of the value channel. These steps are equal to colour segmentation. The resulting regions are filtered based on width, height and pixel number.

There are concerns about the robustness of the recognition technique used in this work. The recognition is done by calculating the summation of the intensity in vertical and horizontal directions. This technique, however, is a symmetrical transform. The characters 2 and 5 are also symmetrical. It is supposed, that this technique performs poorly when distinguishing between those two numbers, and is consequently not used.

A part of this work also is a dataset ([23]) of two segment displays showing every 4 digit number. A part of this dataset was used in this thesis.

6. Approach planning

Based on the research from 5.2. Simple signal light and andon recognition, which provided a framework of an object recognition algorithm primarily for signal lights and andons, but turned out to be a general approach to many recognition tasks, the approach to the subject of this thesis is proposed.

This chapter is a start of the "practical part" of this thesis. It does not only describe the end results, but also the partial results and iterations. It also contains ideas, which turned out to be useless in the end.

6.1. Research based analysis

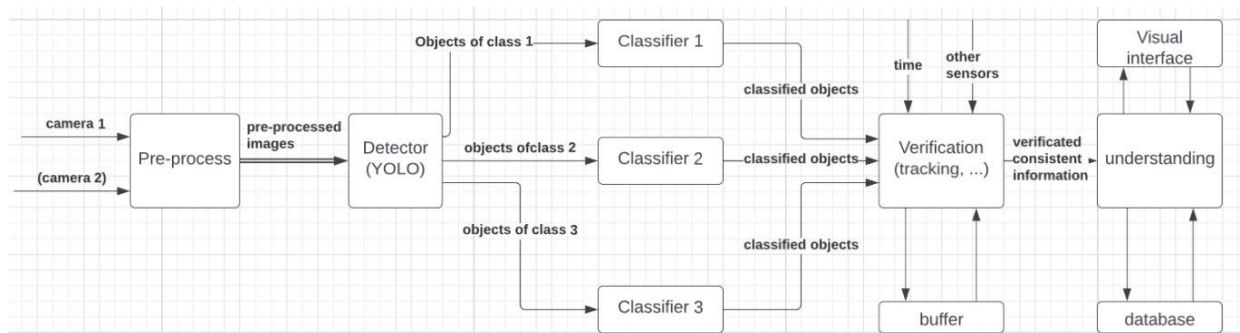


Figure 10 - A diagram of research based analysis of the monitoring system

The main steps formulated in chapter 5.2.2. Algorithm based on previous work are visualised in this flow-chart. This is the first iteration of the flow-chart, showing a rough structure, providing a general idea of how the whole system should work.

The **detector** detects and classifies objects in the images from the camera. Then, for each class, there is a separate **classifier**, which accepts a crop of the original image with the object in it. Each classifier provides high-level classification: state and confidence in its prediction. This information flows into a **verification** process, which also accepts time and other sensor measurements. It also communicates with a buffer with recent history. This block is responsible for comparing results with history and with each other, weighting them based on their confidence and for filtering and completing the data. Output of this block is a continuous flow of information about the VSEs in the camera's field of view. The **understanding** block interprets the information from the verification block. It gives values meaning, saves all useful data in a database and creates a visual interpretation of the data.

6.1.1. Detection, Localisation, low-level classification

Based on the research of traffic light detection and classification in chapter High-level image processing - segmentation, detection, there are two main approaches - colour segmentation and the use of machine learning. As the HMI components being detected in this thesis are more complex than traffic lights and mostly with no prior known colour, the latter approach was chosen

For object detection, localisation and classification, the state-of-the-art method is the YOLO (you only look once) convolutional neural network architecture. At this time (beginning of 2023), the latest publicly accessible release was YOLOv8 from Ultralytics. In this thesis, the YOLOv8m (medium complexity of the model) is used.

The main task of YOLO is to detect objects in an image, create a bounding box and classify the object with one of 7 classes defined in chapter 2.6. Conclusion. Therefore, it fully solves the detection and localisation task. It solves the low level of the classification task, as it recognises the type of the object, but does not recognise additional properties.

6.1.2. High-level classification, state recognition, low-level verification

Based on the research of various detection and classification tasks in chapter 5.2.2. Algorithm based on previous work, the next step after detecting is classification and verification.

This step is highly individual for each class. However, there is a common scheme among all of them: high-level classification, state recognition and low-level verification. **High-level classification** recognises additional information about the VSE like colour, number of segments, precise position of sub-elements, etc. **State-recognition** gives an estimate of the state. State can be activation, number, etc. **Low-level verification** combines confidences of each involved step and compares them with a threshold.

The tools used for high-level classification and state-recognition in this thesis are mostly simple classical computer vision feature extraction algorithms described in chapter 4.5. Feature extraction. Additionally, thanks to many of VSEs having a saturated colour, colour segmentation might also be a useful technique.

6.1.3. Understanding

This step is discussed for completeness. It is highly usecase-dependent. In this thesis, the focus is on developing a framework for VSE reading, not an application for a particular usecase. Therefore, it is involved in all the planning stages, but will be left as a blank building block to be adapted to a particular application.

6.2. Conceptual design

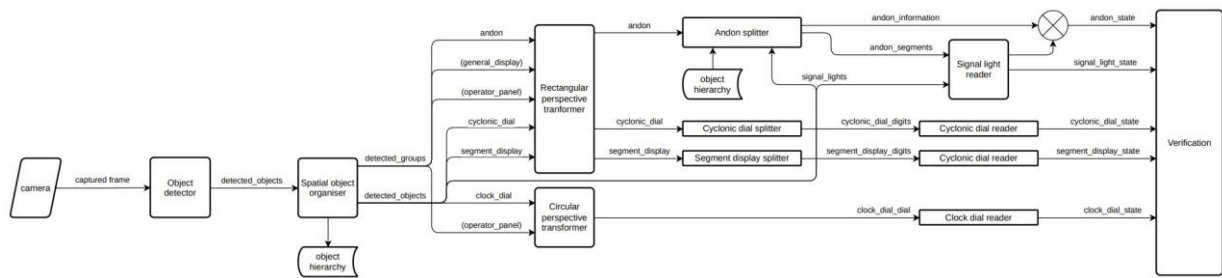


Figure 11 - A diagram of conceptual design of the monitoring system

Based on the first iteration described in chapter 6. Approach planning, a finer, more complex and programming-oriented flowchart was developed. It is focused on the whole system from a development point of view. Therefore, every block of the flow chart represents an object programmed with python. Compared to the first version, a few blocks were added, they are discussed below.

6.2.1. Camera

This simple python object serves as a simple interface between a HW camera and other objects. It is responsible for opening a camera with certain parameters based on user settings. It also prepares the image for other blocks by resizing it.

6.2.2. Spatial object organiser and object hierarchy

Spatial object organiser is a python object containing all VSEs ever detected by the Object detector. It is responsible for organising those objects, it has two main functions - hierarchical arrangement of detected VSEs and tracking. Therefore, it acts as Verification and buffer from previous approach.

The **hierarchical arrangement** is an important task of assigning base objects to group objects. Group objects are the ones, that do not have a certain value on their own, they contain other basic or group objects which determine the state of the super object. In the case of this thesis, group objects are:

- Andons - contain one or more signal lights.
- General displays - they are a special case, they can contain virtually anything, they are even less normalised than physical operator panels. However, they still can contain some of VSEs as discussed in 2.2.3. General 2D display.
- Operator panels - contain any other objects, even another operator panel.

Output of hierarchical arrangement task is an updated **object hierarchy** object. The object hierarchy object assigns a group membership to every detected VSE. This information can be used for filtering detections (e.g. operator panel with containing no VSE), data completion (e.g. one missing object in an array), validation (e.g. certain number of detected signal lights inside an andon) and understanding (e.g. grouping information of two different operator panels in one image).

The second main function is **object tracking**. It is implemented in a simple form - when a new VSE is added, the Spatial object organiser first looks for a duplicity. If the VSE being added is unique, it is saved and an id is assigned to it. If the VSE being added is already present in the same place with the same properties, the information about this VSE is updated.

6.2.3. Rectangular perspective transformer

For a simple high-level classification and state recognition, it is useful to remove as much degrees of freedom as possible. The geometrical variations are one of the most influential and easy to remove at the same time. Object detection removes the variation in position, but variation in rotation and perspective transformation remain. The **Rectangular perspective transformer** python object solves this task. Using the tools described in chapters 4.5.1. Line detection and 4.5.2. Perspective detection, this object rotates the VSE to a normalised orientation and warps the image to account for perspective if necessary.

This process works for linear and rectangular features. A linear feature is such a feature, which has a strong main direction and a weak or undefined perpendicular direction. This is often the case for features discussed in 2.5. HMI components as 1D array. For such features, it is very hard and not useful to find perspective, simple rotation is enough. In practice, those are very thin rectangles (like cyclonic dials) or cylindrical shapes projected close to perpendicular to the main axis (like andons).

6.2.4. Circular perspective transformer

The objective is very same to the object described in 6.2.3. Rectangular perspective transformer, the only difference being the application on circular features instead of rectangular or linear. As most industrial features are either linear or rectangular, this thesis does not focus on this subject. However, for further development, it is also important to detect even circular features, as the majority of dial meters and some smaller operator panels are round.

6.2.5. Andon, segment displays and cyclonic dial splitter

Based on the analysis in chapter 2.5. HMI components as 1D array, splitting the VSE into segments is a good step. Thanks to the similarity between the particular segments, one specialised algorithm can be than used to read each segment.

This function takes a normalised crop as an input. It analyses the input, determines the number of segments and splits the crop. Output of this function is a list of cropped segments.

6.2.6. Signal light, cyclonic dial, segment display and clock dial reader

After extracting elements from each detected VSE, there is a reader for each of them. The aim of the reader is to determine the state of the element. In this thesis, basic cv methods are used for reading.

6.3. Implementation plan

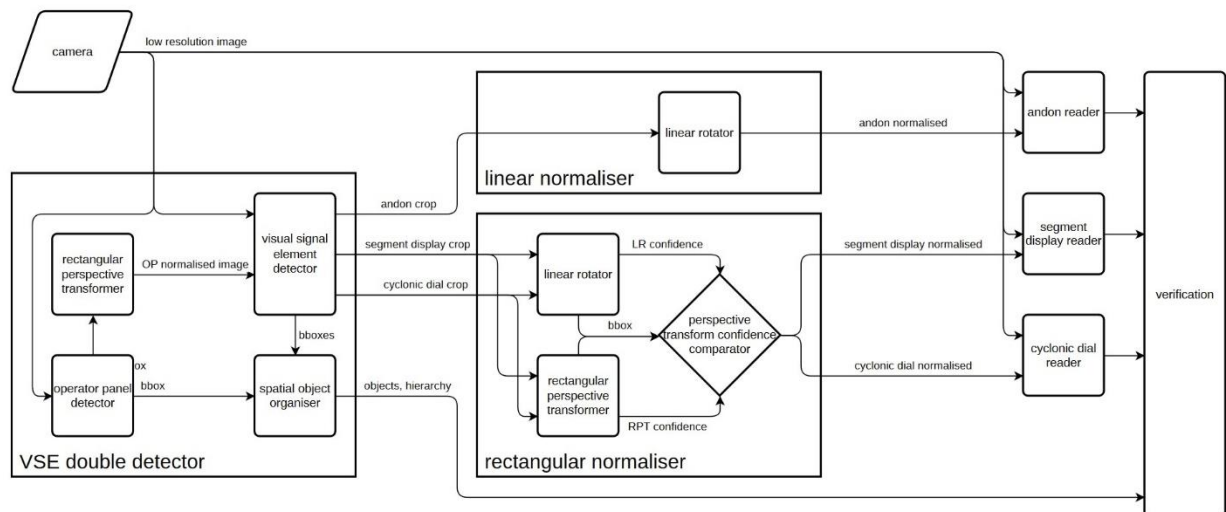


Figure 12 - A diagram of an implementation plan of the monitoring system

Based on the second iteration from chapter 6.2. Conceptual design and some testing, third (an in the context of this thesis the final) flow chart was created. It focuses only on the parts, which are a part of the final thesis. Each block represents a class implemented in python. Also, for easier understanding, the process was divided into several groups.

Compared to the previous version, there are a few changes. They are listed below:

- Double detection - there are two detectors: the first detects operator panels. Each operator panel is perspective corrected. The second detector detects visual signal elements, and accepts the original image from the camera and also the perspective-corrected images of operator panels.

- Rectangular normaliser is responsible for normalising rectangular features. As described in 6.2.3. Rectangular perspective transformer, a simple Rectangular perspective transformer is not reliable for thin rectangles. Therefore, a robust linear rotation and perspective transform is tried. By comparing the confidences, Perspective transform confidence comparator chooses the more reliable better transform.

6.4. Conclusion

Through an iterative process, a final implementation plan was created. The analysis was based on previous research.

The output of this chapter are two useful diagrams:

- A diagram of conceptual design in chapter 7.2. Conceptual design, showing the whole system
- An implementation diagram in chapter 7.3. Implementation plan, showing concrete building blocks of the software developed in this thesis. The concrete building blocks are:
 - camera
 - rectangular perspective transformer operator panel detector VSE detector spatial object organiser linear rotator
 - perspective transform confidence comparator andon reader segment display reader cyclonic dial reader verification

7. Dataset preparation

After the research in chapter High-level image processing - segmentation, detection, there was one concrete conclusion - the use of YOLOv8 as an object detector. YOLOv8 comes pretrained on the COCO dataset. The majority of classes in COCO dataset are man-made objects, therefore it is expected to perform good on industrial VSEs. The pretrained model is used as the first estimator and must be fine-tuned to detect VSEs. For this purpose, the **Industrial visual signal elements detection** dataset was created. Below are the steps of the dataset development. [24]

7.1. Image gathering

During research, no dataset covering the whole problem was discovered. Only some small datasets focusing on a particular problem were found. Therefore, it was necessary to create a new dataset. Primary images for this dataset were gathered from 3 sources.

1. Images captured by the creator of this thesis. These are photos and videos of industrial scenes in laboratories at CTU and photos of other electrical instruments with VSEs.
2. Data from other datasets.
 - 2.1. Some data are a part of the [25] dataset. This is an open-source dataset focused on personal protection equipment in an industrial workplace. Therefore, only images were used and annotations were added.
 - 2.2. A dataset focused on a single seven segment display [23] is also partly used.
3. Images downloaded from the web. Those images were gathered across the internet using google search engine. They were of course without annotations, which needed to be added.

During the image gathering process, following images were gathered:

- 432 images of a laboratory workbench at CTU with changing state and composition
- 35 images of a PLC at a laboratory at CTU with changing state and composition
- 30 photos of various machines with VSEs
- 240 images of unique objects from the internet
- 123 images from the PP02 dataset
- 60 images from the Seven Segment Display on Industry dataset

giving a total of 920 unique images of 314 unique objects.

7.2. Image annotation

Every image was annotated by hand by the creator of this thesis using [this](#) tool. Annotations have a form of bounding box around each object belonging to one of 7 classes. The classes with some information are listed below.

number	name	total count	present in images
0	andon	510	457
1	light	2702	595
2	cyclonic_dial	132	76
3	segment_display	437	127
4	general_display	426	262
5	clock_dial	621	224
6	operator_panel	483	345

Table 2 - Number of occurrences of each class in the base dataset



Figure 13 - A photo from a CTU lab



Figure 14 - A photo from a CTU lab with annotated classes

7.3. Data augmentation

For a robust ML model, a varied dataset is needed. The more variations of the combination of the same object and its background, the more robust will the trained model be. The major groups of variation in a dataset of images is listed below:

Objects of the same class do not generally share all features. A high variety of **specific object instances** leads to detection based on features specific for the object, not a common feature that the objects randomly happen to share.

The objects appear to look differently under different **geometrical conditions**. These can be position on the image, distance from the camera, the angle to the camera. Other conditions like warping and flipping can occur.

Variations in **colour space** also occur when dealing with real-world data. In the HSL colour space, these can be divided into hue, saturation and lightness variations. The lightness corresponds to **brightness**, which can change dramatically based on camera settings and light conditions. Also, it changes with contrast and other transformations. The saturation may differ based on the camera and lighting conditions. **Hue** variation is a special case. Some objects and backgrounds do not have a specific colour (an example would be a screen). Some objects have a fixed colour, but the hue of lighting can change it.

Given the object does not have a fixed **background**, which is generally not the case, there are huge variations in the background.

Filters are unavoidable part of real-world data. Some important cases are noise and blur caused by the optics, dust or relative movement of the scene and camera.

7.3.1. Tools for data augmentation

To enlarge the dataset in the direction of a certain variation, the data are augmented. Below is a list of tools used for this.

1. Geometric transformation - to mimic a real-world scene, the images are randomly flipped, cropped, rotated, stretched and zoomed
2. Colour space transformation - randomly changing pixel values
3. Kernel filters - random sharpening and blurring
4. Erasing - randomly erasing parts of the initial image
5. Mixing - attaching multiple images together [26]

The main tool used in this thesis is the **Albumentations** open-source python library. It enables efficient image augmentations. It implements a wide variety of transform operations. [27]

7.4. Dataset optimisation through selective augmentation

From the table above is clear, that this is a very unbalanced dataset. Also, more unique images lead to a more reliable model. Based on research in chapter 7.3. Data augmentation, data augmentation is used to enrich and balance the dataset. To determine the number of augmentations for each image, it was decided to use an optimisation method.

7.4.1. Terminology

The dataset consists of images from two main sources: single images and videos. Every single image has a different name suggesting that unique objects are in it. Therefore, these images are called **unique images**. Images from videos, however, have same name with different appendix suggesting that they hold the same object. Therefore, only one image from this group is considered as a unique image, the other images are called **image variants**. When augmentation is applied, image variants of a unique image are created. A single unique image is also an image variant of itself.

7.4.2. Problem formulation

The general formula is:

$$C \cdot R = N$$

where:

- C is a $c \times o$ matrix which assigns number of instances of each class per each unique image c is the number of classes o is number of unique images
- R is an $o \times 1$ matrix holding the number of variants of each unique image
- N is a $c \times 1$ matrix holding the number of class instances in all image variants per each class

For the basic dataset described in 7.2. Image annotation, the formula is rewritten as:

$$C \cdot P = N_{\text{start}}$$

where:

- C is a 7×314 matrix
- P is a 314×1 matrix holding number of variants of each unique image in basic dataset
- N_{start} is a 7×1 matrix holding number of class instances for each class in basic dataset

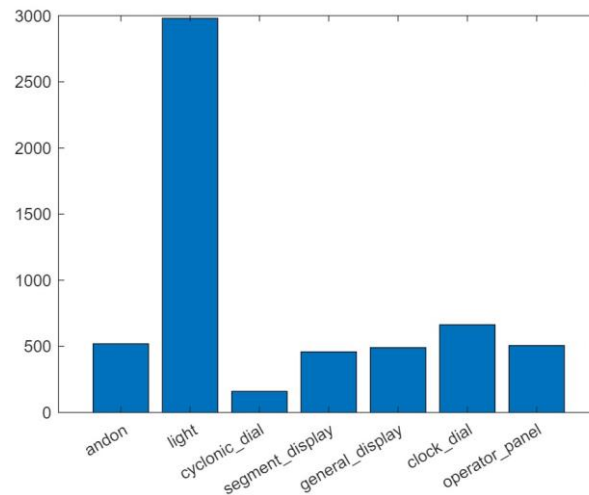


Figure 15 - A bar graph of occurrences of each class in the base dataset

For an optimised dataset:

$$C \cdot (P + X) = N_{aug}$$

where:

- X is a 314×1 matrix holding number of image variants to be added/removed. This matrix is the one being optimised by the optimisation method.
- N_{aug} is a 7×1 matrix holding the number of class instances for each class in the optimised dataset.

The goal is to create a rich dataset, so the number of instances of each class should be at least a certain number.

$$C \cdot (P + X) \geq N_{min}$$

where:

- N_{min} is a 7×1 matrix holding the minimum number of class instances for each class in the optimised dataset.

The form of the formula passed to the optimisation method is:

$$A \cdot x \leq b$$

The formula was adjusted to be consistent with the form described above:

$$-C \cdot X \leq C \cdot P - N_{min}$$

$$A = -C \cdot X \quad b = C \cdot P - N_{min}$$

7.4.3. Loss function

For an optimisation method to be used, it is crucial to formulate the requirements and goals of the optimisation expressed as a loss function J . Based on the research in chapter 3. Physical constraints and influences and 7.3. Data augmentation, following requirements were formulated:

1. **Total number of images**: the lower the total number of images, the better. Lower number of images results in less memory needed and less computing power required when handling the dataset.
2. **Deviation from N_{\min}** : The deviation from the minimum number of class instances should be penalised.
3. **Unique object variety**: the bigger the variety of unique objects presented to YOLO during training, the more general the predictions will be. The premise is, that every image with a unique name contains unique objects.

The first loss function J_1 is defined as follows:

$$J_1 = R^T \cdot K \cdot R$$

where:

- K is a weight matrix, in this case just diagonal 314×314 matrix with ones on the diagonal.

This loss function grows with growing number of images. It also grows with standard deviation of numbers in R .

Therefore, it satisfies the requirement 1. and 3. If N_{\min} is a constant matrix, it even satisfies requirement 2.

In the case of N_{\min} not being a constant matrix, another loss function J_2 can be used. J_2 is defined as follows:

$$J_2 = e^2 = (C \cdot R - N)^T \cdot L \cdot (C \cdot R - N)$$

where:

- L is a weight matrix, in this case just diagonal 7×7 matrix with ones on the diagonal.

In practice, N_{\min} will be a constant matrix and the effect of J_2 is negligible. Therefore, the J_1 loss function was reformulated to be consistent with the form for quadratic programming, which is the preferred method to nonlinear programming due to accuracy and speed:

$$J_{\text{quad}} = x^T \cdot H \cdot x + c^T \cdot x$$

$$J = X^T \cdot K \cdot X + 2 \cdot P^T \cdot K \cdot X$$

$$x = X$$

$$H = K \quad c = 2 \cdot P \cdot K^T$$

7.4.4. Constraints

Quadratic programming enables the use of constraints of x . As there is no way to realise a negative amount of image variants, following constraint is defined:

$$R = P + X \geq 0$$

$$X \geq -P$$

Therefore, the lower bound of X is $-P$.

Also, an upper bound on R was defined to make sure there are no extremely overrepresented unique objects. The upper bound was set to 5 % of the minimum number of instances of each image. In this case, it is 250 for every image.

$$R = P + X \leq I \cdot 250$$

$$X \leq I \cdot 250 - P$$

7.4.5. Optimisation

For optimisation, MATLAB's "quadprog" function was used. The arguments passed are all discussed above in detail. It successfully found a minimum. Plot of N_{aug} is below:

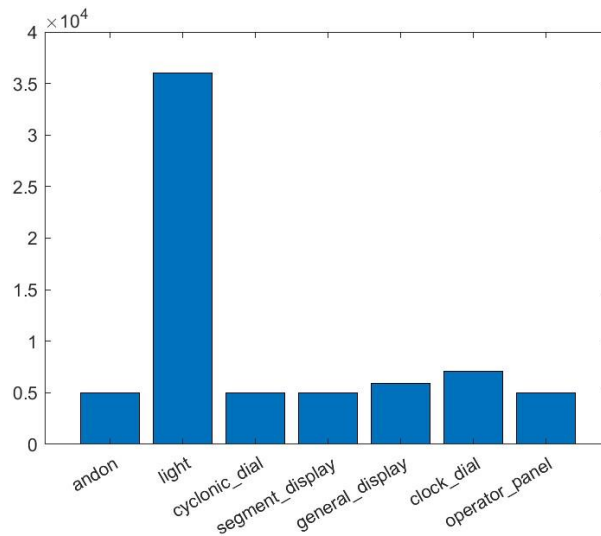


Figure 16 - A bar graph of occurrences of each class in the augmented dataset

At first glance, the main problem is class 1 - light. That is, however, a problem caused by the nature of the dataset, not solvable by optimisation. The cause of the problem are mostly andon lights and operator panels. A normal andon light contains 3 signal lights and nearly every operator panel contains at least a few signal lights. Therefore, even a perfectly balanced dataset would still have 3 times as much lights as andons. However, this high amount of signal lights should not be a major problem.

Actually, during testing of the YOLO detector, performance on signal lights shows to be very poor. So in the end, the YOLO detector is not even applied to detect signal lights, so this overrepresentation is not a problem at all as the signal lights can be ignored. This will be discussed later in great detail.

A bar graph with upper limit set to 10 000:

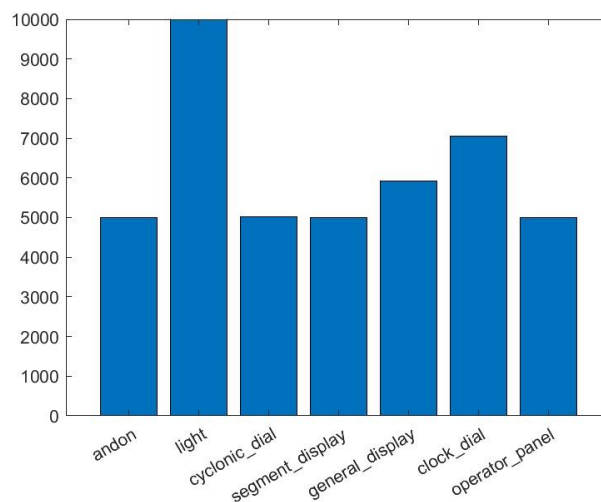


Figure 17 - A bar graph of occurrences of each class in the augmented dataset with y-axis limited to 10 000

When ignoring the light class, it is obvious that the optimisation was successful. General displays are overrepresented by 17 %, clock dials by 37 %, other classes lie within ± 0.4 %.

The total number of images is 3978 with an average of 8.2 instances of a class other than light in each image. There is an average of 9.0 instances of signal lights in each image. Below is an image variants counts histogram:

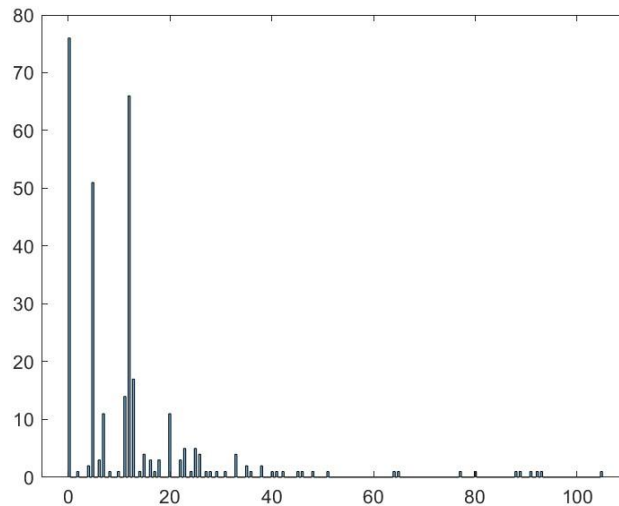


Figure 18 - A histogram of variants counts

There are two positive aspects to what this graph is showing. First, most of the counts are near the average value of 12.9, which is not extremely low nor high. Second, the maximum is 105, which is far below the given constraint of 250.

There is also one negative aspect, being the 76 images with zero variants, meaning they are discarded. That is 24 % of unique objects being discarded, which is a high number. Further development is needed on the front of this behaviour. Losing a quarter of unique images means losing a quarter of unique objects. One solution might be raising the lower limit of X, another is to higher the N_{min} so the non-optimal images will have lower impact, or to add images to dataset that counteract the non-using of those 24 % of images.

7.4.6. Augmentation

Based on the research in chapter 7.3.1. Tools for data augmentation, the python library Albumentations will be used. First, it is necessary do define a pipeline with defined transformations and parameters. Than, images and number of augmentations is passed to the pipeline, which randomly applies the defined transforms. Based on the research in chapter 3. Physical constraints and influences, a pipeline was set up. In the appendix, there is a snippet of the code defining the pipeline.

After the pipeline is set up, the algorithm goes through every unique image or group and either adds variants through augmentation or deletes random variants if the optimisation said so. The resulting dataset contains original images, but also variants like the ones below:



Figure 19 - An augmented image variant - noise, low brightness and horizontal flip [33]

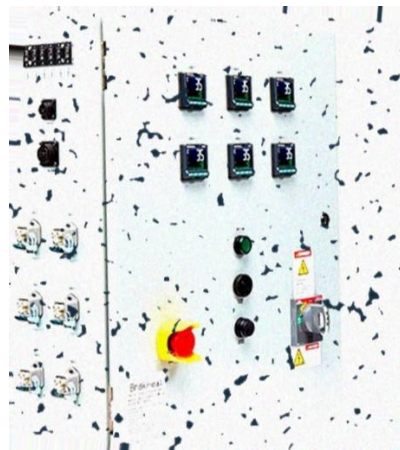


Figure 20 - An augmented image variant - high brightness, artifacts imitating fluid droplets on the camera [33]

7.5. Conclusion

In this chapter, the process of creating a dataset and its optimisation was described. First, images were gathered and labelled. Then, after defining a problem of optimisation, quadratic programming was used to calculate the optimal composition of the dataset. Then, an augmentation pipeline was defined and the dataset was optimised.

The output of work described in this chapter is a professional dataset named **Industrial visual signal elements detection**.

8. Detector training and performance

Based on the research in chapter 6.1.1. Detection, Localisation, low-level classification, the YOLOv8m is used.

8.1. Training platform

Neural network training is a HW demanding process. First, the most powerful HW at the hand of creator of this thesis was used. It was a Asus laptop ZenBook with Intel i7 8th gen CPU with integrated GPU. It was, however, not usable, as it was able to go through a maximum of 10 epochs per day if the laptop was not otherwise at use. Because there was no simple way to utilize any HW at school at the time of training, the choice was to try cloud-based services. Free services were preferred, the first being Google Colab. It offers a free GPU, but for a limited time only. Also, it is sensitive to inactivity, so practically it was hard to get more than 15 epochs at a time. Than, the final choice was Kaggle's notebook. It also offers a free GPU with the time limit comparable to Google Collab. Kaggle's notebook, however, is not sensitive to inactivity, so it was not a problem to run tens of epochs for 10 hours straight. The speed of each option with certain HW selected is listed in the table below:

device	speed
ASUS ZenBook i7 CPU	173 s/it
Collab no accelerator	50 s/it
Collab GPU	0.8 s/it
Collab TPU	36 s/it
Kaggle P100	0.8 s/it
Kaggle Tx2	1 s/it

Table 3 - Comparison of performance of certain HW when training a neural network

A huge difference between CPU (no accelerator) and an accelerated workflow can be seen. Even with the time limits, it was way faster to train the YOLO network on cloud based services than on weak hardware at hand.

8.2. Training

During training, YOLOv8 uses batches of images. In this case, the batch size was set to 8. The batch than consists of 8 images, each consisting of several other images. An example is below:

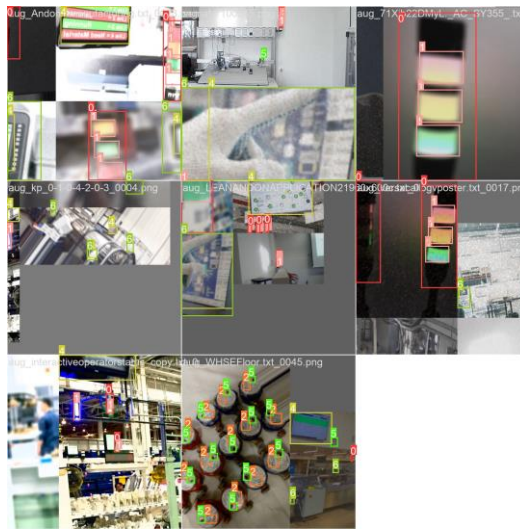


Figure 21 - An example of training data batch

The final detector was trained 9 epochs on the Asus laptop on the non augmented dataset and than 49 epochs on Google Collab and 100 epochs on Kaggle's notebook. The progress of metrics and loss functions can be seen on the figures below

On the figure below, the progress of important metrics is shown.

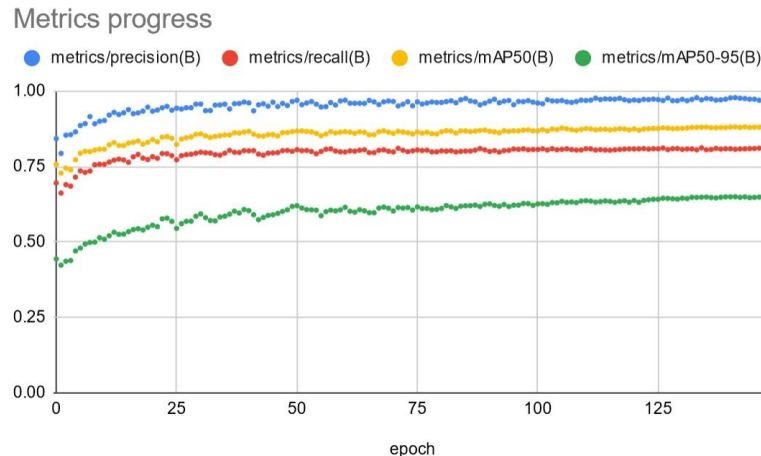


Figure 22 - Scatter plot of progress of 4 important metrics during training

- **precision** is a ratio of correctly predicted positive observations to total predicted positives.
- **recall** is a ratio of correctly predicted positive observations to all observations in actual class.
- **mAP50** (mean Average Precision at IoU=0.50) is average precision of the model when Intersection over Union (IoU) threshold is 0.50.
- **mAP50-95** (mean Average Precision at IoU=0.50:0.95) average precision of the model over a range of IoU thresholds from 0.50 to 0.95. [10]

In the figure below, the progress of loss functions is shown.

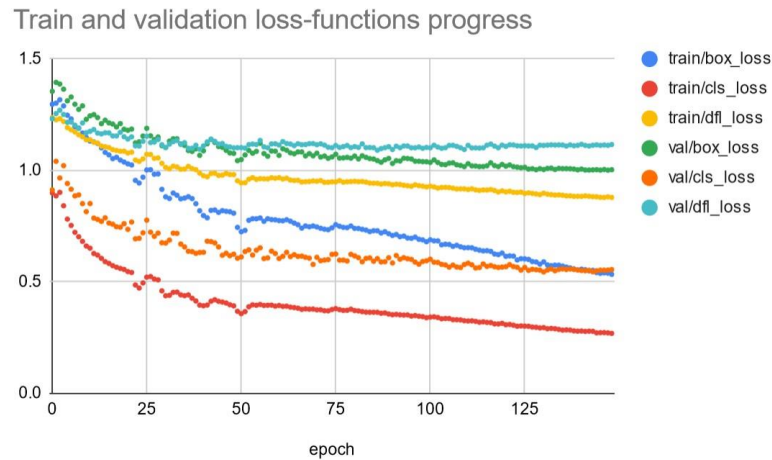


Figure 23 - Scatter plot of progress of loss functions during training

- **Box Loss** is the difference between predicted bounding box coordinates and the actual coordinates in the ground truth data. The lower the loss, the more accurate the model is at locating objects within the image.
- **Class Loss** is the difference between the predicted class probabilities and the actual class labels in the dataset. The lower the loss, the more precise the model is in identifying the classes of the objects.
- **DFL Loss** (Distribution Focal Loss) is a specialized loss function designed to handle extreme sample imbalance issues in object detection. It operates by adjusting the standard loss computation with a distribution-based weighting scheme. This scheme is intended to pay more attention to harder-to-classify examples and less attention to easier ones. Essentially, DFL Loss helps to ensure that the model learns effectively from rare or uncommon classes without being overly influenced by more common classes. This strategy aims to improve the model's overall performance on a diverse range of classes.

It is apparent from the figures, that the validation loss functions are constant or changing slowly at the end. Also all metrics are constant from epoch 100. Only mAP50-95(B) is still growing. The validation loss functions are also at or near their minimum. This indicates, that the process of learning is nearly ended and only small improvements can be expected with further training.

8.3. Performance

After training, the YOLO runs a validation test on all validation data. This generates the confusion matrix, an important metric of the detector. It shows class-dependent true and false positives and negatives.

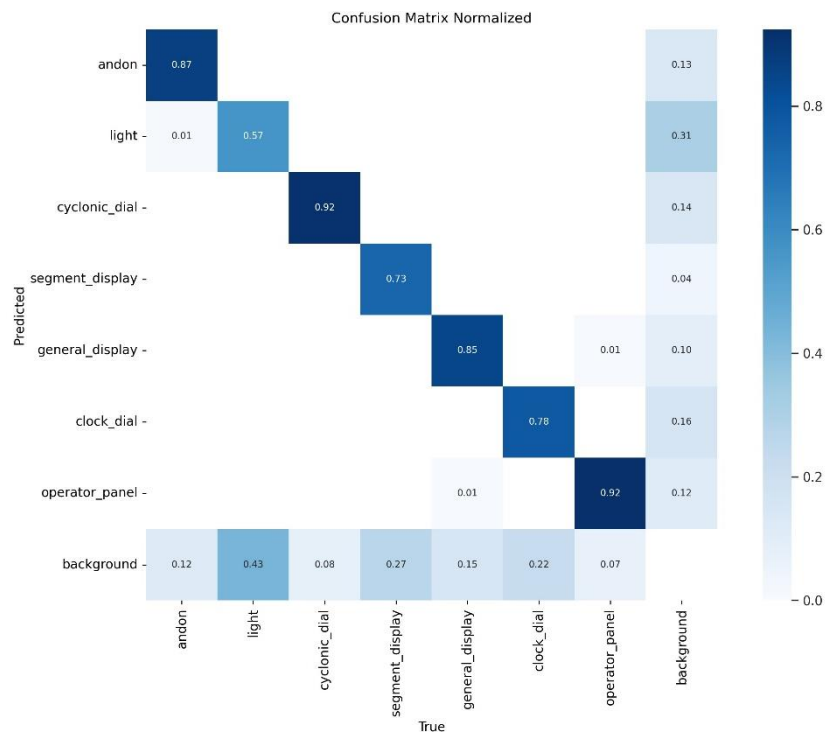


Figure 24 - confusion matrix of the trained YOLO network

The ideal form of the confusion matrix is a diagonal matrix with ones on the diagonal. In reality, a good detector is going to have numbers near 1 on the diagonal. The last column represents false-positive detections, the last row represents false-negative detections. The rest of the matrix represents true detection of an object with falsely assigned class (mislabelled).

In this case:

- the diagonal is >0.85 for andons, cyclonic dials, general displays and operator panels. This is considered as a satisfying result.
- For segment displays and clock dials, the diagonal is lower. Therefore, the results should be analysed and more effort should be put into better detection of those classes.
- The detector shows very poor performance on simple signal lights. This can be caused by a high variety of colours, shapes and types of these lights. Also, it might be due to their small size and lack of distinct features. Therefore, YOLO is not used as a primary simple signal light detector.
- There is no significant mislabelling.

Below, there is an image as an example of a validation image with true labels and the same image labelled by the YOLO detector. Also, the confidence of each label is shown.



Figure 25 - An example of a set of validation images with true labels



Figure 26 - An example of a set of validation images with detected labels

8.4. Conclusion

In this chapter, the platforms for training NN were compared and training process of a NN was roughly described.

- Practically, the YOLOv8m model was successfully learned for its purpose with issues being:
- improvable precision of segment display and clock dial detection
- unsatisfactory precision of signal light detection

As graphs indicate, the training process does not show much place for overall improvement, so the model training is considered as finished, because more training would not result in a significant improvement to the model performance. However, improvement could be achieved by focusing on the two underperforming classes: segment displays and clock dials.

Given the very low precision of signal light detection, the detector can not be reliably used to detect lights. It can be used as a helper, but is not used as a reliable primary simple signal light detector.

9. Implementation

The implementation was done along with the planning phases. After the second iteration described in chapter 6.2. *Conceptual design*, the necessity of a normaliser emerged. Rectangular perspective transformer class was developed, but due to its low reliability in certain situations, it was necessary to also add a simple and robust rotator. The rest of the programming work was done during the third phase described in chapter 6.3. *Implementation plan*, where readers and helper classes were developed. All the steps are discussed in great detail in the next chapters.

9.1. Objects and instances

The word "object" used in this thesis meant a physical object. Every physical VSE is an object. Also, a VSE in an image is called an object. Also, we distinguish 7 different classes of VSEs.

However, in this chapter, the code will be discussed. In OOP (object oriented programming), there are 3 main terms that are widely used. Below, they are described from the OOP point of view:

- **object** - an entity, defined by a blueprint, that contains real data and can be interacted with.
- **class** - a blueprint of an object. Defines the contained data and behaviour of the object.
- **instance** - or a class instance, is an equivalent term to object

For clarity, from now on:

- When talking about OOP, the words **class** and **instance** are used.
- When talking about an image or physical world, the words **object** and **label** (instead of class) are used.

9.2. Packages and modules in general

9.2.1. Code organisation

The project lives in a base folder called `industrial_hmi_monitoring_project`. In this folder, there is the base python package `hmi_monitoring`. This package contains several sub-packages, as well as some general modules. One of them is `utilities.py`, containing functions usable across the whole SW. ^[27]

Every sub-package is focused on a specific task and contains one or more modules. Each package and sub-package contains a `settings.csv` file where settings for the particular package are stored.

9.2.2. The process of algorithm development

The output of chapter 6.3. Implementation plan was respected. One module at a time was developed. The development process can be divided into four steps:

1. Creating an easy-to-debug **working prototype** as soon as possible. In this step, the main focus was on usability on a set of examples and on the speed of creation. Therefore, Jupyter notebook was used for this purpose. It offers a fast workflow and effective in-line visualisation, which is key while developing a computer vision algorithm. The prototype has a form of a script containing some functions.
2. Converting the prototype into a debug class, a production class and functions in a **python module** following the principles of OOP. After the prototype is performing good on testing data, the script is converted to a series of functions. Each function represents a recurring scheme or a larger task. Functions are then divided into three groups:
 - 2.1. **general utilities** - universal functions usable in any part of the whole SW. These functions are placed in the utilities.py module in the base package hmi_monitoring.
 - 2.2. **static functions** - functions usable only in the context of the particular class, which do not act as a method of the class. In other words, it is a function which does not read or change the attributes of the particular class. These functions are placed in the module above the class definitions.
 - 2.3. **class methods** - functions, which act as methods of the particular class. These functions are placed in the module inside the class definition.
3. The whole algorithm is then converted to be callable as a method of a class. As the prototype algorithm outputs a lot of intermediate results, which are important during debugging, but would be slowing down any processes using it. The prototype algorithm is first converted to a debug class. Then, a child of this class is created - the production class. The production class overrides all methods, that display or print at every call with methods without any external output. Printing information during the instance creation is preserved.
4. **Testing** - the usability of the created class is tested. The goal is to test all methods and whether the class works like the prototype algorithm in a general sense.
5. **Compatibility and fine-tuning** - the class is then used by other processes. Any problems with compatibility are solved as the other process is developed. Also, if the class is tested on new data, the parameters are fine-tuned if needed.

9.2.3. The static parameters of algorithms

Real world tasks solving algorithms developed empirically, which is also the case of this thesis, require parameters. Most parameters can be set as a function of other known parameters, some parameters can be found using another sub-algorithm, but there are often some parameters,

which have been set by the programmer. This is often a case of various thresholds and coefficients. These parameters are set either based on a real experiment, a thought experiment, by a "professional guess" or even randomly. Minimizing the number of such parameters is a priority, but it is not always possible or efficient.

In the case of this thesis, all parameters were either set to be a function of another known parameter, if there was a clear relationship, or the value was determined based on an experiment. Sometimes, a group of parameters was set as a function of a custom-made static super-parameter.

During the **development of an easy-to-debug working prototype**, micro-experiments were conducted. As this thesis is focused on developing a framework that will be adjusted to the particular real-world application, rather than developing and fine-tuning a perfectly performing algorithm, the micro-experiments were conducted on a set of several particular images from the `industrial_hmi_detection_dataset`. The micro experiments were not documented. However, if some information appeared to be helpful in the future, they were added to the code as a comment.

During the **converting the prototype into a python module**, the static parameters are recognised and organised to a `settings.csv` file. In the code, they are replaced with a variable loaded from this file. The static parameters are read from the csv file once the particular instance is created. This way, it is easy to adjust the static parameters, even automatically. The `settings.csv` file contains default and user values. User values are always prioritised and the default values should not be changed, unless there is a good reason to do so.

9.3. Spatial object organiser (SOO)

This is a module containing multiple classes for easier handling and organisation of detected VSE.

9.3.1. Helper classes

The **BoundingBox** helper class was developed due to the frequent need to work with bounding boxes. The bounding box is stored like an array of 4 numbers, defining the lower and upper limit on the x and y axis. For cropping, this is sufficient. However, when more complicated operations are applied more frequently, it is helpful to define the bounding box as an instance of the `BoundingBox` class.

The definition is shown in the snippet below:

```
# importing the necessary class
from hmi_monitoring.spatial_object_organiser import BoundingBox

# creating a bounding box instance
BB = BoundingBox((10,10,110,110))

# examples of usage
shape = BB.shape
height = BB.height
```

The **VisualSignalElement** helper class was developed with OOP principles in mind. Each detected VSE is a unique instance of the VisualSignalElement class. It holds information about the origin of the VSE, its bounding box, state and the normalised crop. These instances are used in the main class SOO.

9.3.2. Public methods

The **init** method initialises parameters and reads settings for the instance.

The **add_object** method accepts parameters describing the VSE to be added. The parameters are the bounding box, label, origin of the detection and optionally normalised crop.

First, the VSE being added is tested against all other objects with the same label already listed in the instance. If the intersection with all of them is lower than a given threshold, an ID is assigned to it and it is stored as a new object. Otherwise, the intersecting object is only updated with the input parameters.

Based on the label, the method decides, whether the object being added is a group object (andon, general display, operator panel) or an elemental object. If the object being added is a group, a refresh of all memberships is executed.

Otherwise, only the added object's membership is analysed.

To get a list of information about all objects in the memory of the SOO, the method **get_list** is called.

9.3.3. Usage

Below, there is a snippet showing the creation of an SOO instance.

```
# importing the necessary class
from hmi_monitoring.spatial_object_organiser import SpatialObjectOrganiser

# defining the SOO instance
SOO = SpatialObjectOrganiser()
```

After adding some objects, it makes sense to get a list of all objects. Below is a snippet to do so.

```
obj_list = SOO.get_list()

obj_infos = [obj[:4] for obj in obj_list]
obj_crops = [obj[4] for obj in obj_list]
```

Using list comprehension, the list of objects was split to a list of information about the objects and a list of normalised crops for the purpose of better visualisation. If the `obj_infos` was printed, the results can look like the ones below:

```
[[[ 41, 195, 318, 320], <YoloDetectorClassifier object at 0x0000021BFC3E3070>, 4, [1]]
 [[ 8, 152, 335, 547], <YoloDetectorClassifier object at 0x0000021BFC3E3070>, 6, []]
 [[135, 193, 311, 315], <YoloDetectorClassifier object at 0x0000021BFC3E3070>, 3, [0, 1]]
 [[ 41, 191, 312, 318], <YoloDetectorClassifier object at 0x0000021BFC3E3070>, 3, [0, 1]]]
```

In the first column, there are the bounding boxes defined by lower and upper limits on x and y axis. second column is the instance, which detected the object. The third column is the label and the four column contains a list of IDs of all group objects the given object belongs to. In this case, there is an operator panel containing a general display and two segment displays. Both the segment displays also belong to the general display. All the objects were detected by the same detector.

Below is the visualisation of the obj_crops list.



Figure 27 - 1. element of obj_crop list – a general display [32]



Figure 28 - 2. element of obj_crop list – an operator panel [32]



Figure 29 - 3. element of obj_crop list – a segment display [32]



Figure 30 - 4. element of obj_crop list - a segment display [32]

From this list, it is clear that there are 4 objects stored. The first object was correctly labelled as a general display (because segment display is defined as a display with 1 line of characters) being a sub-object of the second operator panel object. The second object is a correctly labelled operator panel, as it is a panel containing input and output HMI elements. The third and fourth object are both two-row segment displays, but each should be recognised as two distinct segment displays. They both are correctly recognised to be sub-objects of both the general display and the operator panel.

9.3.4. Limitations

It is clear there are no duplicities when considering the limited rules of the SOO. Obviously, there is some sort of duplicity between object 0 and 3, which are spatially equal, but were labelled with two different labels, and object 2 and 3, where 2 is just a cropped version of 3. However, these are high level similarities, which should be evaluated in the verification step.

The SOO, as it is designed at this point, stands on the side of security. In practice, it only discards objects with the same label occupying the same area on the image. At the same time, it stores all possible variants of the same object created by other blocks, especially the detector, leaving space for the correct representation.

9.4. YOLO as a detector and a low-level classifier

The first implementation was the detector. Although it's a simple class, it is necessary for all subsequent algorithms. The class has 3 methods.

9.4.1. Methods

The **init** method initialises the YOLO object detector object and also the YOLO detector itself. It takes the model path as an argument, so it's easy to always choose the best model. Also, it accepts a spatial object organiser object as an argument.

The **load_image** method accepts image data and converts them to a form necessary for the YOLO detector. This means converting it to a numpy array of a size 640x640x3 pixels. It also converts the image from BGR to RGB, as the main technique used to load an image is the function `imload` from OpenCV which uses BGR as a default.

The **detect_single** method takes the list of labels to detect as the main parameter. Optionally, it is possible to set confidence threshold for detections, device used for computing the detection and an option to show the results.

9.4.2. Usage

To use the YOLO detector and low-level classifier, the snippet below can be used to initialise the YoloDetectorClassifier class:

```
# importing the desired function from the python package  
from hmi_monitoring.detector.yolo_detector_classifier import YoloDetectorClassifier  
  
model_path = 'path/to/model/best.pt' # example of the path to the wights  
spatial_obj_organiser = SOO # SOO is an instance of the class SpatialObjectOrganiser  
  
# creating an instance of YoloDetectorClassifier class  
YDC = YoloDetectorClassifier(model_path = model_path,  
spatial_obj_organiser = spatial_obj_organiser)
```

To perform a detection of all 7 labels, the snippet below can be used

```
# imports  
import cv2  
  
# reading an image from disk  
image_raw_bgr = cv2.imread('path/to/image.png')  
  
# loading the image into the YDC object  
image_resized_rgb = YDC.load_image(image_raw)  
# performing the detection task  
_,results = YDC.detect_single(classes=(0,1,2,3,4,5,6), # detecting all labels  
confidence_threshold=0.25, # the minimum confidence  
device='cpu', # device used  
show=False,) # do not show debug images nor print info
```

The resized image, passed to the YOLO detector, is below:



Figure 31 - resized image passed to the YOLO detector [32]

After the detection is performed, the debug image is generated by the YOLO detector. In this case, the detector found 3 objects, one operator panel, one general display and one segment display. Also, the confidence of each detection is shown next to the label.



Figure 32 - debug image generated by the YOLO detector with detected objects [32]

The result returned by the detector is a list of VSE object and detection confidence for each detected object. In this case, the result looks like this:

```

[[<VisualSignalElement object at 0x0000021B97719C90>, 0.9399730563163757]
 [<VisualSignalElement object at 0x0000021B977183A0>, 0.8981794714927673]
 [<VisualSignalElement object at 0x0000021B977184C0>, 0.8936042189598083]]

```

9.5. Camera

Camera is a simple class for easier handling of a single instance of cv2.VideoCapture class.

9.5.1. Methods

The **init** method reads settings from a settings.csv file located in the package. It sets all needed parameters and starts the prepares the connection to the camera output via creating an instance of the cv2.VideoCapture class.

After the camera is ready, the raw output can be obtained by calling the **capture_raw** method. For a pre-processed output of the camera, the method **capture** is called. The pre-processing consists of cropping, scaling and rotating the war output defined in the settings.csv file.

9.5.2. Usage

To create an instance of the Camera class and capture a raw and a pre-processed image, the code snippet below can be used:

```
# importing the class
from hmi_monitoring.camera.camera import Camera

# creating an instance
cam = Camera()

# getting the raw output from the camera
image_raw = cam.capture_raw()

# getting the pre-processed outputu from the camera
image = cam.capture()
```

9.6. Rectangular perspective transformer

As described in chapter 6.2.3.Rectangular perspective transformer, an instance of the class **RectangularPerspectiveTransformer** (RPT) serves as a tool for compensating for rotation and perspective warp. It has 2 main tasks: perspective warp detection and compensation.

9.6.1. Methods

The **init** method only initialises basic attributes and reads the settings from the settings.csv file located in the package.

The first 2 public methods, **load_image** and **load_image_from_path**, enable the user to save an image into the instance's memory to be further worked on.

The **transform** is the main method detecting perspective in the image crop and compensating for it. It accepts `class_detection_results` as an argument. For each detection, it crops the loaded image using the detected bounding box compensates for it's perspective transform.

Additionally, there are 4 getters for warped images, warp confidences, detection confidences and total confidences.

9.6.2. Usage

The code snippet below shows the usage of the transformer:

```

# the transformer module is imported
from perspective_transformers import rectangular_perspective_transformer as rpt

# creating the transformer instance
RPT = rpt.RectPerspTransformerDebug()
# saving an image into the instance's memory
RPT.load_image(image)
# performing the transform
RPT.transform(results)

# getting the first crop with compensated perspective
image_comp_persp = RPT.get_warped_images()[0]

```

9.6.3. Steps of the algorithm

For a one particular crop, the images describing individual steps are shown below:



Figure 33 - The input image of a segment display [34]

This is the input image. The panel is rotated and slightly warped by perspective.



Figure 34 - Successfully detected operator panel [34]

The input image is cropped according to the detection results. In this case, an operator panel was successfully detected.



Figure 35 - Detected operator panel with added context [34]

For a more robust detection, context is added to the RPT. In this case, the context is a few pixel boundary around the detected object. This technique often exposes more lines helpful for perspective detection, but does not mystify the detector by showing too much background possibly containing lines in other directions.



Figure 36 - Detected image with added context with lines detected by the *lu_vp_detect* algorithm [34]

Then, the *lu-vp-detect* algorithm tries to find the vanishing points. In this case, it successfully found the left and horizontal vanishing points:

vertical: [141,2792], confidence = 0.9743309636808276
R: [390,-298], distance = 444, rotation = 55°, perpendicularity = 145°
J_distance = 0.318, J_rotation = 0.619, J_perpendicularity = 0.619, confidence = 0.508
L: [-4405,-298], distance = 4561, rotation = 4°, perpendicularity = 94°
J_distance = 0.031, J_rotation = 0.051, J_perpendicularity = 0.051, confidence = 0.958

The RPT also evaluates several metrics, which are used to choose the more reliable vanishing point (left or right) and to return a confidence value for its solution. The metrics are distance and perpendicularity and are based on the presumption 2 from chapter 3.5.2. Rules and assumptions: "The VSEs must be well visible by the camera". The best visibility is reached when the object has the lowest perspective warp.

The value of perspective warp can be expressed as the distance of the vanishing point from the centre of the image. So the bigger the distance, the more confident the algorithm is in using the vanishing point.

The second way to express the value of perspective warp is by calculating the angle between the L/R vanishing point, the image centre and the vertical vanishing point. This angle should be ideally $\pm 90^\circ$. Anything between $\pm 180^\circ$ and 360° can't be trusted at all.

These metrics are expressed as loss functions J , which are then used to calculate the overall confidence. It is clear, that the algorithm correctly chose the left vanishing point.



Figure 37 - Detected operator panel with applied perspective warp compensation [34]

After the more trustworthy vanishing point is chosen, the warp itself is performed. The algorithm assures to maintain all data contained in the crop, therefore resizing the output image to be able to accommodate the whole warped image.



Figure 38 - A comparison of a warped image (right) with a rotated-only image (left) [34]

A comparison of a warped image (right) with a rotated-only image (left) is shown. For better comparison, magenta lines parallel to the image's axes were added.

9.6.4. Limitations

This algorithm, however, is not robust. It performs well in good conditions, but can perform poorly when the input data are mystifying or without strong edges. Below, there are two example of poorly warped images.



Figure 39 - Input image with hard-to-detect perspective [35]

The input image. There is one operator panel and one general display. It is clear, that the operator panel consists of two parts connected at an angle.

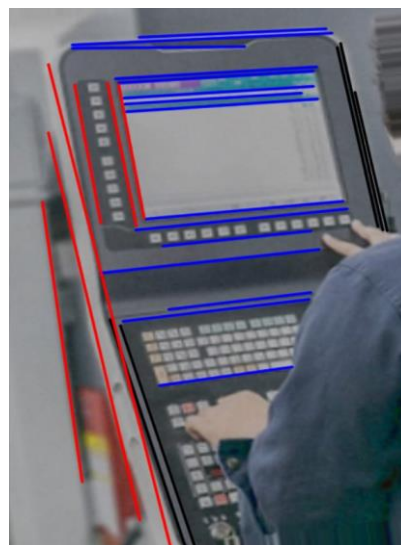


Figure 40 - Input image with lines detected by the `lu_vp_detection` algorithm [35]

The vanishing point detection debug image. The black colour of the lines indicates that they are strong lines, but are treated as outliers. On the other hand, the algorithm is very confident in detecting lines which are not really present in the image and even one line from the background.

The warped image than shows that the blue lines were helpful and are now truly horizontal. However, the algorithm didn't choose a particular part of the operator panel, but instead chose a compromise, which did not turn out very well.

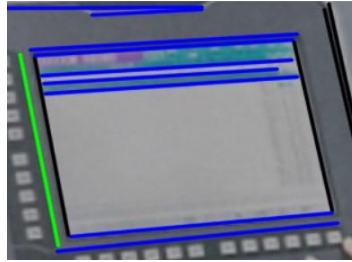


Figure 41 - detected general display with lines detected by lu_vp_algorithm [35]



Figure 42 - Input image with wrongly applied perspective warp compensation [35]

The general display, compared to the whole operator panel, appears to be much simpler to find perspective in. However, the algorithm discards most vertical lines, which in fact are correct.



Figure 43 - Detected general display with wrongly applied perspective warp compensation [35]

Again, the blue lines were detected correctly, but the vertical direction was compensated for incorrectly.

9.7. Linear rotator

Given the limitations of the RPT described in chapter 10.6.5. Limitations, the need of a more robust and universal solution emerged. A simpler transformation is a simple rotation. Instead of

4 parameters (the positions of vertical and L/R vanishing point), the rotation needs to find only 1 parameter. The linear rotator (LR) proved to be a robust algorithm usable in many cases.

9.7.1. Methods

The **init** method only initialises basic attributes and reads the settings from the settings.csv file located in the package.

The first 2 public methods, **load_image** and **load_image_from_path**, enable the user to save an image into the instance's memory to be further worked on.

The **load_bb** method is used for loading of a single bb into the instance's memory. Unlike rectangular perspective transformer, which accepts a list of bounding boxes as a parameter of the main method, a separate setter method was used in this case, accepting only one bb. The goal is the ability to compare those two approaches.

The main method **find_rotation** performs the orientation detection using the hough lines algorithm as described in chapter 5.2.1. Line detection. The method returns rotated input image, rotated crop defined by the bounding box, points of rotated bounding box, a bounding box expanded to fit the object on the rotated input image and the confidence of the orientation detection. The method takes sensitivity as the only input, which turned out to be the only parameter which is not a function of any other known parameter, but also is a function of the input image. A usable range was found, but the parameter was left to be defined by the user. In the final implementation, an iterative approach based on the returned confidence is recommended.

9.7.2. Usage

```
# importing the module
import hmi_monitoring.perspective_transformers.linear_rotator as lr

# creating an instance of the linear rotator class
LR = lr.LinearRotator()
# loading an image to the instance's memory
LR.load_image(image)
# loading a bounding box to the instance's memory
LR.load_bb(bb)
# performing the rotation with sensitivity 1
rotated_image, rotated_crop, rotated_rectangle, bounding_box, rotation_confidence =
LR.find_rotation(1)
```

9.7.3. Steps of the algorithm

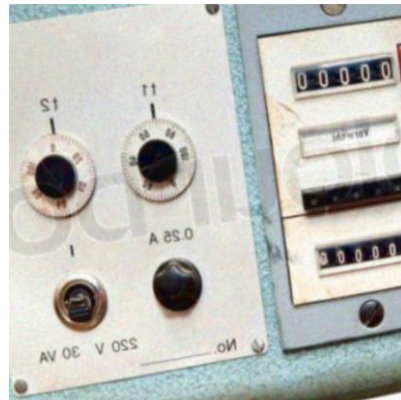


Figure 44 - An operator panel containing two cyclonic dials [36]

First, the algorithm is going in 3 paths using a classical grayscale image in the first, a lightness channel extracted from HSL in the second and saturation channel also extracted from the HSL in the third path. The saturation channel shows poor performance on unsaturated images (like the one being analysed here), but shows great performance when detecting highly saturated objects (like andons). The visualised steps of the algorithm are shown below:



Figure 45 - edges of one detected cyclonic dial on three channels

Edges are detected for each path. An iterative approach was chosen, starting with low thresholds of the Canny edge detector and gradually increasing them until the black to white pixel ratio drops below a defined value. This turned out to be a robust method.



Figure 46 - Lines detected on one detected cyclonic dial on three channels [36]

Then, for each path, lines are detected using hough transform and ordered by the number of votes. In the image, the first 10 lines are magenta, the rest are red. The number of votes is then used to weigh those lines when calculating the median of their angle.

Also, the sensitivity is used as a threshold for the line detection. Here, it was set correctly, as no lines were detected in the saturation channel image. The algorithm then lets each path vote for

the orientation it detected. It returns the median of those and the difference between the two top votes is used to calculate the confidence of orientation detection.



Figure 47 - Successfully rotated cyclonic dial [36]

A resulting returned rotated crop of the find_rotation method. In this case, the algorithm was successful.

For completeness, the input image also contains a whole operator panel. The LR algorithm can also be used in that case, the result is below:

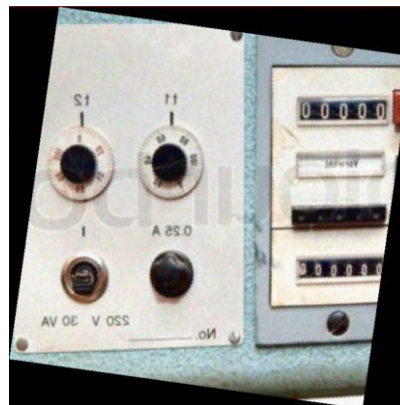


Figure 48 - Successfully rotated operator panel [36]

Although the input is slightly warped by perspective, the rotated image would still be useful in the case of a unsuccessful RPT application.

9.8. Andon reader

From all the 7 labels defined in the chapter 2.6. Conclusion, andons turned out to be the best combination of usability and ease of reading. Therefore, an andon reading algorithm is proposed in this thesis. The algorithm has a form of a jupyter notebook, as it enables fast development and easy visualisation.

9.8.1. Inputs and outputs

Input is an image, the detector assumes it is a photo of an upright andon light. It also assumes, that the andon light is horizontally at least near the centre. It also assumes, that the andon occupies about 90 % of the vertical space. These assumptions come from the system design

from chapter 6.3. Implementation plan. The detector and linear rotator, when keeping the rule from chapter 3.5.2. Rules and assumptions in mind, will return an image corresponding to the assumption stated in this paragraph.

The output is a list of segments. Each element of the list is another list containing information about a single segment, the information being the number of the segment, bounding box, colour, confidence and state prediction.

- The number is a positional number. The first segment is on top.
- The bounding box is a rectangle, that contains only pixels from the given segment with high confidence. It can't be regarded as a bounding box containing the whole segment.
- The colour is a median pixel value of the area bound by the bounding box.
- The confidence is the estimation of correctness of the information about the segment and even the probability of the segment's existence.
- The state is a list of two confidences, one for the on state, the other for the off state. This state estimation, however, works very poorly even for slightly overexposed images and for white segments.

The algorithm does the analysis on a set of evenly spaced vertical lines in the neighbourhood of the image centre.

Robust statistics is used in order to filter outliers and get a robust result.

9.8.2. Steps of the algorithm

First, the image is blurred. As a derivation is used, blurring is necessary. The blurring is done by a gaussian filter with high standard deviation in the direction perpendicular to the andon main axis and low standard deviation in the direction parallel to the andon main axis. Using this technique, any texture disappears from the segments, but the edges between the segments remain sharp.

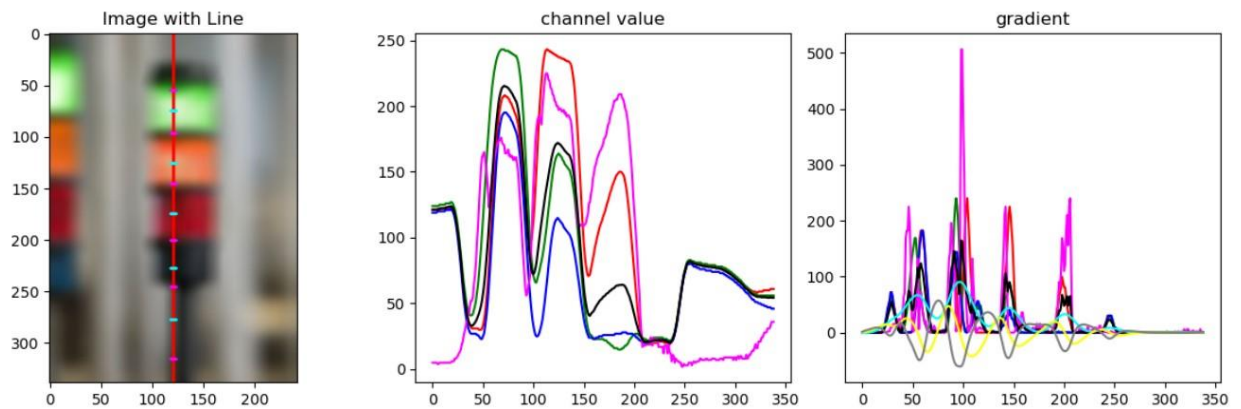


Figure 49 - graph of change of channel values and their derivations along a line through the image [37]

The change of red, green, blue, saturation(magenta) and lightness(black) channel is acquired. Then, the derivation is calculated and a median is calculated for each point across the line. Thanks to the approach of calculating a median of many different channels, this solution is robust in finding changes along the line. These changes correspond to a border between two segments.

Given the research in chapter 2.1. Signal lights, andon lights are supposed to have a maximum of 5 segments. Therefore, it is simple to calculate a maximal spatial frequency of a signal indicating the change of segment. So, a low pass filter is used on the derivation (cyan). In the filtered signal, every maximum is a potential border and every minimum is a potential centre of a segment. Therefore, second derivation (yellow) is performed and zero-crossings are found. Based on the third derivation (gray) being either positive or negative, the zero-crossing corresponds either to a border or a centre.

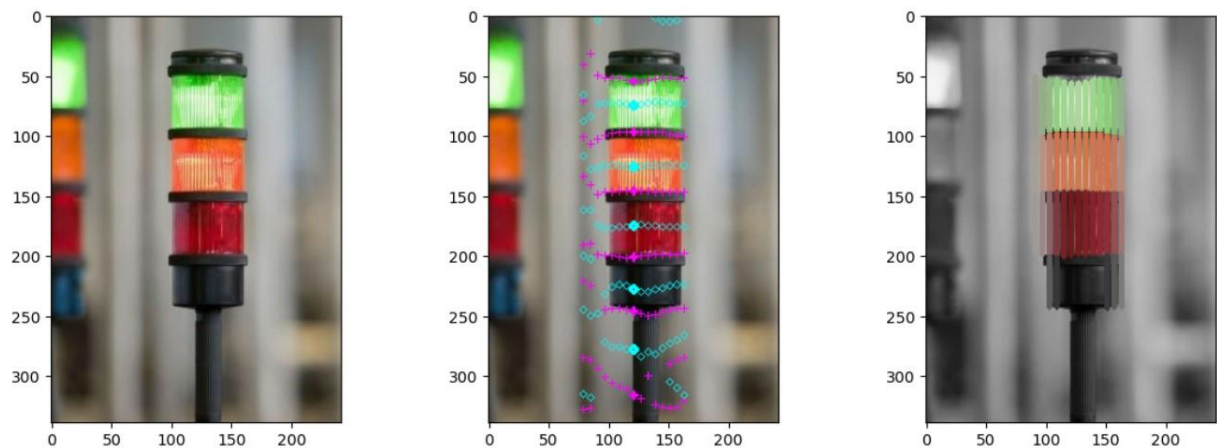


Figure 50 - Detected edges and centres of line segments, filtered line segments [37]

On the left, there is the original input image.

In the middle image, all boundary points are drawn as a magenta cross and all centre points are drawn as cyan diamonds. However, every centre with a missing boundary is soon discarded as the algorithm assumes that the whole andon is visible.

In the right image, filtered line segments are drawn with their average colour. The filtering is done on the base of lightness and saturation. If both of those metrics are low, the line segment is filtered. Also, extremely short and long lines are also discarded. From this step, the number of segments is estimated using the median of line segment count on every line.

The segment count is crucial for the next step. To find the centre and colour of each segment, a K-medians algorithm is used, which needs only the number of clusters as a parameter. It is fairly similar to the well-known K-means algorithm, which is however much more sensitive to outliers. As it is clear from the images shown, the bottom of the andon is also detected as a segment on some lines. Therefore, the presence of outliers is inevitable. The K-medians algorithm, however, performs well in this scenario. The algorithm is restricted to only look for colour and y coordinate of the centre, the x coordinate is calculated as a median of all line segments.

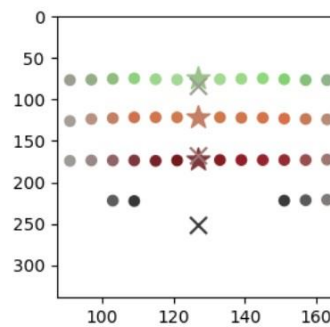


Figure 51 - A graph of data used for K-medians algorithm

On this graph, every point represents a line segment. The x, y coordinate corresponds to the x, y coordinate of its centre. The colour corresponds to its average colour. The x marks correspond to initial guesses for the K-medians algorithm, and the stars correspond to the cluster centroids detected.

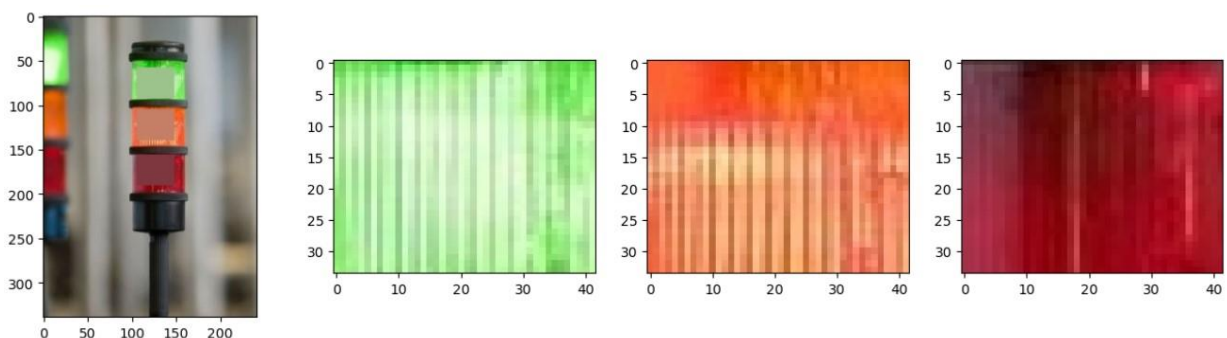


Figure 52 - Segments extracted from an andon [37]

After that, the centres of all segments are known. The height can be calculated based on the average spacing, the width can be estimated by the horizontal spread of the segment lines. The created rectangles are shrunk by a given amount to assure they do not contain any background pixels.

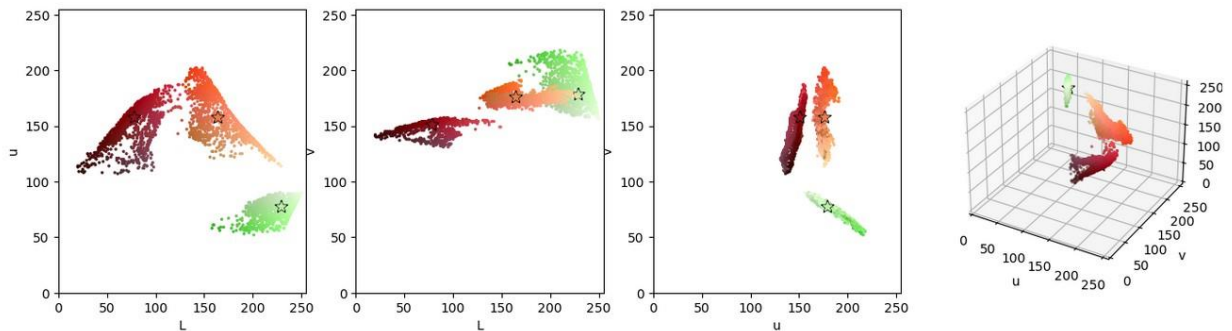


Figure 53 - graph of all segment pixels when represented in Luv colour space

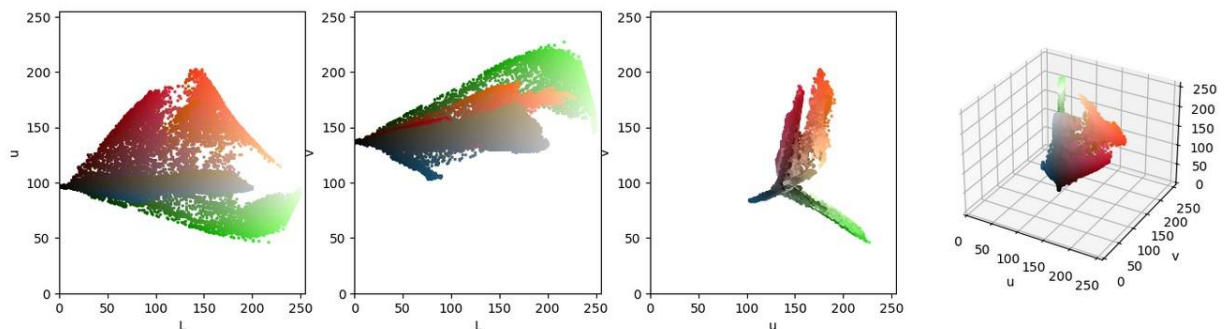


Figure 54 - graph of all image pixels when represented in the Luv colourspace

For intuition, the graphs of each channel in the Luv colour space for the detected segments and for the whole image. The stars correspond to the centroids detected by the K-medians algorithm. As it is clear from these graphs, the centroids of segments being on have a higher L (lightness) value. When observing the original image, the green segment appears to be 100 % on, the yellow segment appears to be 50 % on and the red appears to be off. So based on these graphs, these conclusions were drawn:

1. The lightness value of the segments centroid is higher than the image average if the segment is on
2. The lightness value of the segments centroid is lower than the image average if the segment is off.
3. If the segment is on, there are white pixels present in the detected segment rectangle.

The rules 1. and 2. are weak, they only work in ideal cases. The 3. rule, however, is strong when dealing with non-overexposed images.

The actual output of the algorithm is shown below:

```
[0, <BoundingBox object at 0x000001AECDFD0190>, [188.5, 240.9, 172.8], 0.9583, (0.8902, 0.)]
[1, <BoundingBox object at 0x000001AECDFD27A0>, [236.5, 126.3, 83.7], 1., (0.2051, 0.)]
```

The detection confidences are high for all segments. The first segment, thanks to the presence of white pixels, has a high confidence in being on and a low confidence in being off. The second segment shows small confidences in either being on or off, which corresponds to the personal observation.

Obviously, detecting whether a segment is off is not reliable. The same applies to segments which are half turned on. Therefore, the state estimation should be used with caution. A better approach would be tracking the appearance until both states are captured. Then, it is easy to determine, which is on and which is off.

9.9. Segment display reader

Based on the brief research in chapter 5.3. Segment display recognition, a segment display reader algorithm was proposed. The segment display reader has a form of a jupyter notebook, as a jupyter notebook enables fast development and easy visualisation.

As the word segment was defined in the context of splitting a 1D-array-type VSE in chapter 2.5. HMI components as 1D array, it will still be used here. The 7 segments of a digit in the segment display will be called a dash.

The algorithm consists of two steps, segment detection and segment recognition. For segment detection, the algorithm proposed by another work researched in chapter 5.3. Segment display recognition is used. For classification, a template matching technique is used.

9.9.1. Steps of the algorithm



Figure 55 - Blurred image of a red-LED segment display [34]

This is the input image to the algorithm. It is a blurred image with noise. It is slightly warped, but as it is an output of a linear rotator, the main axis of the display is perfectly horizontal.

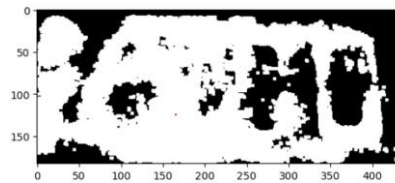


Figure 56 - The sure foreground mask of the segment display [34]

Next, the sure foreground is calculated. The sure foreground are pixels with the value channel higher than 0 and hue channel within a specified range. The sure foreground is a mask, that is used in further analysis.

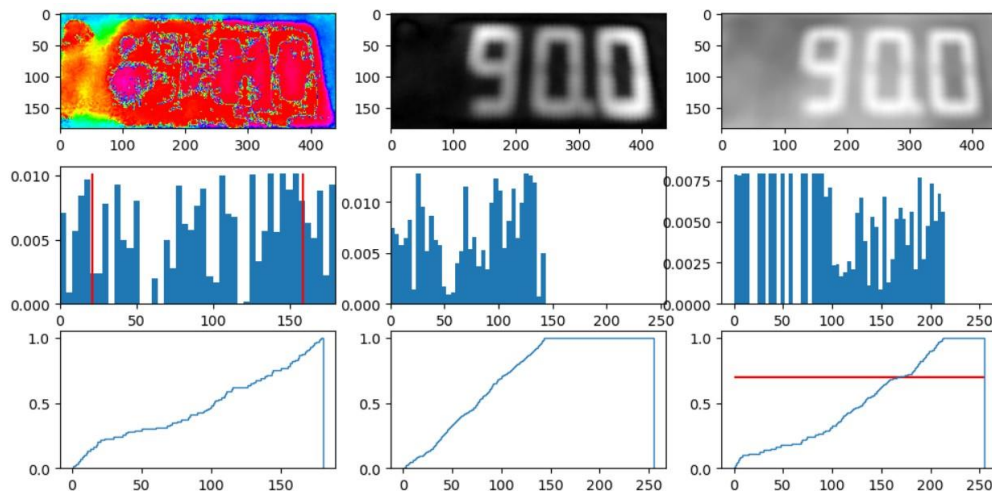


Figure 57 - Each channel of the HSV colour space, along with histograms and cumulative distribution functions [34]

Each channel of the HSV colour model is visualised, along with histograms and cumulative distribution functions. The graphs are already considering only the pixels defined by the sure-foreground mask from previous step. In the histogram of the hue channel, the two red vertical lines show the borders of the specified hue range. In the cumulative distribution function of the value channel, the horizontal red line indicates the percentage threshold for further image thresholding.

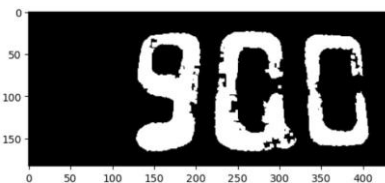


Figure 58 - Segment display with applied threshold [34]

After thresholding based on the maximum value of the value channel, 3 segments were successfully segmented. Each continuous region is then analysed and discarded, if it does not fulfil a set of rules. A bounding box around all remaining segments is drawn, corresponding to the display area. Based on the segment height to width ratio, the display area is divided into a certain number of segments.

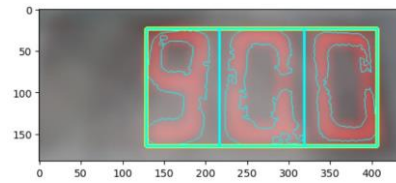


Figure 59 - Segment display with detected particular segments [34]

The content of each rectangle is assumed to be a simple segment. At this point, the image is divided into background and a certain number of segments (in this case 3). Recognition is from now done on every segment alone.

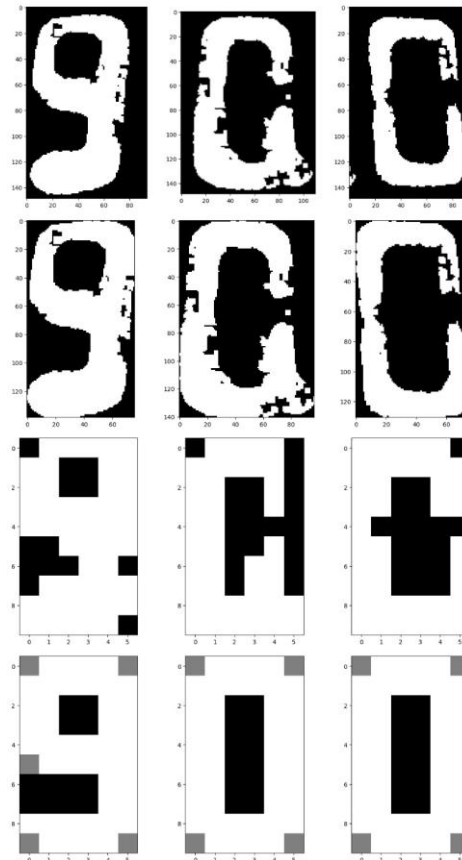


Figure 60 - The procedure of digit recognition of each segment

This image shows four stages of recognition.

1. The pre-process stage consists of cropping and adding a little context. This step assures, that the whole segment is visible. It also may cause the emergence of artifacts on the border of the image.
2. Producing "tight" segments. The segment is straightened by the linear rotator and cropped tightly. The cropping is done based on the biggest contour, which discards all artifacts that emerged in stage 1.

3. The segment is downsized to 6x9 pixels. This shape was chosen based on the nature of segment displays.
4. Lastly, the resulting image is compared with templates. Here, the corresponding templates are shown. The comparison with every template produces a confidence value equal to the degree of similarity. The number with highest confidence is chosen.

The confidences for the first segment are shown below:

```
0 0.6833
1 0.6829
2 0.6332
3 0.6999
4 0.6667
5 0.7166
6 0.6749
7 0.5250
8 0.7500
9 0.7916
segment classified as 9 with confidence 0.7916
```

The overall output of this algorithm is a number with a confidence value. In this case, the algorithm returned:

```
the number is 900 with conf 0.7916
```

The algorithm was successful in detecting and recognising the number 900. The dot present in the original display is ignored. Recognising the dot in an image of poor quality is not reliable. It is better to define the order in the understanding step.

10. Conclusion

In this thesis, an innovative and unifying system was designed and implemented for monitoring industrial human machine interfaces (HMIs) utilizing machine vision. This task was realized through a multi-step process, starting with a comprehensive review of standard HMI elements in the industry to identify a set of typical basic elements and their properties for monitoring.

A research of machine vision methods suitable for monitoring industrial HMIs was undertaken. The outcome of this research shaped the design of the system, leading to a solution that combines machine learning and traditional computer vision techniques. A key part of this task was the creation of the "Industrial visual signal elements detection" dataset. This dataset, consisting of 3978 images, was the base for training the YOLOv8m network to detect and classify basic HMI elements identified during the research.

A robust and adaptable framework was proposed and partly implemented with an object-oriented programming approach. Different algorithms, such as the rectangular perspective transformer, andon reader and segment display reader, were developed to read some of the identified VSEs and serve the purpose of demonstrating the framework's overall capability to monitor and interpret VSEs in HMIs effectively.

The final step was to test the implemented system. The performance evaluation indicated promising results for andons, cyclonic dials, segment displays and operator panels. For other VSEs, there is room for further improvement. Moreover, the system's performance in recognizing simple signal lights turned out to be poor, so this matter needs to be further researched. This shows the potential for further improvement and optimization in upcoming work.

In conclusion, this thesis has successfully designed a machine vision system that can monitor industrial HMIs, an accomplishment introducing another step towards digitalization in manufacturing. It provided a unique dataset and a robust framework that future research can build upon to improve and expand the system's capabilities. The implementation of the system and the initial results provide a proof of the applicability and effectiveness of the proposed solution. Still, there is a place for improvements and extensions of this research to further enhance the monitoring of industrial HMIs, thus increasing manufacturing efficiency and workplace safety.

BIBLIOGRAPHY

- [1] Sonka, Milan, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Fourth edition. Stamford, CT, USA: Cengage Learning, 2015.
- [2] “ElektroPrůmysl.Cz, Červenec 2022.” Accessed June 4, 2023. <https://www.elektroprumysl.cz/casopis/2022/cervenec/34/>.
- [3] Filip Geerts, Maitane Olabarria, Karmenu Vella, Ralf Reines, Adam Gontarz and CECIMO Technical Matters Group, *CECIMO Circular Economy Report, Edition April 2019*. Stamford, Cecimu, EU: 2019.
- [4] “How to Read Your Meter.” Accessed June 4, 2023. <https://www.essentialenergy.com.au/at-home/how-to-read-your-meter>.
- [5] Nilsson, Felix, Jens Jakobsen, and Fernando Alonso-Fernandez. “Detection and Classification of Industrial Signal Lights for Factory Floors.” In *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, 1–6. Fez, Morocco: IEEE, 2020. <https://doi.org/10.1109/ISCV49265.2020.9204045>.
- [6] Mataré, Victor, Tim Niemueller, and Gerhard Lakemeyer. “Robust Multi-Modal Detection of Industrial Signal Light Towers.” In *RoboCup 2016: Robot World Cup XX*, edited by Sven Behnke, Raymond Sheh, Sanem Sarel, and Daniel D. Lee, 9776:416–27. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-68792-6_35.
- [7] Ing. Mgr. Jakub, Jura, Ph.D. Online consultation of hmi practices. MS Teams, January 4, 2023.
- [8] “Your Perfect Partner for Machine Vision | STEMMER IMAGING.” Accessed June 4, 2023. https://www.stemmer-imaging.com/s/?language=en_US.
- [9] “YOLO: Real-Time Object Detection.” Accessed June 4, 2023. <https://pjreddie.com/darknet/yolo/>.
- [10] Terven, Juan, and Diana Cordova-Esparza. “A Comprehensive Review of YOLO: From YOLOv1 and Beyond.” arXiv, May 19, 2023. <http://arxiv.org/abs/2304.00501>.
- [11] “OpenCV: OpenCV Modules.” Accessed June 4, 2023. <https://docs.opencv.org/4.x/>.
- [12] Phan, Ray. “XiaohuLuVPDetection.” Python, June 3, 2023. <https://github.com/rayryeng/XiaohuLuVPDetection>.
- [13] Lu, Xiaohu, Jian Yaoy, Haoang Li, Yahui Liu, and Xiaofeng Zhang. “2-Line Exhaustive Searching for Real-Time Vanishing Point Estimation in Manhattan World.” In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 345–53. Santa Rosa, CA: IEEE, 2017. <https://doi.org/10.1109/WACV.2017.45>.
- [14] “Traffic Light Detection Dataset.” Accessed June 4, 2023. <https://www.kaggle.com/datasets/wjybuqi/traffic-light-detection-dataset>.
- [15] Mascetti, Sergio, Dragan Ahmetovic, Andrea Gerino, Cristian Bernareggi, Mario Busso, and Alessandro Rizzi. “Robust Traffic Lights Detection on Mobile Devices for Pedestrians with Visual Impairment.” *Computer Vision and Image Understanding* 148 (July 2016): 123–35. <https://doi.org/10.1016/j.cviu.2015.11.017>.
- [16] Diaz-Cabrera, Moises, Pietro Cerri, and Javier Sanchez-Medina. “Suspended Traffic Lights Detection and Distance Estimation Using Color Features.” In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 1315–20. Anchorage, AK, USA: IEEE, 2012. <https://doi.org/10.1109/ITSC.2012.6338765>.
- [17] Diaz-Cabrera, Moises, Pietro Cerri, and Paolo Medici. “Robust Real-Time Traffic Light Detection and Distance Estimation Using a Single Camera.” *Expert Systems with Applications* 42, no. 8 (May 2015): 3911–23. <https://doi.org/10.1016/j.eswa.2014.12.037>.
- [18] Behrendt, Karsten, Libor Novak, and Rami Botros. “A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification.” In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1370–77. Singapore, Singapore: IEEE, 2017. <https://doi.org/10.1109/ICRA.2017.7989163>.
- [19] Haltakov, Vladimir, Jakob Mayr, Christian Unger, and Slobodan Ilic. “Semantic Segmentation Based Traffic Light Detection at Day and at Night.” In *Pattern Recognition*, edited by Juergen Gall, Peter Gehler, and Bastian Leibe, 446–57. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-24947-6_37.

- [20] Mataré, Victor, Tim Niemueller, and Gerhard Lakemeyer. "Robust Multi-Modal Detection of Industrial Signal Light Towers." In *RoboCup 2016: Robot World Cup XX*, edited by Sven Behnke, Raymond Sheh, Sanem Sarel, and Daniel D. Lee, 9776:416–27. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-68792-6_35.
- [21] "Power Plant Indicator Light Detection System Based on Improved YOLOv5." Accessed June 4, 2023. <http://journal.bit.edu.cn/jbit/en/article/doi/10.15918/j.jbit1004-0579.2022.079>.
- [22] Popayorm, Sorawee, Taravichet Titijaronroj, Thanathorn Phoka, and Wansuree Massagram. "Seven Segment Display Detection and Recognition Using Predefined HSV Color Slicing Technique." In *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 224–29. Chonburi, Thailand: IEEE, 2019. <https://doi.org/10.1109/JCSSE.2019.8864189>.
- [23] S. Popayorm, T. Titijaronroj, T. Phoka and W. Massagram, "Seven Segment Display Detection and Recognition using Predefined HSV Color Slicing Technique," 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE), Chonburi, Thailand, 2019, pp. 224-229.
- [24] "Detect - Ultralytics YOLOv8 Docs." Accessed June 4, 2023. <https://docs.ultralytics.com/tasks/detect/>.
- [25] Roboflow. "Project Overview." Accessed June 4, 2023. <https://universe.roboflow.com/personal-protective-equipment/pp02>.
- [26] "A Complete Guide to Data Augmentation | DataCamp." Accessed June 4, 2023. <https://www.datacamp.com/tutorial/complete-guide-data-augmentation>.
- [27] "How to Organize Python Code 📁 📄." Accessed June 4, 2023. <https://guicommits.com/organize-python-code-like-a-pro/>.
- [28] Szeliski, Richard. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer, 2011. <https://doi.org/10.1007/978-1-84882-935-0>.
- [29] Onyx Industries. "1.5" Diameter Stack Lights – Compact Series." Accessed June 5, 2023. <https://stack-lights.com/products/compact-series-stack-lights/>.
- [30] "Seven-Segment Display." In *Wikipedia*, March 15, 2023. https://en.wikipedia.org/w/index.php?title=Seven-segment_display&oldid=1144723018.
- [31] "EDV111-3280-IND-RGB." Accessed June 5, 2023. <https://www.electronicdisplays.com/edv111-3280rgb-ind>.
- [32] "Blancett-B2800-Datasheet.Pdf," <https://www.instrumart.com/assets/Blancett-B2800-datasheet.pdf>
- [33] "Rekonstrukce elektročásti lyžařského vleků Tatrapoma P | J-Controls s.r.o." Accessed June 5, 2023. <https://www.j-controls.cz/lv-alsovka-2-klasterec-nad-ohri-2014?id=1&action=detail&oid=4176718&nid=10077>.
- [34] "0 10V 0 20mA 2 10V 4 20mA Analog Input Digital Display Meter With Display Head Expansion Board Electronic Component|Instrument Parts & Accessories| - AliExpress." Accessed June 5, 2023. <https://www.aliexpress.com/item/1005003715512168.html>.
- [35] Helios. "Integrační řešení | HCV Informační systémy Helios." Accessed June 5, 2023. <https://www.is-helios.cz/integra%C4%8Dn%C3%AD-%C5%99e%C5%A1en%C3%AD>.
- [36] "Vintage Machine Control Panel Stock Image - Image of Appliance, Electronic: 19539801." Accessed June 5, 2023. <https://www.dreamstime.com/stock-image-vintage-machine-control-panel-image19539801>.
- [37] "Marc Onetto on His Experience with Jidoka at GE and Amazon - Planet Lean." Accessed June 5, 2023. <https://planet-lean.com/jidoka-lean-thinking-amazon/>.

Table of Figures

Figure 1 - Andons with 1 to 5 segments [29].....	5
Figure 2 - Possibilities of segment arrangement in a segment display [30]	6
Figure 3 - Dot matrix display [31]	7
Figure 4 - A water flow meter with a cyclonic dial	7
Figure 5 - A manometer with a round clock dial	8
Figure 6 - An example of an operator panel [35].....	9
Figure 7 - A cyclonic dial representing an 1D array of features [36].....	9
Figure 8 - Diagram of influences affecting a machine vision system [8].....	11
Figure 9 - A flow monitor with a segment display with detected lines by the lu_vp_detection algorithm [32]	21
Figure 10 - A diagram of research based analysis of the monitoring system	31
Figure 11 - A diagram of conceptual design of the monitoring system.....	33
Figure 12 - A diagram of an implementation plan of the monitoring system	35
Figure 13 - A photo from a CTU lab	38
Figure 14 - A photo from a CTU lab with annotated classes	38
Figure 15 - A bar graph of occurrences of each class in the base dataset.....	41
Figure 16 - A bar graph of occurrences of each class in the augmented dataset	44
Figure 17 - A bar graph of occurrences of each class in the augmented dataset with y-axis limited to 10 000.....	44
Figure 18 - A histogram of variants counts	45
Figure 19 - An augmented image variant - noise, low brightness and horizontal flip [33].....	46
Figure 20 - An augmented image variant - high brightness, artifacts imitating fluid droplets on the camera [33]..	46
Figure 21 - An example of training data batch	48
Figure 22 - Scatter plot of progress of 4 important metrics during training.....	48
Figure 23 - Scatter plot of progress of loss functions during training.....	49
Figure 24 - confusion matrix of the trained YOLO network	50
Figure 25 - An example of a set of validation images with true labels	51
Figure 26 - An example of a set of validation images with detected labels	51
Figure 27 - 1. element of obj_crop list – a general display [32].....	57
Figure 28 - 2. element of obj_crop list – an operator panel [32].....	57
Figure 29 - 3. element of obj_crop list – a segment display [32].....	57
Figure 30 - 4. element of obj_crop list - a segment display [32]	57
Figure 31 - resized image passed to the YOLO detector [32].....	60
Figure 32 - debug image generated by the YOLO detector with detected objects [32]	60
Figure 33 - The input image of a segment display [34]	62
Figure 34 - Successfully detected operator panel [34]	63
Figure 35 - Detected operator panel with added context [34]	63
Figure 36 - Detected image with added context with lines detected by the lu_vp_detect algorithm [34]	63
Figure 37 - Detected operator panel with applied perspective warp compensation [34].....	64
Figure 38 - A comparison of a warped image (right) with a rotated-only image (left) [34].....	64
Figure 39 - Input image with hard-to-detect perspective [35]	65
Figure 40 - Input image with lines detected by the lu_vp_detection algorithm [35]	65
Figure 41 - detected general display with lines detected by lu_vp_algorithm [35]	66
Figure 42 - Input image with wrongly applied perspective warp compensation [35]	66
Figure 43 - Detected general display with wrongly applied perspective warp compensation [35].....	66
Figure 44 - An operator panel containing two cyclonic dials [36]	68

Figure 45 - edges of one detected cyclonic dial on three channels.....	68
Figure 46 - Lines detected on one detected cyclonic dial on three channels [36]	68
Figure 47 - Successfully rotated cyclonic dial [36]	69
Figure 48 - Successfully rotated operator panel [36].....	69
Figure 49 - graph of change of channel values and their derivations along a line through the image [37]	71
Figure 50 - Detected edges and centres of line segments, filtered line segments [37].....	71
Figure 51 - A graph of data used for K-medians algorithm.....	72
Figure 52 - Segments extracted from an andon [37].....	72
Figure 53 - graph of all segment pixels when represented in Luv colour space	73
Figure 54 - graph of all image pixels when represented in the Luv colourspace	73
Figure 55 - Blurred image of a red-LED segment display [34]	74
Figure 56 - The sure foreground mask of the segment display [34]	75
Figure 57 - Each channel of the HSV colour space, along with histograms and cumulative distribution functions [34]	75
Figure 58 - Segment display with applied threshold [34].....	75
Figure 59 - Segment display with detected particular segments [34]	76
Figure 60 - The procedure of digit recognition of each segment	76

Table of tables

Table 1 - Comparison of colours occurring in the reality of industrial plants and in the "Traffic light detection dataset"	23
Table 2 - Number of occurrences of each class in the base dataset.....	38
Table 3 - Comparison of performance of certain HW when training a neural network.....	47

APPENDIX

The Alumentation pipeline definition

```

# defining the augmentation pipeline
transform = A.Compose([
    A.ColorJitter(brightness=0.4,
                  contrast=0.4,
                  saturation=0.4,
                  hue=0,
                  always_apply=True), # variance in illumination
    A.Downscale(scale_min=0.2,
                scale_max=0.9,
                always_apply=False,
                p=0.25), # variance in image quality
    A.GaussNoise(50,
                 mean=0,
                 per_channel=True,
                 always_apply=False,
                 p=0.25), # variance in noise
    A.ImageCompression(quality_lower=90,
                       quality_upper=100,
                       compression_type=A.ImageCompression.ImageCompressionType.JPEG,
                       always_apply=False,
                       p=0.25/2), # variance in image quality
    A.ImageCompression(quality_lower=90,
                       quality_upper=100,
                       compression_type=A.ImageCompression.ImageCompressionType.WEBP,
                       always_apply=False, p=0.25/2), # variance in image quality
    A.ISONoise(color_shift=(0.01,0.05),
               intensity=(0, 0.5),
               always_apply=True), # variance in sensor quality
    A.MultiplicativeNoise(multiplier=(0.9, 1.1),
                          per_channel=True,
                          elementwise=True,
                          always_apply=False,
                          p=0.5), # variance in noise
    A.PixelDropout(dropout_prob=0.05,
                   per_channel=False,
    
```

```
drop_value=(147,58,22),
mask_drop_value=None,
always_apply=False,
p=0.25), # rusty dust
A.RandomFog(fog_coef_lower=0.7,
fog_coef_upper=1,
alpha_coef=0.08,
always_apply=False,
p=0.1),
A.RandomShadow(shadow_roi=(0, 1, 0, 1),
num_shadows_lower=1,
num_shadows_upper=2,
shadow_dimension=5,
always_apply=False,
p=0.10),
A.Spatter(mean=0.65,
std=0.3,
gauss_sigma=2,
cutout_threshold=0.68,
intensity=0.6,
mode='rain',
always_apply=False,
p=0.10), # water droplets
A.Spatter(mean=0.65,
std=0.3,
gauss_sigma=2,
cutout_threshold=0.68,
intensity=0.6,
mode='mud',
always_apply=False,
p=0.10), # mud particles
A.AdvancedBlur(blur_limit=(3, 5),
sigmaX_limit=(0.2, 1.0),
sigmaY_limit=(0.2, 1.0),
rotate_limit=90,
beta_limit=(0.5, 8.0),
noise_limit=(0.9, 1.1),
```

```
        always_apply=False,  
        p=0.5),  
A.Defocus(radius=(3, 5),  
        alias_blur=(0.1, 0.5),  
        always_apply=False,  
        p=0.25),  
A.MotionBlur(blur_limit=11,  
        allow_shifted=True,  
        always_apply=False,  
        p=0.5),  
A.BBoxSafeRandomCrop(erosion_rate=0.0,  
        always_apply=True),  
A.RandomScale(scale_limit=0.1,  
        interpolation=1,  
        always_apply=False,  
        p=0.5),  
A.Rotate(limit=10,  
        interpolation=1,  
        border_mode=cv2.BORDER_REPLICATE,  
        rotate_method='largest_box',  
        crop_border=False,  
        always_apply=True, ),  
A.Rotate(limit=35,  
        interpolation=1,  
        border_mode=cv2.BORDER_REPLICATE,  
        rotate_method='ellipse',  
        crop_border=False,  
        always_apply=False,  
        p=0.25),  
A.HorizontalFlip(p=0.5),  
A.OpticalDistortion(distort_limit=0.05,  
        shift_limit=0.05,  
        interpolation=1,  
        border_mode=4,  
        value=None,  
        mask_value=None,  
        always_apply=False, p=0.5),  
A.Perspective(scale=(0.05, 0.1),
```

```
        keep_size=True,  
        pad_mode=0,  
        pad_val=0,  
        mask_pad_val=0,  
        fit_output=False,  
        interpolation=1,  
        always_apply=False,  
        p=0.5)],  
bbox_params=A.BboxParams(format='yolo',  
                           min_area=400,  
                           min_visibility=0.1,  
                           label_fields=['label_list']))
```