VLAIO-TETRA
Machine Vision for Quality Control
Case 4 - Determining the proper functioning of an LED display

Contents

List of Figures

1 Introduction

1.1 Quality control in Industry 4.0

In the ever-evolving world of Industry 4.0, it is crucial for companies to ensure that their products meet the expected industry standards. To accomplish this most companies invest heavily in the process of quality control. This is a set of procedures and techniques used to detect and correct defects and errors in a product or service before it is released to the market. This process occurs at multiple stages of production, with the ultimate goal of ensuring that the final product is of the highest possible quality.

However, quality judgment is subjective in nature, and while there are standards and guidelines, the final verdict is usually left to a trained employee. But due to the tedious and repetitive nature of this work it is possible for mistakes to be overlooked and a faulty product to be released to the client. To fix this companies can utilize one or more machine vision applications to assist or replace visual inspections performed by employees eliminating the subjectivity in the control process and ensuring consistent quality control at all times.

1.2 Case 4

Case 4 centers around the quality control of LED displays. This case involves Thermote & Vanhalst, also known as TVH, a Belgian-based international supplier of various parts for forklifts, as well as other industrial, construction, and agricultural machinery.

As part of this project, TVH proposed a machine vision application to assist with testing and validating the primary display of a forklift. This display shows various information, such as speed, fuel, lights, time, and more.
The quality control process here involves looking at and validating the correct state of various LEDs. The correct output of digital and analog instruments. And the validation of some extra's like warning signs, backlighting, and more.

1.3 Goal

At current TVH's solution involves a user visually inspecting the display using the Functional Test Bench (FTB), which is a station used to connect the display and simulate it being in a forklift. Here we want to introduce an application that in the first stage, can examine the display and verify if its different parts are in the correct state. This can be achieved using a camera and a Python application that utilizes OpenCV to map the various required tests to their respective locations and execute them. A later research can then be setup where we determine whether it is possible to integrate the proposed validation system into the current FTB setup. Doing so would enable us to complete all quality control tasks in one step, significantly reducing the time required while maintaining display standards.

To assist us in this research, we are joined by a team of machine vision developers from Beckhoff Automation who will generously provide us with their expertise and optional hardware. Their knowledge not only lies in the field of machine vision but also in its implementation in an industrial setting, most notably using their TwinCAT software suite, which is used to create machine vision applications using a PLC environment.

1.4 Importance

The importance of a good quality control process cannot be underestimated. As this will make sure that the final product is of the highest possible quality and that there are no faulty products delivered to a customer. Delivering quality products more consistently will drastically increase a companies profit margin by making sure the customer remains happy and by reducing the chance of products being returned thus eliminating the cost for repairing products.
Next to the financial importance we also have the academic as this project in unique among the other cases of Machine vision for quality control because we are dealing here with a classic computer vision project. This means that, unlike other cases, we do not utilize a machine learning model in the solution. All the tests will be created using computer vision techniques.

Lastly the industrial application can not be overstated. As the modular nature of the display will lead us to per component solutions that can be utilized in many different applications. Tests that look at the state of a LED, that read the angle of a gauge or that inspect the value of a seven segment display can be utilized in many different contexts.

1.5 Structure of the paper

In the following sections we will delve deeper into the workings of case 4. We begin with a literature overview. This means that we will provide a review of existing literature pertaining to the usage of machine vision to validate the functions of a display. This may include papers that tackle the general topic or research that covers individual components like gauges or seven segment displays.

After this, in Section three we will describe the methodology used in this study. We describe the method used for collecting data and processing it. Then we describe the tests created used to evaluate the different components. For each test we will explain the thought process, the techniques utilize and how to correctly implement them.

Then we will inspect the results of the research. This contains the outcome of the approaches for each component part. We will use the created tests in practice and see if the results are accurate and consistent. Paired with this we will also inspect the difficulties that might arise as well briefly discussing different hardware possibilities.

2 Literature study

As stated, this case is in a unique position, not only due to the lack of a machine learning model but also because of the modular nature of the display. This allows us to search for the most optimal way to test each individual component before combining them. Although no research specifically exists on our current topic, the use of machine vision for quality control is a widely popular research area. Research on this topic has been conducted in multiple fields, ranging from the industrial sector to the food sector. This implies that while there are no direct studies available, other studies related to the testing of digital and analog instruments might be valuable to us. Additionally, the topic of calibration can assist us in understanding how to accurately interpret the values obtained from the different instruments.

2.1 General literature

Over the years, many different studies have been conducted in the field of computer vision. However, before one can begin, it is essential to understand the basics, as many of these techniques are applicable to practically every problem. For example, techniques like image scaling, cropping, and other transformations will be used to separate the different components from the full display. Additionally, binarization, edge detection, and Hough transforms will be utilized to identify objects such as LEDs and warning symbols. There are several online courses and books available that can provide beginners with the necessary foundational knowledge. Books like "Practical OpenCV" by Samarth Brahmbhatt and "Learning OpenCV" by Gary Bradski and Adrian Kaehler cover these aforementioned basics while also discussing hardware and basic machine vision setups.

2.2 Gauge inspection literature

Next, we are going to test if the value of a gauge matches the one provided to it. This value is displayed as a pointer pointing to a certain value, such as the speed or fuel level. Although the value is indicated by the pointer, the actual representation is the angle the pointer takes. Lower values correspond to angles on the right, while higher values are located on the left-hand side. Numerous studies have been conducted on this topic using various methods. In this case, we are particularly interested in those related to pure machine vision. We have identified three highly interesting studies relevant to this topic:
The studies in question are:

- "A Computer Vision System to Read Meter Displays"
- "Development of an Analog Gauge Reading Solution Based on Computer Vision and Deep Learning for an IoT Application"
- "Automatic Value Identification of Pointer-Type Pressure Gauge Based on Machine Vision"

The main difference among these studies lies in the process of locating and focusing on the display, while the process of identifying the value is largely similar in each case. However, in our specific case, these differences are irrelevant because our display will always be in the same static position and have the same layout. Therefore, we can simply crop out the area of interest and employ techniques similar to those described in the research. These processes involve first isolating the pointer from the rest, which can be achieved through techniques such as binarization, as demonstrated in the study conducted by Ye, Xie, and Tao (2013), or HSV filtering, as employed by Peixoto et al. (2022). Once

the pointer is identified, a Hough transform operation is performed to find the line of the pointer and extract the angle value.

2.3 Digital meter inspection literature

Here we differentiate 3 parts. The seven segment displays, the board computer and the traction.

A seven-segment display is, as the name implies, a display with seven lights that can represent any number based on which lights are on or off at a given time. Some studies have used a machine learning model to identify these numbers. While this approach can be effective and yield good results, it is also possible to achieve the same using only machine vision. In the aforementioned study by Lima, Danilo, Pereira, Guilherme, and Vasconcelos, Flávio (2008), the authors demonstrate that this can be accomplished by examining each segment's location on the display. They convert the display to a black and white image and then inspect the black pixel count for each segment. If the number of black pixels surpasses a certain threshold, it can be assumed that the segment is active. After examining the state of each segment, the resulting values are mapped to a number, providing the final value..

In a similar vein, in a blog post on Pyimagesearch by Adrian Rosebrock (2017), a similar approach is outlined for reading numbers displayed on a thermostat. Again, the method involves inspecting the different segments and determining if the number of black pixels is high enough to be considered "on."

Traction and board computer are 2 digital components that do not utilize a seven segment display and display digital values in different ways. Due to the nature of the data we want to collect these problems fall under the object detection banner mentioned in point 2.1.

It must however be noted that numerical values from the seven segment displays and board computer can also be read using an OCR. Many studies exist delving deeper into using OCR systems like tesseract to inspect numerical values. A popular topic her is the extraction of number plates as outlined in a paper by Sawalkar, Pathan, Anuja Kakade,Telvekar and Chandne (2022) where they extract the text of a number plate using opencv and an OCR

3 Methodology

In the following section, we will discuss the methodology used. First, we will discuss what data we need and how we collected said data. We then differentiate the different component categories and how to best extract each component from the display for later use. Following this, we discuss the techniques used to analyze and evaluate these components. Here we analyze the best possible way to write a test for a certain component, what OpenCV functions are utilized and why, and why certain options were not chosen

3.1 Provided Data

Before we start, it's important to note that this case is an example of a classic or traditional machine vision project. These terms are used interchangeably but mean the same thing, namely that the project is a machine vision case without machine learning being intertwined with it. Often, these two terms are used together, where machine vision is used to alter the image, acquire regions of interest, or mark certain areas, while the machine learning model is then used to make predictions based on the provided image. The two are inexorably linked, but many do not realize that they can be used separately, as in this project where we only use machine vision. Due to the lack of a machine learning model, this project will be defined as a classic/traditional machine vision project.

Due to the reasons stated above, there will be a few differences between this and other projects. One of the main differences is the dataset used. Since we do not need to train a model, our data collection is much smaller than one seen in a machine learning project.
However, the data must contain a few vital parts

- One or more images of the display in full with every component visible
- Each component must be shown in its minimum and maximum state, with some intermediate values for certain components.

The first point is clear: we need to try and extract regions of interest. To do this, we need to use the entire display and then cut out the desired part and save the coordinates. The second point is needed to determine the values associated with a certain state of a component. This varies heavily based on the component type. For example, the LEDs must be seen in their on and off states. The gauges must be seen at their minimum and maximum states, which usually translates into pointing to the far right and left, respectively. And the seven-segment displays must show both all segments off and all segments on, which translates into showing nothing and the number eight, respectively.

It is necessary to conduct further study into how one can implement these tests and the required hardware to successfully do so. Although most of the tests can still be programmed and used using the provided data, it should be acknowledged that when changing cameras, some parameters may need to be adjusted to obtain the necessary values.

Lastly, the TVH team has provided us with valuable insight into their hardware. They have provided us access to a working CAN Display, along with a guide on how to control it. They have also offered the option to conduct in-house testing and seek assistance when needed. While our current focus is primarily on the software aspect of the project, this information will be useful for future research and when we delve deeper into implementing everything in an industrial setting.

3.2 Parts

Because of the structure of the display we can differentiate 4 types of output which are the following:

Figure 1: Full Display

1. LED indicators
2. Analog instruments/indicators
3. Digital instruments/indicators
4. The extra parts (not visible on the image)

The first aspect we will discuss is the LED indicators. These indicators consist of a symbol and a corresponding LED. These LEDs are mostly independent of each other and operate in a binary manner, indicating the on/off or active/inactive status of specific functionalities of the forklift. To ensure the accurate functioning of these components, we need to develop software capable of detecting the presence and color of an LED. This software will play a vital role in validating the proper operation of the LED indicators.

Next, we have the analog/mechanical parts, such as the fuel gauge, temperature gauge, and speedometer. These components display values using a moving arm, where the position of the arm corresponds to a specific value. For example, the lower the speed, the more left-leaning the gauge, and the faster the speed, the more right-leaning the gauge. The main objective is to locate the pointer of these gauges. We represent the pointer as a line passing through the tip of the arm in the same direction. By determining the angle of the line, we can convert it to the actual value represented by the gauge.

Moving on to the digital parts, these components consist of smaller screens that display numeric information to the driver, with one exception. We need to categorize them into three sub types.

1. The first sub type is the board computer, where we need to verify if four of the same symbols are visible during the testing process.
2. The second sub type includes displays that utilize seven segment displays to show numbers. For each given number, we need to check if all segments are turned off and if all segments are turned on.
3. The third sub type is the traction display, which is a black bar indicating traction. Here, we need to verify if all segments of the traction display are either on or off.

Additionally, there are some extra features that are unrelated to each other but still require testing. The first one is the warning sign on the speedometer, represented by a red triangle that appears if there is any issue. We need to test if the warning sign is functioning correctly.

Paired with this is another feature namely the back light, which serves as a light source to illuminate the components in darker settings. We need to develop a test to verify if the back light is turned on or off when necessary.

3.3 Extracting

Now that we have identified the different parts, we need a method to extract the correct regions from the display. In the case where the display remains consistent and the individual part locations remain the same, we can define regions of interest (ROIs) as squares using two sets of (x, y) coordinates representing the top-left and bottom-right corners, respectively. By defining these coordinates, such as (x1, y1) for the top-left and (x2, y2) for the bottom-right, we can crop out the desired area using array slicing: image[y1:y2, x1:x2]. This allows us to extract the specific regions for conducting the respective tests. However, it is crucial to maintain consistency in the regions and ensure that the camera is positioned at the exact same location each time. Deviations in position may result in incorrect cropping and the omission of important aspects for the tests. To address this, a small structure can be created with a designated space to insert the display and a fixed distance for the camera above it

If the display changes frequently, making predefined ROIs impractical, an additional step needs to be implemented in each test: a search step to locate the required component. Since each component has an associated symbol, the user can try to locate the corresponding symbol and use its coordinates as a starting point to crop out the desired region. This can be achieved using template matching, where the template (symbol) is compared to different regions of the larger image using convolution. By sliding the template over the larger image and calculating the similarity measure, we can identify regions that closely match the template. Some padding must then be added to the identified region to obtain the desired part in full.

Although template matching could be used in our case, it poses resource challenges. Template matching is computationally expensive, particularly for large, high-quality images with numerous components to be found. Moreover, the operation is sensitive to changes in the template, such as size, orientation, shape, as well as other factors like lighting conditions, viewpoint, texture, and

occlusion. These variations can cause the operation to fail and result in the inability to locate the desired area. Hence, we opted for the predefined region approach to ensure successful testing and to save computational resources, ultimately making the overall test more efficient.

3.4 LED's

One of the most common tests we need to perform is the testing of an LED light. At the top of the display, there are 12 LEDs present, each indicating the status of a specific functionality, such as lights or direction indicators. The desired behavior is that the LEDs are active when necessary and off when not. When a LED is active, it usually emit a green-ish color, but there are exceptions, such as the lights, which have a blue color. Therefore, we will also need to check if the color is correct.

To identify the correct functioning of the LED, we can divide our thought process into two parts:

1. Do we see a light of a certain color?
2. Is the light source circular?

In the first part, we apply a filtering technique to isolate the specific light we are searching for. This process serves two purposes: it helps us identify the light source and removes noise caused by internal reflections within the display. And, we can perform a check to ensure the color correctness of the light.

To achieve this, we utilize HSV (Hue, Saturation, Value) filtering, which allows us to define an upper and lower bound for the desired color range. By selecting these bounds, we create a mask that retains only the colors falling within the specified range, while eliminating all other colors. This approach effectively filters out unwanted noise and focuses our attention on the target light source.

Figure 2: LED component

Figure 3: HSV filtered LED component

By applying this filtering, we can obtain an image with only the desired light source, reducing the interference from neighboring lights and noise. We can also perform a preliminary check by examining if the resulting image is fully black. If it is, we can conclude that no light is visible, indicating that the LED is off. However, relying solely on a fully black image is insufficient for a thorough check, as it is challenging to eliminate all noise, and a fully black image is rare.

To enhance the reliability of our test, we add a second check using a Hough transform. Specifically, we utilize the Hough circles operation to detect if we can find a circle of a certain size within the filtered image. If a circle is detected, it indicates the presence of a circular light source of the desired color, confirming that the LED is active. On the other hand, if no circle is detected, we can reasonably assume that the LED is either not on or not present in the captured image.

Figure 4: Active Led found

3.5 Analog instruments

There are three gauges present on the CAN display. The first two are the fuel and temperature gauges, which have a maximum range of 180 degrees. The third gauge is the speed gauge, which indicates the vehicle's speed and has a maximum range of 360 degrees. To determine the value indicated by these gauges, we need to draw a line, referred to as the pointer line, along the gauge. By finding the angle of this line, we can translate it into the corresponding value.

We have implemented two different techniques to find this pointer line.

For the first two gauges (fuel and temperature), we calculate the angle using a static validation line. This validation line serves as a reference for determining the angle of the pointer line.

In the second method, we eliminate the use of the validation line and instead draw the second point of the line at a fixed distance from the first point. This approach allows us to establish the pointer line without relying on a predetermined reference line and is useful when you cannot reliably place the reference line.

3.5.1 Using a validation line

To apply this technique, we have chosen to use gauges with a range between 0 and 180 degrees. The objective is to utilize OpenCV to find the angle indicated by the pointer and then translate it accordingly. The initial focus is on drawing a line from the tip of the pointer to the lower part of the image which we call the pointer line.

The first step is to isolate the pointer from the rest of the image. This can be achieved using HSV filtering, a technique described in section 3.4. In this example, we use a lower bound of [0, 0, 162] and an upper bound of [255, 167, 255] for the filtering range. While this range may capture some

noise, the primary goal is to ensure that the entire pointer is captured.

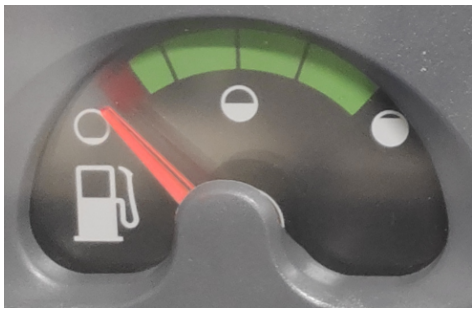

Figure 5: Analog component

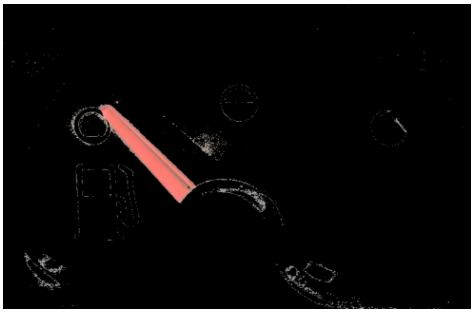

Figure 6: HSV filtered Analog component

Once we have isolated the pointer, we can proceed with extracting the lines that represent it. This process involves three steps.

Firstly, we apply Canny edge detection, which is a technique used to identify the boundaries of objects by detecting significant changes in intensity within the image. These changes are classified as edge points, helping us to identify the edges of the pointer.

Next, we utilize the Hough transform to find the shapes associated with these detected edges. Specifically, we aim to find lines that pass through these edge points. For this purpose, we use the HoughLines function in OpenCV. It's worth noting that there are two types of HoughLine functions available: HoughLines, which is the standard algorithm, and HoughLinesP, which is an extension of the standard algorithm and can extract line segments. However, since we require the full connected line, it is best to use the standard HoughLines function, which provides us with a collection of discovered lines.

The discovered lines are initially represented in polar coordinates, meaning they are defined by pairs of rho and theta values. However, for our purposes, it is more convenient to work with Cartesian coordinates. Therefore, we need to convert the lines from polar to Cartesian coordinates using the following mathematical operation:

$$x = r \cdot \cos(\theta), y = r \cdot \sin(\theta) \tag{1}$$

Next, we utilize the obtained point on the line to redefine each line in terms of two pairs of (x, y) coordinates. This representation allows us to work with the lines more effectively and perform further calculations.

Once we have the lines represented as (x, y) coordinates, we proceed to filter out the unnecessary lines. Our objective is to identify the two lines that best represent the edges of the pointer, while eliminating any overlapping lines. To achieve this, we calculate the distance between all the lines and retain only those within a certain range. Additionally, during this step, we draw the validation line as mentioned earlier. The position of this line remains fixed since our display does not change over time. It is worth noting that using a line for validation, rather than a singular fixed point, provides more consistency in the results as the precise positioning of this point is crucial and leaves

a lot of room for error.

With these three lines available, we can construct our pointer line. The pointer line is defined by two pairs of (x, y) coordinates:

1. The intersection point of the two discovered lines that represent the tip of the pointer.
2. The midpoint between the intersections of our two lines and the validation line, representing the end point of the pointer.

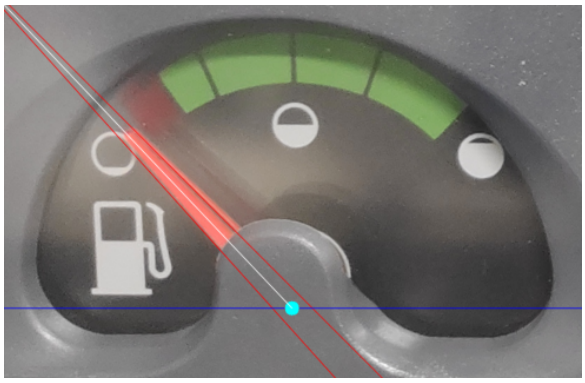

Figure 7: Fuel gauge pointer line

Extracting the angle of this white pointer line is then accomplished by finding the angle between the 2 aforementioned point using the formula

$$\theta = atan2(y2 - y1, x2 - x1) \tag{2}$$

3.5.2 Using a fixed distance

For the secondary category of gauges, we have chosen a different approach and decided not to use a validation point/line. Instead, we will draw the endpoint using an alternative technique. It is important to note that the techniques outlined in the previous section can still be applied here to find the angle. However, this provides an opportunity to demonstrate a slightly different approach, particularly for inconsistent images where adding a validation line/point may not be feasible as in the previous case.

We follow the same operations as before until we reach the step where we typically draw a validation line. We apply HSV filtering to isolate the gauge and then utilize canny edge detection combined with HoughLines transform to detect the lines. After filtering the lines, we are once again left with two lines representing the pointer.

Instead of drawing the validation line, we find the intersection point that represents the tip of the pointer. Then, we define two points at a certain distance from this intersection. The specific distance can be chosen arbitrarily as it does not affect the angle measurement. Once these two points are defined, we draw a line connecting them and find the midpoint of that line. This combination of points will provide us with the pointer line as seen on 8, from which we can determine the angle.

Figure 8: Speed gauge pointer line

3.5.3 Converting the values

In both cases after we have found the angle value we will need to convert it to the appropriate value which we can do this in 2 ways. We can calculate our minimum and maximum value and then use Linear normalization to find the desired value

Alternatively, if some values are known, we can use a process of linear interpolation. This method uses known values to estimate the new values, which can be helpful when the translation process is not straightforward.

Regardless of the chosen method, it is important to consider that analog instruments do not provide an exact indication of a value but rather point to a range. Therefore, a certain margin of error must be implemented to ensure that the evaluation works as intended.

3.6 Digital instruments

Digital instruments can encompass a wide range of data, and in our case, we specifically focus on three different types of information that we want to capture.

The first type is the seven-segment display, where we aim to extract the numbers displayed on the screen. Since the technique for extracting the numbers is the same for each segment, we will primarily focus on extracting the number itself.

The second type is the board computer, where we need to check if four of the same symbols appear on the computer. If four instances of these symbols appear, then the test is considered passed.

Lastly, we have the traction value, which is indicated by a black bar. Our objective is to test if the full black bar is working properly and in the correct order.

By employing appropriate techniques and algorithms, we can effectively extract and analyze the desired information from these digital instruments.

3.6.1 Evaluating Seven segment display

A seven-segment display is a type of electronic display commonly used to represent numbers from 0 to 9, as well as some additional characters. It consists of seven individually controllable segments that can be lit up or turned off to display specific numbers. In our case, we are interested in extracting the numerical data displayed on the seven-segment display.

To extract the displayed values, we need to identify each individual instance of a seven-segment display and map the combination of the different states to a certain value eg: if every segment is active the value displayed would be the number eight. Since we are dealing with a static display we know the location of the individual segment and can therefor extract and inspect these. To achieve this, we divide the segment into seven distinct areas of interest as seen on figure 9.

We iterate over these areas and calculate the percentage of black pixels within each area. By comparing this percentage against a predefined threshold, we can determine whether the segment is considered "on" or "off." We store the results for each segment and map them to an actual value. In our case, the mapping is represented by an array with the following values: [Top, Top right, Bottom right, Bottom, Bottom left, Top left, Middle]. For example, an array of [1, 1, 1, 1, 1, 1, 0] would correspond to the value 0, while [1, 1, 1, 1, 0, 0, 1] would represent the value 3.

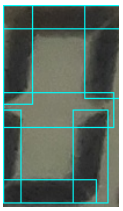

Figure 9: Seven segment parts

Since we want to test if all lights are either on or off we just have to test for the following values [0,0,0,0,0,0,0] which translates into all the lights being off, and [1,1,1,1,1,1,1] which translates into all the lights being illuminated and displaying the value 8.

3.6.2 Detecting board computer symbols

To validate the proper functioning of the board computer, a CAN signal is sent, causing four symbols to appear on the display. If all four symbols are present, the computer is deemed to be working correctly. Otherwise, if fewer than four symbols are visible, the test is considered a failure.

There are different approaches to tackle this task. One method is to utilize template matching to identify the exact symbols on the display. The theory behing Template matching is explained more in depth in subsection 3.3.

This involves using one of the required symbols as a template and transforming both the template and the target image to grayscale. The template matching function is then applied to find potential

locations of the symbol. However, there are challenges associated with this approach. First off, any deviation in one of the four symbols can result in the template matching step failing. Additionally, instead of searching for specific symbols, this method searches for areas on the image that bear the most resemblance to the template, potentially causing overlap and ambiguity leading to the need for additional filtering.

Another approach is to employ contour detection to identify the symbols on the board computer. Since the four symbols have the same width and height, contour detection can be utilized effectively. Before applying contour detection, thresholding, a process explained in 3.6.1 is performed on the image resulting in a binary image where the symbols are clearly visible.

After thresholding, the image is cropped to focus only on the screen and the black symbols. This step helps to isolate the symbols and eliminate unnecessary elements from the image. By doing so, the symbols become more distinguishable and distinct from the rest of the image.

Once the image is properly preprocessed, contour detection can be applied to identify the contours of the symbols. Contour detection identifies the boundaries of objects or shapes in an image, and in this case, it will effectively locate the contours of the symbols on the board computer. By filtering the contours based on their size, only the contours of the correct size, corresponding to the symbols, are retained.

If four valid contours are detected, it indicates that all four symbols are visible on the board computer, confirming the successful execution of the test. On the other hand, if fewer than four valid contours are found, it suggests a failure in the test, indicating that not all symbols are present as expected.

3.6.3 Evaluating traction value

The traction value is a percentage between 5 and 30 indicated by a meter filling up. At minimum there are no black cubes visible, as the traction increases a new black cube will be visible to the right of the last one with a maximum of 10. We therefor have to just validate if the full 10 can appear without trouble. If even one of them is missing then the test is considered a failure.

To achieve this, we can use a similar technique as for the board computer. First, we convert the image to grayscale, followed by thresholding to create a binary image. This process helps in isolating the black cubes and separating them from the rest of the image. Finally, contour detection is employed to identify the contour that represents the full traction bar based on its area.
If the contour representing the full traction bar is found, it indicates that all 10 black cubes are visible, and the test can be considered successful. However, if the contour is not detected, it suggests that at least one cube is missing, resulting in an invalid test.

3.7 Extra's

3.7.1 Warning sign

To determine the presence of the warning sign on the speedometer, we can employ a similar approach as before. First, we need to binarize the image by applying a threshold. In this case, the red triangle shape of the warning sign makes it easy to define a suitable threshold value that separates the red

region from the rest of the image.

Once the image is binarized, we can use contour detection to identify any objects present in the image. If a contour representing a red triangle is detected, it indicates the presence of the warning sign. On the other hand, if no such contour is found, it suggests that the warning sign is not visible, implying that there is no problem or issue.

By utilizing contour detection and analyzing the presence or absence of the red triangle contour, we can effectively determine the status of the warning sign on the speedometer.

3.7.2 Backlight

To evaluate the correct functioning of the backlight located behind the visible components, we have inspect the presence of its light indirectly. Since the backlight illuminates multiple components, we can choose one representative component to assess the presence of the backlight.

In this case, we can examine the board computer to determine if the backlight is turned on. We can extract a specific area from the board computer image and analyze the collected pixels within that area. Specifically, we are interested in evaluating the color of these pixels. We can define the desired color, in this case, an orange color, using the BGR color notation.

Once we have defined the color, we can create a filter mask based on this color and introduce a slight tolerance to account for any variations in lighting conditions. The filter mask allows us to isolate the pixels that match the desired color.

Next, we can count the number of pixels that pass the filter. While it is unlikely to have a fully orange image due to the presence of other displayed data and noise, if the filtered pixels constitute around 80% of the total pixels in the extracted area, we can confidently assume that the backlight is turned on. By extension, this indicates that the backlight is functioning, and the other components are also being illuminated by this light.

By evaluating the proportion of orange pixels within the extracted area, we can indirectly determine the operational status of the backlight without directly observing the light itself.

3.8 Using a OCR

In this final subsection, we briefly explore the potential implementation of Optical Character Recognition (OCR) to read the numerical values displayed on the digital meters. OCR technology allows for the recognition and extraction of text from scanned documents, files, and images, enabling digital processing and analysis of the content. It could be used, for example, to read the numbers from the board computer or PDF.

There are various OCR solutions available, and one popular option for Python is Python-Tesseract or pytesseract, which serves as a wrapper for Google's Tesseract-OCR Engine. To use it, you would need to install the Tesseract OCR software and the corresponding Python package. Setting certain path variables, such as the location of the Tesseract software, can help streamline the development process. Once installed, you can import the pytesseract module into your Python script and utilize

the image_to_string function to convert the numbers in your image into text data.

However, for several reasons, we have chosen not to use OCR in this context. The main reason is that the board computer value does not need to be tested, and the numerical displays in the form of seven-segment displays can be extracted and inspected using the technique described in the previous subsection (point 3.6.1). OCR is computationally demanding compared to segment extraction and inspection, and the installation and configuration process can be cumbersome. Additionally, considering the implementation with a PLC device that may not have access to OCR technology, we have opted not to utilize OCR in this particular scenario.

4 Results

In the following section we will discuss the results for the various functions designed to test the correct working of a display

4.1 Results LED's

In section 3.4, we discussed the method used to evaluate the on state of an LED. We checked for the presence of a circular source of a specific color. To ensure accurate results, our function required the HSV_lower and HSV_upper ranges, as well as an image or a part of the image to be inspected. If the evaluation criteria were not met, it indicated that the LED was either not present or not turned on.

To validate the results, we mapped all the areas where an LED was expected to be present and applied the appropriate HSV filter for each LED. Among the 12 LEDs, only 4 had different colors, and therefore have different HSV filters. We checked if each LED was on, and if it was, we drew a green border around it. If the LED was off, we drew a red border.

By applying this approach, we were able to successfully validate the working state of the LEDs. All LEDs were active and identified as such, the results can be seen on image 10

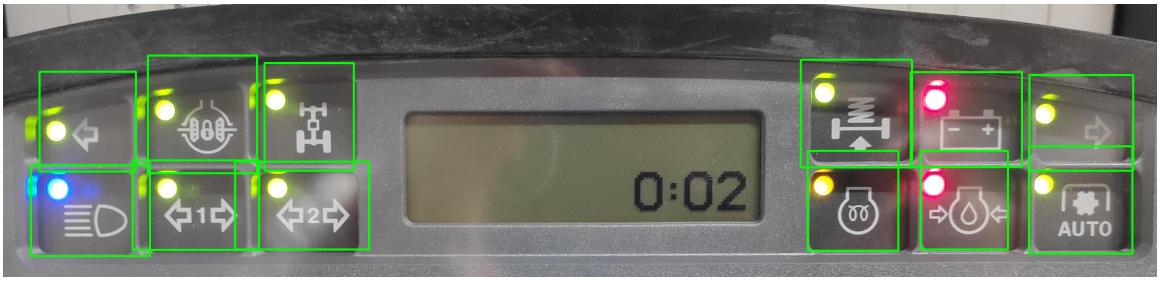

Figure 10: Detecting all active LED's

In addition to the initial test, we also conducted a second test where multiple LEDs were randomly turned on and off. This test was designed to further evaluate the robustness of our method. The results of this successfull test can be seen on figure 11

Figure 11: Detecting randomly active LED's

This successful outcome demonstrates the effectiveness of our approach in reliably testing the correct working of the LEDs, even when faced with variations in their states. It provides confidence in the

accuracy and reliability of our LED testing function.

It is important to note that the accuracy of the LED evaluation function heavily relies on the correct specification of the HSV value ranges. Choosing a range that is too narrow may result in the exclusion of valid light sources, leading to the incorrect classification of an LED as off. On the other hand, selecting a range that is too broad may introduce noise and incorrectly identify an LED as on when it is not.

To enhance the evaluation process, the function also requires an additional parameter called "expected." This parameter specifies the expected state of the LED. By comparing the expected state with the detected state, the function determines whether the test is passed or not. If the expected state matches the found state, the function returns a boolean value indicating the success of the test.

4.2 Results Analog instruments

Following the explanation provided in section 3.5, we have identified two methods to determine the angle of the pointer line on analog instruments and convert it into corresponding values. However, due to the inherent nature of analog instruments and potential measurement inaccuracies, a margin of error has been incorporated into the calculations.

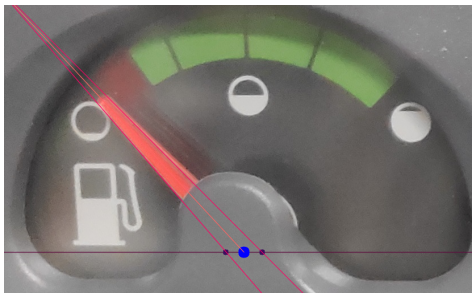

Figure 12: Gauge evaluation

```
This line has and angle of -133.02506598911802
This is a value of 1
Actual value is 0 is the value in range?: True
```

Figure 13: Gauge evaluation results

We can see that the program is able to distinguish the angle and evaluate it against the expected. Other similar tests have been conducted and have succeeded with a usual range of about 5 percent, there are however some outliers that are discussed later

In a similar way we can calculate the angle the value of a speedometer by again extracting and converting the angle of a pointer line as seen on figure 14 and 15.

In most cases, the tests for analog instruments are conducted successfully as planned. However, there are instances where outliers occur, particularly when the program is unable to detect the lines running along the edges of the pointer. This issue can be attributed to various factors, primarily related to lighting conditions and reflections on the display surface.

For example, the reflection visible in Figure 15 does not pose a problem in this particular case. However, similar lighting conditions have caused failed tests in other scenarios.

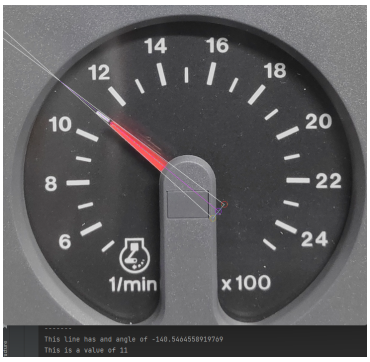

Figure 14: Gauge speed 11

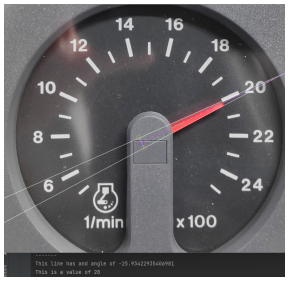

Figure 15: Gauge speed 20

To address this challenge, a potential solution will be discussed later in this section, aiming to improve the detection of pointer lines in the presence of reflections and challenging lighting conditions.

4.3 Results seven segment

There are multiple instance of seven segment displays located on the display. The techniques to read them would remain the same the only exception would be how to locate the different starting points for each instance. Once located we can utilize our functions the determine the results which give the expected value.

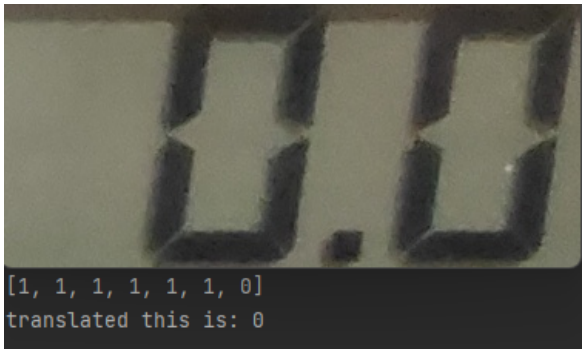

Figure 16: Results seven segment

4.4 Board computer

Detecting symbols on the board computer can, as explained in 3.6.2, be done using either template matching or contour detection. Our personal preference goes out to contour detection for reasons stated in 3.3 in practice contour detection also does not seem as the right fit for this problem. If one looks closely at image 17 we can see that template matching has the 4 figures but some areas are highlighted with a thicker rectangle. This because we are looking for the areas that look the most like our template and we don't look for the symbols specifically. Template matching searches for all areas that fall within our tolerance margin which in this case results in 65 detected targets.

Reducing the tolerance will cause the the left most symbol to not be included in the solution as this one deviates the most out of the 4 meaning our test cannot be reliably concluded

The detection of symbols on the board computer can be achieved using either template matching or contour detection methods, as discussed in subsection 3.6.2. Although contour detection may initially seem less suitable for this problem, it offers certain advantages over template matching.

In the case of template matching, as shown in image 17, multiple overlapping locations are found due to the tolerance margin. While the symbols are enclosed within the thicker rectangles, there are 65 detected targets in total. Reducing the tolerance may result in the exclusion of the leftmost symbol, which deviates the most among the four symbols. This inconsistency poses challenges in reliably concluding the test.

.

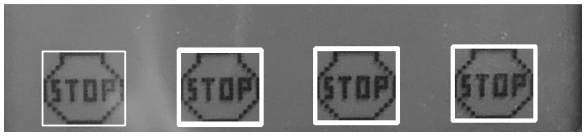

Figure 17: Board computer template matching

On the other hand, contour detection provides a more efficient and reliable approach. By using contour detection and drawing bounding boxes around the symbols, only four instances are detected, as seen in the result. Although the bounding boxes may not perfectly encase the symbols like in template matching, this method ensures that exactly four symbols are detected consistently as you can see on image 18.

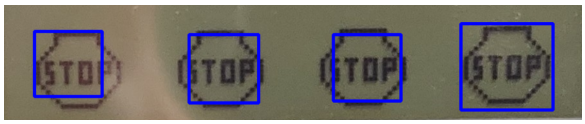

Figure 18: Board computer contour detection

Therefore, despite the visual appeal of the perfectly encased symbols in template matching, contour detection proves to be a more robust and accurate option, detecting the desired number of symbols without the ambiguity of multiple matches.

4.5 Result Traction

In the case of the traction indicator, its functionality is to display a horizontal bar that fills up as the value increases. To ensure that the bar works correctly, we only need to test if the bar fills up completely when it should. If any part of the bar is missing, it indicates a potential issue.

When comparing figure 19 (representing a correct full traction bar) with figure 20 (representing a faulty one), we can observe the following results. The software successfully detects the correct full traction bar in the case of the good image, indicating that the test is passed. However, in the case of the bad image, the faulty bar is not detected, indicating a failure in the test.

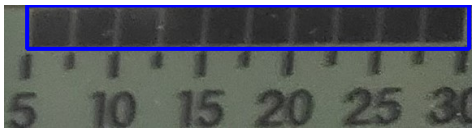

Figure 19: Correct Traction indicator

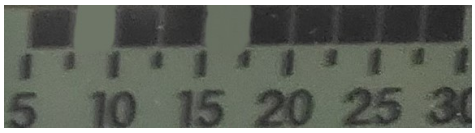

Figure 20: Faulty Traction indicator

This demonstrates that the software is capable of accurately identifying the presence or absence of a complete traction bar, allowing us to verify the proper functioning of this particular display component.

4.6 Testing the warning sign

To test the warning sign on the display, we cropped out the specific area where the indicator would be located. Using contour detection, we were able to detect the presence of a symbol in that region. If a symbol is detected, it indicates that the warning sign is on and there might be a problem.

Figure 21: Warning symbol

4.7 Testing backlight

To test the presence of the backlight on the display, we focused on the area where the backlight is expected to illuminate the components, such as the board computer and the LED indicators. By capturing a part of the board computer and analyzing the colors present, we can determine if the backlight is active or not.

Using color filtering techniques, we filtered out the non-relevant colors and extracted the pixels within the desired color range. In the case of an active backlight, the resulting image would contain visible pixels, indicating that the light is on. Conversely, if the image is completely black, it signifies that the backlight is off.

In Figure 22, the filtered image shows visible pixels, confirming the presence of an active backlight. On the other hand, Figure 23 displays a fully black image, indicating that no light was detected within the specified color range, indicating that the backlight is off.
This approach allows us to assess the functionality of the backlight by evaluating the presence or absence of the expected color, providing valuable information about its working status.

4.8 Hardware testing

In the context of machine vision, the hardware components play a crucial role in achieving accurate and reliable results. Throughout this research paper, our focus has primarily been on the software

Figure 22: Active Back light detected

Figure 23: Non active Back light detected

and techniques used for component evaluation. However, it is important to acknowledge that utilizing high-quality hardware, including cameras, lenses, and lighting, can significantly enhance the performance of a machine vision application.

In our research, we worked with images provided by TVH, where our liaison activated different components and captured pictures using a mobile device. While these images were sufficient for our research purposes, we encountered some challenges related to the hardware setup, including:

1. Inconsistent lighting
2. Inconsistent distance
3. internal and external reflection

Figure 24: Full display image taken with a phone

To address these challenges evident in Figure 24, we utilized a specific hardware setup consisting of an IDS camera, specifically the U3-3280CP Rev.2.2 model. Additionally, we incorporated additional accessories such as a polarization filter lens and a pair of small polarized lights. These hardware additions can help mitigate the issues mentioned earlier.
Note that due to the limited hardware we had at our disposal we were not able to get the full display in frame without rotating it. However it is evident from the comparison between the images captured using the IDS U3 camera and the previous images that the hardware upgrade has significantly improved the quality of the captured images. The higher resolution, improved dynamic range, and enhanced sensitivity of the U3 camera have contributed to sharper and more detailed images.

The most notable improvement is the reduction in reflections on the display. With the use of the polarization filter lens and polarized lights, the unwanted reflections are minimized, resulting in

Figure 25: Full display using a IDS camera

better visibility of the components and a more uniform lighting across the display.

However, it is important to note that the polarization filter and other hardware modifications may have an impact on the overall brightness of the image. As mentioned, the images captured with the U3 camera may appear darker compared to the previous images. This trade-off between image brightness and reduced reflections should be taken into consideration when adjusting software settings or selecting additional hardware components.

To test this we edited the GAIN value using the IDS peak cockpit tool.

Figure 26: IDS with Increased GAIN

Increasing the gain helps with the darker areas but also intensifies the visible reflection. Therefore,

we may need to explore alternative solutions to minimize the reflection. Resetting the gain and removing the polarization filter might yield better results, as shown in the following image.

Figure 27: IDS image without polarization filter

It is immediately apparent that the filter is necessary, as many parts are overexposed and there are numerous bright spots that undoubtedly affect the accuracy of our tests.

Subsequently, we attempted to combine the camera and lens with a pair of polarized light sources. We positioned them in different configurations to examine the potential results. Firstly, we placed them on opposing sides next to the short side, as seen in image 28. Alternatively, we positioned them opposing each other next to the long side, as shown in 29. In both cases, the resulting image is clear and of high quality. However, the lighting is insufficient, and certain areas, particularly the middle, remain as dark as before.

Figure 28: Lights opposing sides, short side

Figure 29: Lights opposing sides, long side

Next, we positioned the lights next to each other, which resulted in more uniform lighting and provided us with better detail. Unfortunately, we did not have enough components to fully test this solution, but the initial results are promising.
Lastly, we positioned the lights next to the camera perpendicular to the display. At first glance, this seemed like a poor decision as it created two large bright spots. However, we discovered that if we could move these bright spots to cover a non-essential area, such as the blank area next to the speed gauge, we might be able to use this setup. The resulting lighting was more uniform and clean, which could potentially be sufficient to illuminate the important parts of the display.

Figure 30: Lights long side next to each other

Figure 31: Lights above display

As demonstrated, the use of a higher-quality camera significantly improved the clarity and quality of the results. When combined with a pair of polarized lights, it has the potential to eliminate certain issues discussed earlier, such as the challenges with the seven segment displays. However, further research is needed to determine the optimal utilization of these components in practical machine vision applications. This topic will be a focus of future research.

5 Conclusion

In conclusion, this research paper delved into the evaluation of various components in a display using machine vision techniques. The tests conducted on LEDs, gauges, seven segment displays, traction indicators, warning signs, and board computers demonstrated the effectiveness of the developed software algorithms in detecting and assessing the correct functioning of these components.

The research emphasized the significance of selecting appropriate thresholds, HSV ranges, and other parameters to ensure accurate evaluation. It also identified several challenges that can impact the accuracy and reliability of the assessments, including inconsistent lighting, variable distances, and internal/external reflections.

To mitigate these challenges, the study explored the utilization of high-grade hardware such as the IDS U3-3280CP Rev.2.2 camera, polarization filter lens, and polarized lights. The incorporation of these hardware components resulted in improved image quality, with reduced reflections and enhanced detail.

Overall, the research findings highlight the potential of machine vision in the evaluation of display components. The combination of sophisticated software algorithms and advanced hardware presents an opportunity to achieve more reliable and accurate assessments. Future research endeavors will focus on refining the hardware setup, optimizing image processing techniques, and addressing the remaining challenges to develop a robust machine vision system for display evaluation.

6 Bibliography

References

[1] HAISAM ABDEL MALAK. 9 Discreet Disadvantages of Optical Character Recognition.

[2] Sawalkar Ankita, Pathan Anas, Kakade Anuja, Telvekar Prachi, and Chandne Bhushan. NUMBER PLATE RECOGNITION SYSTEM USING PYTESSERACT & OPEN CV. *International Research Journal of Modernization in Engineering Technology and Science*.

[3] Gary R. Bradski and Adrian Kaehler. *Learning OpenCV: computer vision with the OpenCV library*. Software that sees. O'Reilly, Beijing, 1. ed., [nachdr.] edition, 2011.

[4] Samarth Brahmbhatt. Practical OpenCV, Samarth Brahmbhatt.pdf.

[5] Danilo Alves de Lima, Guilherme Augusto Silva Pereira, and Flávio Henrique de Vasconcelos. A COMPUTER VISION SYSTEM TO READ METER DISPLAYS. 2008.

[6] João Peixoto, João Sousa, Ricardo Carvalho, Gonçalo Santos, Joaquim Mendes, Ricardo Cardoso, and Ana Reis. Development of an Analog Gauge Reading Solution Based on Computer Vision and Deep Learning for an IoT Application. *Telecom*, 3(4):564–580, October 2022.

[7] Sorawee Popayorm, Taravichet Titijaroonroj, 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)*, pages 224–229, Chonburi, Thailand, July 2019. IEEE.

[8] Adrian Rosebrock. Recognizing digits with OpenCV and Python, February 2017.

[9] Tuomas Savolainen, Daniel Keith Whiter, and Noora Partamies. Automatic segmentation and classification of seven-segment display digits on auroral images. *Geoscientific Instrumentation, Methods and Data Systems*, 5(2):305–314, July 2016.

[10] Wang Shuangxi, Hu Mengxia, Ye Jiaxing, Xiao Yin, and Fan Zhun. An Auto Backlight Screen Inspection Equipment Based on Machine Vision. In *2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, pages 98–101, Nanchang, China, June 2015. IEEE.

[11] R. Smith. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, pages 629–633, Curitiba, Parana, Brazil, September 2007. IEEE. ISSN: 1520-5363.

[12] OpenCV team. OpenCV: OpenCV modules.

[13] Xiaofeng Ye, Dailiang Xie, and Shan Tao. Automatic Value Identification of Pointer-Type Pressure Gauge Based on Machine Vision. *Journal of Computers*, 8(5):1309–1314, May 2013.