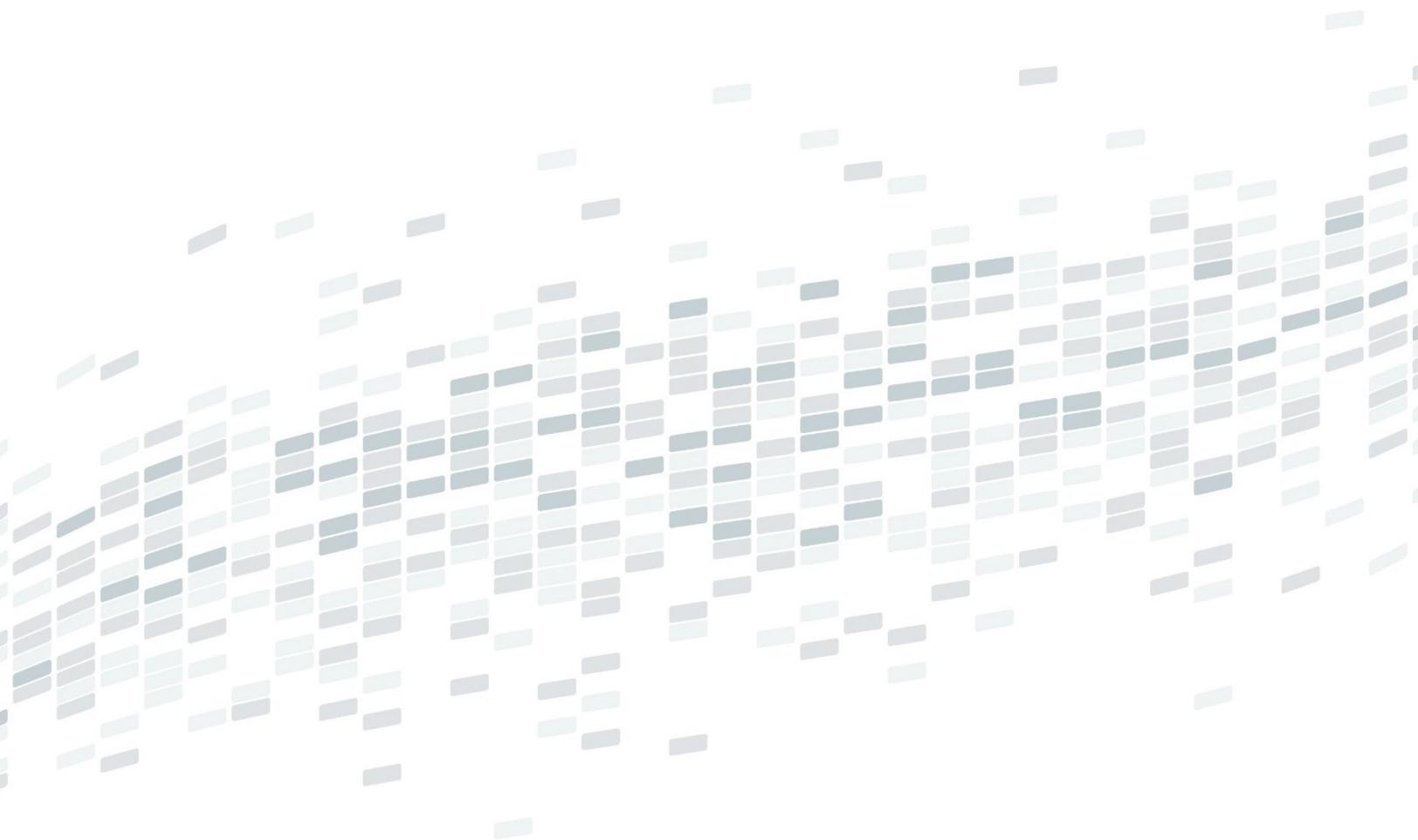


Visteon®

Detection of telltales,
intensity level and colors



Abstract

A key characteristic of the automotive industry is its highly automated mass production and emphasis on the strictest requirements for quality and high-cost sensitivity. To support these methods, industrial image processing – or machine vision – offers an essential technology for the auto industry. This innovation is used to optimize a variety of processes in the value chain – including production, quality assurance and logistics. In automotive systems, a telltale light acts as a warning. It may be an indicator of a malfunctioning system, signposted by a binary (on/off) illuminated light, symbol or text legend, or used as a running indication of the status of the vehicle showing turn signals, high beam or transmission mode, for example.

As vehicles have become increasingly complex, more indicator lights have been added to the instrument cluster to show the status of safety and convenience systems, presenting an important element of the overall cockpit experience. The detection of the telltales, intensity level and colors can be done automatically with the machine vision methods as well as tools and integrated in a vision-automated test bench.

Author Keywords

Telltales; intensity; intensity level; color; color image; grey scale; grey scale image; RGB; red; green; blue; HSL; hue; saturation; luminance; value; threshold; ROI; region of interest; HMI; human machine interface; machine vision; vision recognition; inspection; validation; automation; test bench

Introduction

The telltale symbols (Figure 1) represent more than 160 different ISO-approved illuminated color-coded graphical symbols on the dashboard that gives information regarding the status, or correct or incorrect function, of circuits, systems, auxiliaries or devices. The symbols illuminate in blue, green, red, or yellow. A blue illuminated symbol on the dashboard is normally only used to inform the driver that the main beam headlamps are turned on. A green illuminated symbol on the dashboard informs the driver that a normal system, such as the dip headlamps or direction indicators, etc., is in a service. A red illuminated symbol on the dashboard informs the driver a system is not in a drivable state and the vehicle must not be driven, e.g. the handbrake is on, or oil pressure is low, etc. A yellow illuminated symbol on the dashboard informs the driver that an auxiliary system, such as the rear screen heater, is in service, e.g. the rear screen heater [1].



Figure 1. Telltales

The dashboard is a device under test (DUT). Depending on individual OEM architecture, a telltale can be created as a telltale lamp (the source of illumination for the telltale symbol) or as a graphical symbol on a display. Machine vision technologies are used in a host of industrial applications to garner rapid and unambiguous detection of objects, which results in the automation and acceleration of validation processes. These methods are used to inspect graphical components of human machine interface (HMI) of the DUT. This automation is a special feature that makes machine

vision so desirable. Definitions for machine vision vary, but all of them include “the technology and methods used to extract information from an image on an automated basis”. The main stages for building the machine vision system follow a standard process: selection of image-acquisition hardware, calibration of image, and selection of tools and methods for creating and deploying machine vision operations in research and development for an efficiency recognition. This article will be focused on the detection of telltales, intensity level and color.

1. Problem Statement

- A telltale has more than one color

If a telltale has more than one color, RGB or true-color images are used. RGB, or true-color images, are 3-D arrays that assign three numerical values to each pixel, with each value corresponding to the red, green and blue (RGB) image channel component, respectively. RGB is the most common color space used for digital image representation because it corresponds conveniently to the three primary colors, which are mixed for display on a monitor or similar device. We can easily separate and view the red, green and blue components of a true-color image. It is important to note that the colors typically present in a real image are nearly always a blend of color components from all three channels. A common misconception, for example, is that items perceived as blue will appear only in the blue channel, and so forth. While items perceived as blue will certainly appear brightest in the blue channel (i.e. they will contain more blue light than the other colors) they will also have milder components of red and green. If we consider all of the colors that can be represented within the RGB representation, we appreciate that the RGB color space is essentially a 3-D color space (cube) with axes R, G and B (Figure 2). Each axis has the same range: 0-255 for the common 1 byte per color channel, 24-bit image representation. The color black occupies the origin of the cube (position (0, 0, 0)), corresponding to the absence of all three colors; white occupies the opposite corner (position (255, 255, 255)), indicating the maximum amount of all three colors. All other colors in the spectrum lie within this cube.

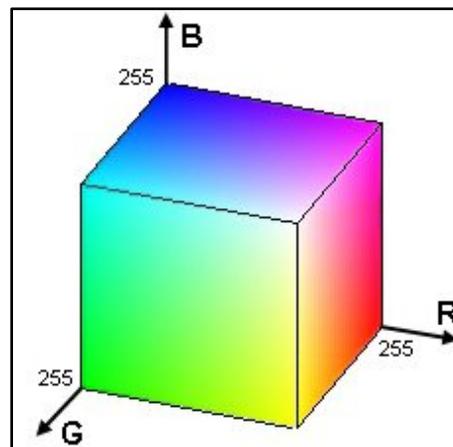


Figure 2. 3-D color space (cube)

The RGB color space is based upon the portion of the electromagnetic spectrum visible to humans (i.e. the continuous range of wavelengths in the approximate range 400-700 nm). The human eye has three different types of color receptor over which it has limited (and non-uniform) absorptency for each of the red, green and blue wavelengths.

In digital image processing, we use a simplified RGB color model (based on the CIE color standard of 1931) that is optimized and standardized for graphical displays. However, the primary problem with RGB is that it is perceptually nonlinear. By this we mean that moving in a given direction in the RGB color cube (Figure 2) does not necessarily produce a color that is perceptually consistent with the change in each of the channels. For example, starting at white and subtracting the blue component produces yellow; similarly, starting at red and adding the blue component produces pink. For this reason, RGB space is inherently difficult for humans to work with and reason about because it is not related to the natural way we perceive colors. As an alternative we may use perceptual color representations such as HSV (hue, saturation and value) [2] (section 1.3.1).

For detection of a telltale, the relation between average red (1), average green (2) and average blue (3) components is measured in the region of interest (ROI) where the telltale appears.

$$RED_{avg} = \frac{1}{n*m} \sum_{i=1}^n \sum_{j=1}^m R_{i,j} \quad (1)$$

$$GREEN_{avg} = \frac{1}{n*m} \sum_{i=1}^n \sum_{j=1}^m G_{i,j} \quad (2)$$

$$BLUE_{avg} = \frac{1}{n*m} \sum_{i=1}^n \sum_{j=1}^m B_{i,j} \quad (3)$$

R is the red color component, G is the green color component and B is the blue color component of a pixel in a ROI with width n and height m .

For example:

If the following telltale in the green ROI TT CLR1 has color BLUE, the average red component of the pixels is 10.0, the average green component of the pixels is 50.5 and the average blue component of the pixels is 84.1:

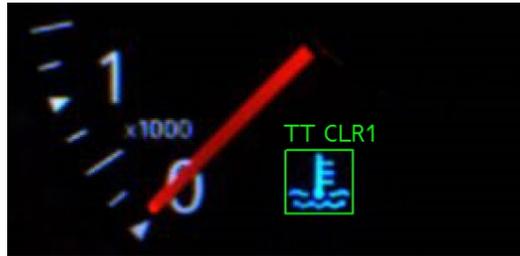


Figure 3. A blue TT in color image

If the following telltale in the green ROI TT CLR1 is OFF, the average red component of the pixels is 0.6, the average green component of the pixels is 0.6 and the average blue component of the pixels is 0.7:



Figure 4. TT is OFF in color image

State	Average Red in ROI	Average Green in ROI	Average Blue in ROI
ON	10.0	50.5	84.1
OFF	0.6	0.6	0.7

Table 1. Relation between the color components of the telltale

On the basis of the information above, the following conclusion can be made for available blue color:

$$BLUE_{avg} > RED_{avg} + GREEN_{avg} \quad (4)$$

If average blue is more than average green plus average green, the telltale is BLUE.

- A telltale has single color

If a telltale has a single color, a grey scale image is used. Intensity or grey scale images are 2-D arrays that assign one numerical value to each pixel that is representative of the intensity at this point [2] (section 1.3.1). We can convert to a grey scale image using a simple transform. Grey scale conversion is the initial step in many image analysis algorithms, as it essentially simplifies (i.e. reduces) the amount of information in the image. Although a grey scale image contains less information than a color image, the majority of important, feature-related information is maintained, such as edges, regions, blobs, junctions and so on. Feature detection and processing algorithms then typically operate on the converted grey scale version of the image [2] (section 1.4.1.1)

The conversion of color to a grey scale image can be done on the basis of the following formula:

$$\text{Grey_Picture}_{n,m} = \sum_{i=1}^n \sum_{j=1}^m (\alpha * R + \beta * G + \gamma * B)_{i,j} \quad (5)$$

Where (n, m) indexes an individual pixel within the grey scale image. R is the red color component, G is the green color component and B is the blue color component of one pixel in the image. The grey-scale image is essentially a weighted sum of the red, green and blue color components or channels. The weighting coefficients (α , β and γ) are set in proportion to the perceptual response of the human eye to each of the red, green and blue color channels, and a standardized weighting ensures uniformity (NTSC television standard, $\alpha = 0.2989$, $\beta = 0.5870$ and $\gamma = 0.1140$). The human eye is naturally more sensitive to red and green light; hence, these colors are given higher weightings to ensure that the relative intensity balance in the resulting grey scale image is similar to that of the RGB color image. RGB to grey scale conversion is a noninvertible image transformation: the true color information that is lost in the conversion cannot be readily recovered [2] (section 1.4.1.1).

In a grey scale (Figure 5) image, every pixel has an intensity with a value from 0 to 255, where black is 0 and white is 255.

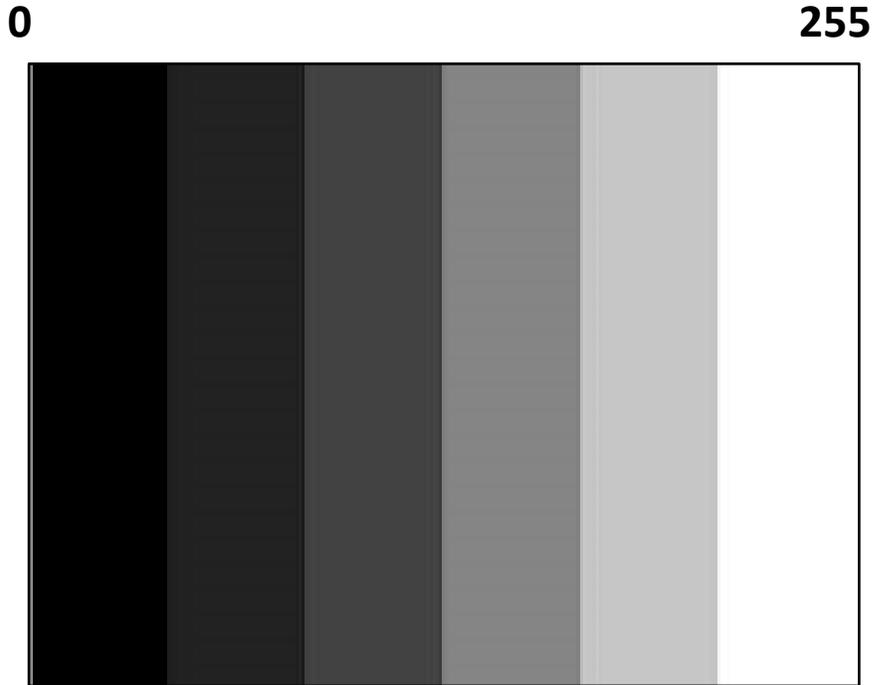


Figure 5. Grey scale: min to max

The recognition is done by measuring the average intensity level of the region of interest (ROI) where the telltale appears.

$$\text{ROI_IL}_{\text{avg}} = \frac{1}{n*m} \sum_{i=1}^n \sum_{j=1}^m IL_{i,j} \quad (6)$$

IL is intensity level of a pixel in a ROI, the width of ROI is n , and the height of ROI is m .

A threshold value (Figure 6) is set to distinguish whether the telltale is ON or OFF.

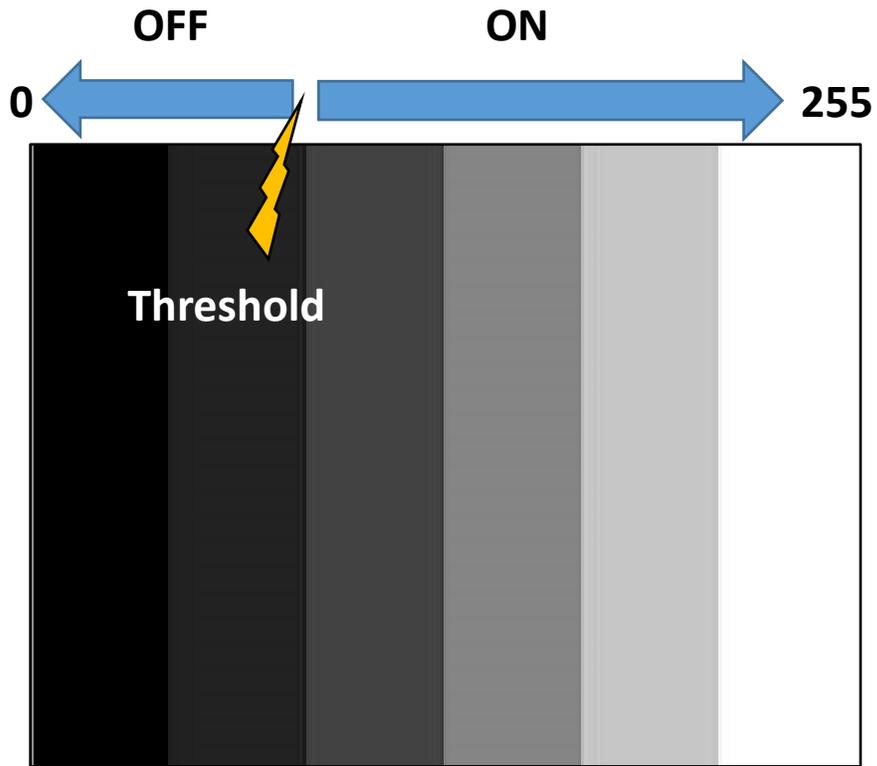


Figure 6. Threshold

For example:

If the following telltale is ON, the average intensity of the pixels in the green ROI TT1 is 45.68 (Figure 7):

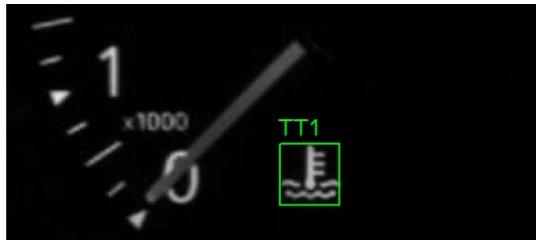


Figure 7. A telltale in state ON

If the same telltale is OFF, the average intensity of the pixels in the green ROI TT1 is 0.57 (Figure 8):

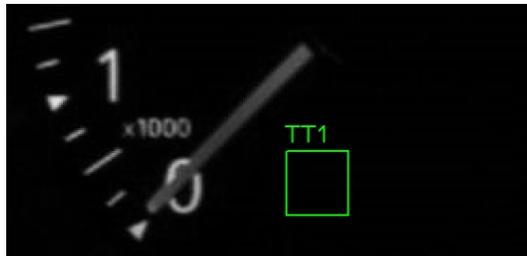


Figure 8. A telltale in state OFF

State	Average Intensity in ROI
ON	45.68
OFF	0.57

Table 2. Relation between ON and OFF states of the telltale

On the basis of Table 2 the following threshold can be defined:

$$\text{Threshold} = \frac{AVG_ON + AVG_OFF}{2} \quad (7)$$

Threshold = $(45.68 + 0.57)/2 \approx 23$. And now we can conclude: if the average intensity is less than 23, the state of the telltale is OFF; otherwise the state of telltale is ON.

2. Design of Alternative Solution

In the previous methods, a particular case is defined and the full range (Figure 9) of a color cannot be detected.

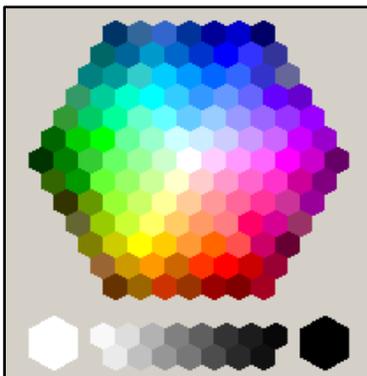


Figure 9. Full range of the colors presented in a hexagon

Perceptual color space is an alternative way of representing true-color images in a manner that is more natural to the human perception and understanding of color than the RGB representation. Many alternative color representations exist, but here we concentrate on the hue, saturation and luminance (HSL) color space popular in image analysis applications. Changes within this color space follow a perceptually acceptable color gradient. From an image analysis perspective, it allows the separation of color from lighting to a greater degree. An RGB image can be transformed into an HSL color space representation [2] (section 1.4.2).

Each of these three parameters can be interpreted as follows:

H (hue) is the dominant wavelength of the color, e.g. red, blue, green (Figure 10).



Figure 10. Hue

S (saturation) is the “purity” of a color in the sense of the amount of white light mixed with it – for example, red. (Figure 11).



Figure 11. Saturation

L (luminance), also known as value, is the brightness of the color. (Figure 12).



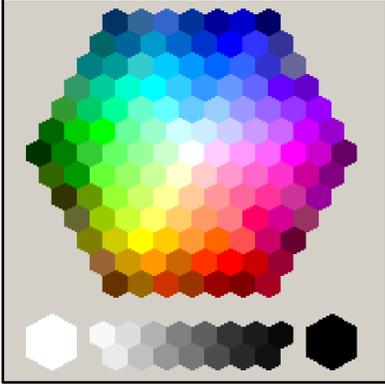
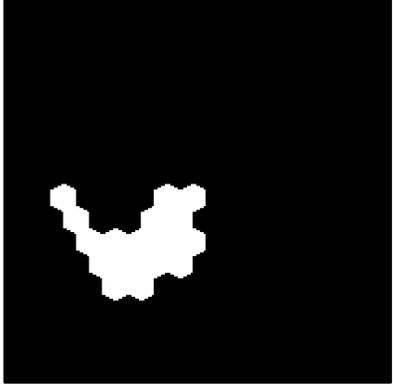
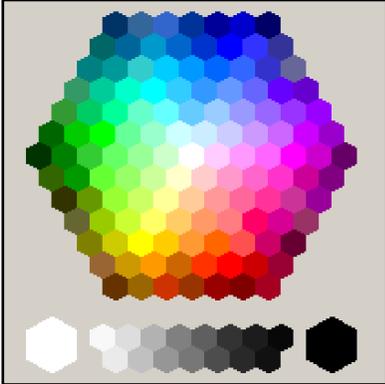
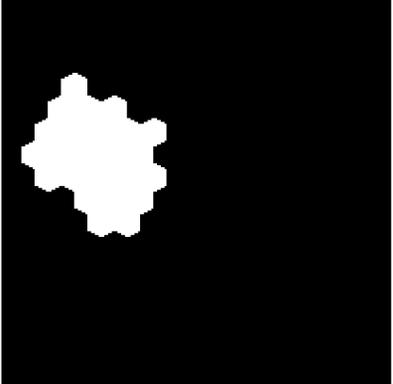
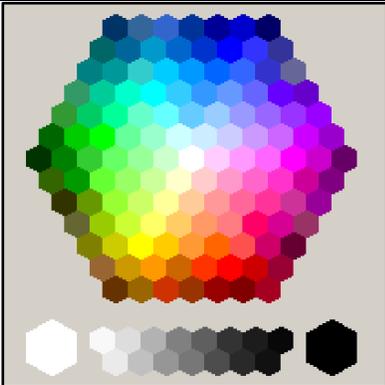
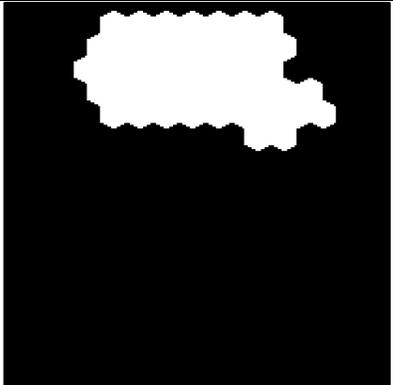
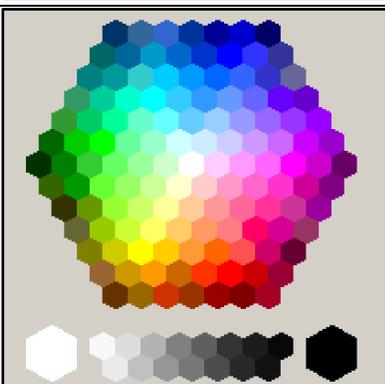
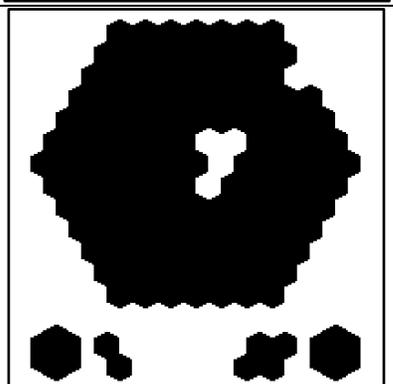
Figure 12. Luminance

These components have values from 0 to 255 in NI Vision Builder for Automated Inspection [3].

By examining the individual color channels of images in the HSL space, we can see that image objects are more consistently contained in the resulting hue field than in the channels of the RGB representation, despite the presence of varying lighting conditions over the scene. As a result, HSL space is commonly used for color-based image segmentation using a technique known as color slicing. A portion of the hue color is isolated as the color range of interest, allowing objects within that color range to be identified within the image. This ease of color selection in HSL color space also results in its widespread use as the preferred method of color selection in computer graphical interfaces and as a method of adding false color to images [2] (section 1.4.2).

The following color threshold ranges are defined as primary. They are set in six arrays: H_{min} – minimum value of hue, H_{max} – maximum value of hue, S_{min} – minimum value of saturation, S_{max} – maximum value of saturation, L_{min} – minimum value of luminance, L_{max} – maximum value of luminance

Input	Colors	Values	Output preview color in white
	BLACK or very dark colors	H_{min}^0 H_{max}^0 S_{min}^0 S_{max}^0 L_{min}^0 L_{max}^0	
	RED	H_{min}^1 H_{max}^1 S_{min}^1 S_{max}^1 L_{min}^1 L_{max}^1	

	<p>YELLOW or AMBER</p>	H_{min}^2 H_{max}^2 S_{min}^2 S_{max}^2 L_{min}^2 L_{max}^2	
	<p>GREEN</p>	H_{min}^3 H_{max}^3 S_{min}^3 S_{max}^3 L_{min}^3 L_{max}^3	
	<p>BLUE</p>	H_{min}^4 H_{max}^4 S_{min}^4 S_{max}^4 L_{min}^4 L_{max}^4	
	<p>GREY or not so pure colors</p>	H_{min}^5 H_{max}^5 S_{min}^5 S_{max}^5 L_{min}^5 L_{max}^5	

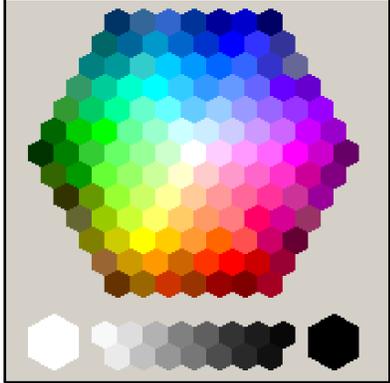
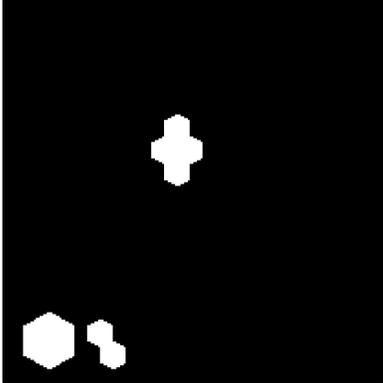
	WHITE or very light colors	H_{min}^6 H_{max}^6 S_{min}^6 S_{max}^6 L_{min}^6 L_{max}^6	
---	-------------------------------------	--	--

Table 3. Color threshold ranges

Of course, if a project has a need for other colors, a new range can be defined, as some of the current color ranges have to be changed.

Usually the color of the background of the object is the most frequently occurring. Due to this reason, background color must be ignored. The first defined color with index 0 is the background of the image for analysis. If the background color is occurred in more than 99% in pixels in the ROI, the state is OFF (Formula 8). Otherwise the state is ON with the correct recognized color (Formula 9).

$$\text{State} = \begin{cases} \text{OFF, Color}^0, & \text{if } \text{Cnt}^0 > 99\% \text{ in the ROI} \\ \text{ON, Color}_{\max} = \text{Color}^k, & \text{if } \text{Cnt}^k = \text{Cnt}_{\max}, \text{ for } k > 0 \end{cases} \quad (8)$$

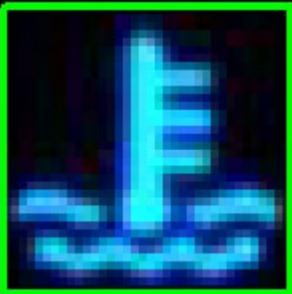
A correct recognized color (Color_{\max}) has the greatest number of the pixels (Cnt_{\max}). The value k is an incremented index of defined foreground colors.

Every defined color is scanned in the ROI of the image for analysis. If the current pixel with HSL data belongs to the defined threshold, the number of the pixels (Cnt^k) for the defined color is incremented with one (Formula 9). Where i is the index of the current scanned pixel in the ROI.

$$\text{Cnt}^k = \begin{cases} \text{Cnt}^k + 1, & \text{if } (H_i \leq H_{\min}^k \text{ OR } H_i \geq H_{\max}^k) \text{ AND } (S_{\min}^k \leq S_i \leq S_{\max}^k) \text{ AND } (L_{\min}^k \leq L_i \leq L_{\max}^k), \text{ for } k = 1 \\ \text{Cnt}^k, & \text{if } (H_{\min}^k < H_i < H_{\max}^k) \text{ OR } (S_i < S_{\min}^k \text{ OR } S_i > S_{\max}^k) \text{ OR } (L_i < L_{\min}^k \text{ OR } L_i > L_{\max}^k), \text{ for } k = 1 \\ \text{Cnt}^k + 1, & \text{if } (H_{\min}^k \leq H_i \leq H_{\max}^k) \text{ AND } (S_{\min}^k \leq S_i \leq S_{\max}^k) \text{ AND } (L_{\min}^k \leq L_i \leq L_{\max}^k), \text{ for } k > 1 \\ \text{Cnt}^k, & \text{if } (H_i < H_{\min}^k \text{ OR } H_i > H_{\max}^k) \text{ OR } (S_i < S_{\min}^k \text{ OR } S_i > S_{\max}^k) \text{ OR } (L_i < L_{\min}^k \text{ OR } L_i > L_{\max}^k), \text{ for } k > 1 \end{cases} \quad (9)$$

The red color is situated in both ends of the HUE range (Figure 10). Due to this reason for (Cnt^1) color with index 1 (Red), the increment is complete when the value of the current pixel is excluded from the defined HUE threshold.

Let's check the following example (Table 4).

Input region of interest (ROI)	Color for analysis	Output preview color in purple	Percent or number of pixels (in purple preview) in ROI
	BLACK or very dark colors		53.40%
	RED		0 numbers
	YELLOW or AMBER		0 numbers
	GREEN		0 numbers
	BLUE		502 numbers

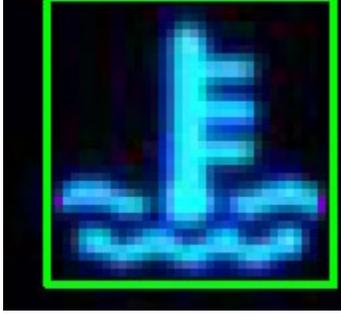
	GREY or not so pure colors		3 numbers
	WHITE or very light colors		0 numbers

Table 4: Example analysis of color in a given ROI

The percentage of the pixels (in purple preview) is 53.40 for the background color. If this percentage is more than 99, we can accept that the object is not shown and the color is black or the state is OFF. Because the state is not OFF, analysis for the defined foreground colors can be done. The greatest number of the pixels is for the blue colors. The conclusion is that the main color of the object under test is BLUE.

3. Integration into the Vision Builder for automated inspection

Vision Builder AI is application software that helps you rapidly develop and deploy machine-vision inspection systems. Using menu-driven development tools, you can configure cameras, customize image processing from hundreds of algorithms and inspection steps, interface with automation hardware and generate inspection results. You can use the menu-driven development tools in Vision Builder AI to connect and configure your camera, configure the image processing and analysis you need and benchmark your inspection to ensure it meets your test requirements. Vision Builder AI makes building production-ready inspection systems faster and easier. Configurable inspection tools that don't require programming let you quickly process images to gain the insight you need, streamline your production and save inspection time and money [3].

The place of Vision Builder AI in an automated vision test bench using PROVEtech in a validation framework can be seen in Figure 13.

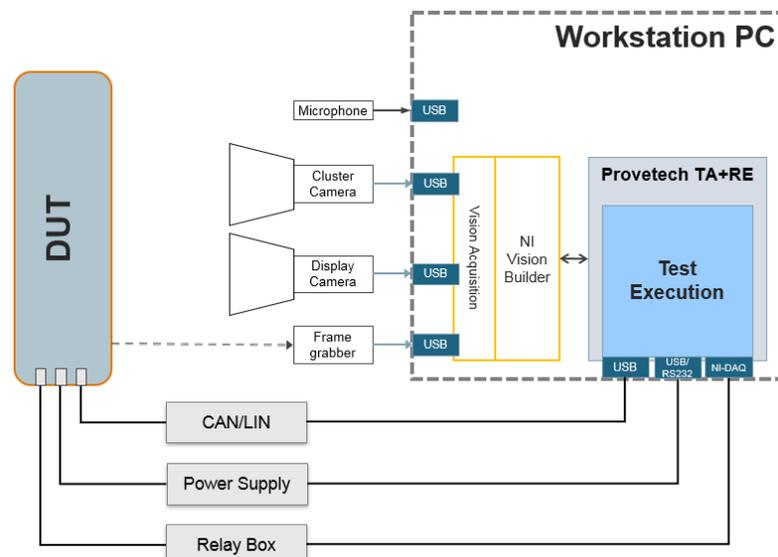


Figure 13. NI Vision Builder in the automated vision test bench

Two states are used in NI Vision Builder: “1st CLR” and “COLOR” (Figure 14.).

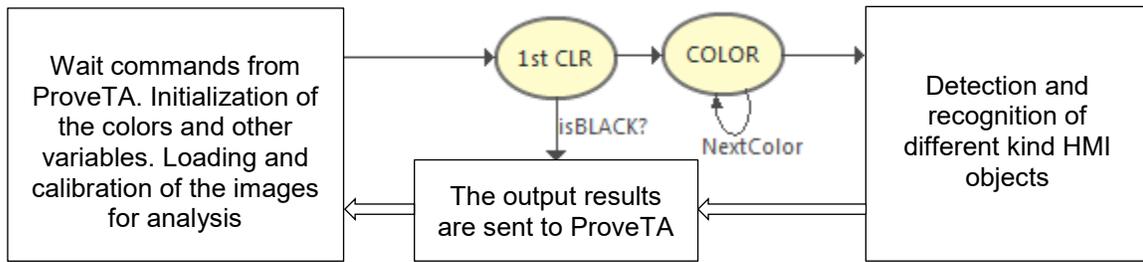


Figure 14. States for detection and recognition of color implemented in NI Vision Builder

The colors variables for threshold values should be initialized in six array variables in the vision builder inspection before the check. The names of colors for check are set in a string array.

The following inspection steps are set in the first state “1st CLR”:

- The first step is region of interest (ROI), because in the next inspection steps, we want to focus on the analyses of specific areas in the image. The ROI step is based on real-world units. This requires the image to be previously calibrated. For detection of color and states (ON/OFF) of telltales, the ROI is a rotated rectangle [4]. It is defined by the coordinates of a center point and rotated by a specified angle. The default value of the angle is zero. However, for some specific requirements and cases, the angle can be set to value greater than zero in range (0; 359].
- Two inspection steps “Count Pixels” are set [5]. The step calculates the percentage and count of pixels in a region where pixel HSL values fall within a specified threshold range. The steps are linked with the region of interest in the previously step. The first step “Count Pixels” gets Cnt^0 and uses for black or very dark color $Color^0$. It uses the threshold values $H_{min}^0, H_{max}^0, S_{min}^0, S_{max}^0, L_{min}^0, L_{max}^0$. The second step, “Count Pixels,” gets Cnt^1 and uses the threshold values: $H_{min}^1, H_{max}^1, S_{min}^1, S_{max}^1, L_{min}^1, L_{max}^1$ for the red colors $Color^1$ but with excluded interval for Hue.
- We calculate whether the pixels for the red colors are zero. If the number of the pixels is zero, then for now we accept that the color is “Other”. This is done in the step “Calculator”. With this step arithmetic, Boolean and string operations can be performed based on the outputs of the previous steps for creation of a new result [6].
- The index of the next color and Cnt^1 is transferred as variables to the “COLOR” state with the step “Set Variable”. With this step, the variables in an inspection can be updated [7]. The variable for “ $Color_{max}$ ” is initialized with $Color^1$ or “Other”.

The transition “isBLACK?” is with priority. It is executed when the percentage of black or very dark pixels of the first “Count Pixels” (Cnt^0) is more than 99% in the ROI. Then, the result in the ROI is set to OFF and sent to ProveTA.

The following inspection steps are set in the second state “COLOR”:

- In a step “Calculator” based on the current color index threshold values are set
- The result values from the previous step, “Calculator,” are set in step “Count Pixels”. It gets Cnt^k .
- Second “Calculator” calculates whether the current scanned color is the greatest color: $Color_{max}$.
- The index k of a next color is incremented in the last step from type “Set Variable”.

When all colors are scanned, result in the ROI is set to ON with a string variable for the name of the greatest color: $Color_{max}$. They are sent to ProveTA.

4. Conclusion and Impact of Your Research

- The innovative solution that we have integrated into NI Vision Builder in our automated vision test bench makes the detection of telltales, intensity level and colors efficient.
- It is integrated as a common method before detection or recognition of a concrete HMI object. By this way, it can be used not only for telltales but also and for detection of the colors of other kinds HMI objects.
- The common execution is quicker if a background color is available only in a region of interest.
- HSL color space is more natural to the human perception and understanding of color. And if a new color appears as a requirement, threshold values can be corrected easily, specified or added to the HSL arrays.

5. References

- [1] Keith Lane, **Automotive A-Z: Lane's complete dictionary of automotive terms**
<https://books.google.bg/books?id=xmDAX9JIzAIC&printsec=frontcover&dq=inauthor:%22Keith+Lane%22&hl=en&sa=X&ved=0ahUKEwiwhIPSss7nAhVjlosKHTduBdIQ6AEILzAB#v=onepage&q&f=false>
- [2] Chris Solomon | Toby Breckon, **Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab**
<https://books.google.bg/books?id=NoJ15jLdy7YC&printsec=frontcover&dq=Principles+of+Digital+Image+Processing:+Core+Algorithms&hl=en&sa=X&ved=0ahUKEwjric2uzc7nAhUPlosKHVT7CR8Q6AEITzAE#v=onepage&q&f=false>
- [3] **NI Vision Builder for Automated Inspection**
<https://www.ni.com/pl-pl/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-vision-builder-for-automated-inspection.html>
- [4] **NI Vision Builder for Automated Inspection: Region of Interest**
<https://documentation.help/NI-Vision-Builder/ROIs.html>
- [5] **NI Vision Builder for Automated Inspection: Count Pixels**
https://documentation.help/NI-Vision-Builder/CountPixels_HowTo.html
- [6] **NI Vision Builder for Automated Inspection: Calculator**
https://documentation.help/NI-Vision-Builder/Calculator_Concepts.html
- [7] **NI Vision Builder for Automated Inspection: Set Variable**
https://documentation.help/NI-Vision-Builder/GlobalVariable_HowTo.html