

ECAM BRUSSELS ENGINEERING SCHOOL

TI4L-L2: IMAGE PROCESSING

Cheat the Ishihara test

Student :

IRANKUNDA Régis [195311]

Professor :

DELHAYE Quentin

Table des matières

1	Introduction	2
2	Description of the context	2
3	Key challenges of the problem	2
4	Strategy for the key problem	2
4.1	Highlight the colours	2
4.2	Black and white	3
4.3	Translate the number	3
4.4	On the camera	4
5	Result and discussions	4
5.1	Highlight the colours	4
5.2	Black and white	5
5.3	Translate the number	5
5.4	On the camera	5
6	Conclusion	5
7	About the AI	5
8	Appendix	7
8.1	Appendix A : Final code for the webcam	7
8.2	Appendix B : Link Github repo	7
8.3	Appendix C : Link vidéo	7
9	Bibliography	8

1 Introduction

As part of the computer vision course given by Mr Delhay, we have to do a project of our choice on image processing in python using the OpenCV library. I decided to do one of the projects presented by Mr Delhay : Cheating for the Ishihara test.

2 Description of the context

As mentioned in the introduction, my project will involve cheating on the Ishihara test. The purpose of this test is to detect potential colour blindness in patients. The test will consist of coloured plates in shades of red and green with a 'hidden' number (see fig 1). A person with no visual difficulties will have no problem recognising the number, whereas a person suffering from colour blindness will.

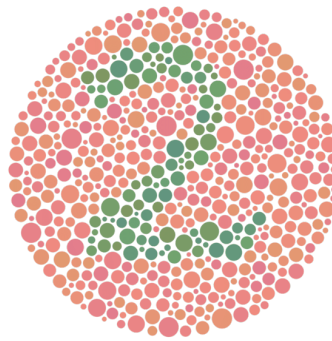


FIGURE 1 – Ishihara plate

3 Key challenges of the problem

It's an interesting project because there are several problems to solve from a computer vision point of view. Firstly, the computer has to be able to read an image that it understands, quantify it and process it. Once this image has been processed and is ready to be decoded, we need to find a technique for recognising the digits. Once we've managed to analyse the image, we'll have to do it from the computer's camera.

4 Strategy for the key problem

4.1 Highlight the colours

As we said earlier, Ishihara's figures are made up of two shades of colour : green and red. So we're going to start by dividing the figure into two images (one with the green colours and the other with the reds). This will give us a more accurate initial view of the figure. To do this, we need to create colour masks. These masks, when properly sized, allow us to keep only the colours we want. With OpenCV, it is much easier to create these masks in HSV (hue, saturation, value) space rather than BGR (blue, green, red). That's why, before creating and applying these masks, we're going to change the space domain. On the figure 2, You can find the code used to create these two masks and highlight the colours.

```

lower_red1 = np.array([0, 50, 50])
higher_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 50, 50])
higher_red2 = np.array([180, 255, 255])

lower_green = np.array([66, 9, 63])
higher_green = np.array([66, 24, 145])

# Colors masks
#mask_blue = cv2.inRange(hsv, lower_blue, higher_blue)
mask_red1 = cv2.inRange(hsv, lower_red1, higher_red1)
mask_red2 = cv2.inRange(hsv, lower_red2, higher_red2)
mask_red = cv2.bitwise_or(mask_red1, mask_red2)
mask_green = cv2.inRange(hsv, lower_green, higher_green)

# Image in the BRG channel
#img_blue = cv2.bitwise_and(img, img, mask=mask_blue)
img_red = cv2.bitwise_and(img, img, mask=mask_red)
img_green = cv2.bitwise_and(img, img, mask=mask_green)

# Result
while (True):
    #cv2.imshow('Ishihara plate in blue', img_blue)
    cv2.imshow('Ishihara plate in red', img_red)
    cv2.imshow('Ishihara plate in green', img_green)
    if cv2.waitKey(1)==27:
        break
cv2.destroyAllWindows()

```

FIGURE 2 – Code for the colours highlighting

In contrast to green, we have created two masks for red, due to the fact that in colour space, colours are represented in circular form (see fig 3) and so the 0° point corresponds to the 360° point.

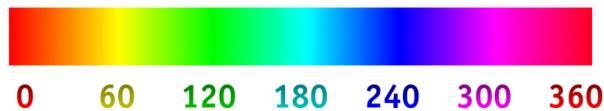


FIGURE 3 – Hue

4.2 Black and white

With the previous step, we have theoretically separated the number from the background, so normally a colour-blind person will be able to recognise the image, but this is not yet the case for the computer, which sees lots of values. That's why we're going to convert our plate to black and white to make the image binary. It's very simple and has already been done : just display the red or green mask and we'll have a black and white plate. You can find the code on the figure 4.

```

cv2.imshow('test', mask_red)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

FIGURE 4 – Code for the binary plate

4.3 Translate the number

Now that the image is more or less clear, we need to translate the figure. To do this, we use the Pytesseract tool. And with the right configuration, you can get the figure very quickly. You can find the code in the following figure :

```
import pytesseract

custom_config = r'--psm 8 --oem 0 -c tesseract_char_whitelist=0123456789' #config find with chatgpt
result = pytesseract.image_to_string(mask_red, config=custom_config)
print('Chiffre identifié:', result)
```

FIGURE 5 – Code for the translation

4.4 On the camera

The last step involves performing the four previous steps on our webcam feed. You can find the code in the appendix.

5 Result and discussions

To analyse the results, we will work on the Ishihara plate shown in figure 6.

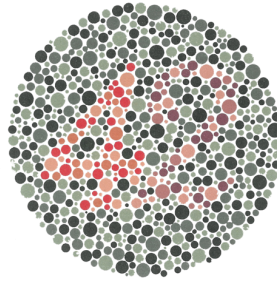


FIGURE 6 – Ishihara plate for test

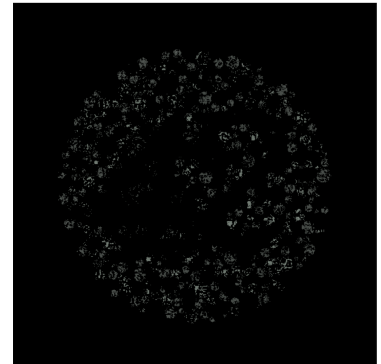
5.1 Highlight the colours

On the figure 7, you can find the 2 boards in their respective colours. The first observation is that the red mask is much more effective and precise than the green mask. I had a few problems sizing the green mask. I used trial-and-error values for HSV. These trial-and-error values were found via the paint colour dropper tool.

I limited myself to this final result because the red mask was very effective and you only need one of the two masks for the rest of the project.



(a) Part red of the plate



(b) Part green of the plate

FIGURE 7 – Result of the colours highlighting

5.2 Black and white

On the figure 8, we can clearly see our black and white board with the clear appearance of the numbers four and two.

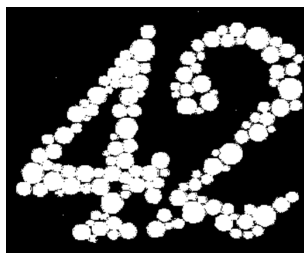


FIGURE 8 – Result of the test in black and white

5.3 Translate the number

As I said earlier, we use the pytesseract tool to translate the number. When I first tried it out, it didn't work. It was the wrong characters that were being read. I had to make a special configuration (see fig 5 to make sure that the image is analysed correctly and that only figures are output. This worked as you can see in Figure 9.

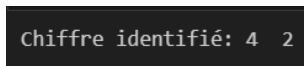
A dark rectangular box with a light-colored border. Inside, the text "Chiffre identifié: 4 2" is written in a light-colored, monospaced font.

FIGURE 9 – Result of translation

5.4 On the camera

You can watch a demonstration of the test in the video attached to the report. Everything works fine except for digit detection. The program detects the digits four and two, but it also detects other digits. These other figures may be due to noise around the Ishihara figure. Unlike the png file, the webcam stream contains much more noise.

6 Conclusion

I have a programme that works, but there's always room for improvement. For example, as we said in the result section, the green mask is not optimal and this is not a problem for Ishihara tests, but if we wanted to do this kind of test with blue and green, this programme would become obsolete. The second point where we can improve the code is in the last stage of translation. For the translation to work, I had to put in a very specific configuration that only works for reading numbers. If I want the computer to be able to read any characters, I need to improve the image in figure 8 by applying filters and making the figure more 'continuous'.

7 About the AI

During this project, I used chatGPT twice to generate lines of code. Each time this was indicated in the code as a comment next to the line in question. The first time I used AI was to size the red colour mask because it was easier than starting to look for the values myself. Obviously I didn't just copy the line without checking. I tested the mask with different images that had different shades of red to ensure that the

mask was optimal. The mask generated by chatgpt was even better than the one I sized myself, the green mask.

The second time I used AI was to configure the cipher translation. ChatGPT was a great help in this respect, as I had initially tried to size the configuration myself but I was never getting any convincing results.

In conclusion, I don't think I've abused ChatGPT and every time I've used it, I've carried out tests myself and done research on the side to make sure that the proposed solutions made sense.

8 Appendix

8.1 Appendix A : Final code for the webcam

```
import cv2
import numpy as np
import pytesseract

pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'

cam = cv2.VideoCapture(0)
while True:
    ret, frame = cam.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_red1 = np.array([10, 50, 50]) #valeur array find via chatgpt
    higher_red1 = np.array([10, 255, 255]) #valeur array find via chatgpt
    lower_red2 = np.array([170, 50, 50]) #valeur array find via chatgpt
    higher_red2 = np.array([180, 255, 255]) #valeur array find via chatgpt
    mask_red1 = cv2.inRange(hsv, lower_red1, higher_red1)
    mask_red2 = cv2.inRange(hsv, lower_red2, higher_red2)
    mask_red = cv2.bitwise_or(mask_red1, mask_red2)
    cam_red = cv2.bitwise_and(frame, frame, mask=mask_red)

    custom_config = r'--psm 8 --oem 0 -c tessedit char whitelist=0123456789' #config find with chatgpt
    result = pytesseract.image_to_string(mask_red, config=custom_config)
    print('chiffre identifi : ', result)
    cv2.imshow('Original', frame)
    cv2.imshow('frame red', cam_red)
    cv2.imshow('black and white', mask_red)
    if cv2.waitKey(1) <= 27:
        break
cam.release()
cv2.destroyAllWindows()
```

8.2 Appendix B : Link Github repo

<https://github.com/rirankunda/IT4L-Image-processing—Final-project>

8.3 Appendix C : Link vid o

https://ecambxl-my.sharepoint.com/personal/195311_ecam_be/_layouts/15/stream.aspx?id=%2Fpersonal%2F195311%5Fecam%5Fbe%2FDocuments%2Fvid%C3%A9o%5Fprojet%5Fishihara%2Emkv&referrer=StreamWebApp%2EWeb&referrerScenario=AddressBarCopied%2Eview

9 Bibliography

- https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html
- <https://medium.com/neurosapiens/segmentation-and-classification-with-hsv-8f2406c62b39>
- <https://realpython.com/python-opencv-color-spaces/>
- <https://github.com/parastuffs/TI4L-L1-image-processing>
- <https://github.com/tesseract-ocr/tessdata/tree/main>
- https://www.youtube.com/watch?v=6DjFscX4I_c&t=452s