Mathable Score Automatic Calculator

Anamaria Hodivoianu

December 3, 2024

1 First task

The first task was to locate the position of the piece that has been added. Before solving this task, I decided to extract the game board (only the part where pieces can be placed).

1.1 Extracting the game board

First, I eliminated the table on which the board is placed. I used HSV in order to obtain a mask that separates the table from the board (see Figure 1). I then used the findCountours() function from OpenCV to find the contours in the image, and then looked for the contour with the biggest area, which is the board contour, and saved the four coordinates. I then extracted the board using these coordinates (see Figure 2). The next step was to eliminate the blue region of the board and keep only the relevant game board. I estimated the coordinates as percentages of the full board, and used these to extract the game board. I also resized them to some chosen width and height. Now that I could extract only the relevant game board, I did this for all the images. Since I knew the board layout, I also knew how many lines, columns and squares there are, and I knew exactly the position of each square in the image (between which pixels it is).

1.2 Locating the piece position

In order to locate where a piece was placed, I simply subtracted each two consecutive images and looked for the square with the biggest difference. For this I used the absdiff() function from OpenCV. I also made sure that the position is a valid one (meaning a piece can only be placed next to two other consecutive pieces or the start positions).

2 Second task

The second task was to determine the number on the newly placed piece. I used template matching for this task.

2.1 Extracting the templates

I extracted the templates with the numbers by hand, using the auxiliary images. I made sure that the templates I cropped were all the same size. I then preprocessed them: resized them to 150x150 pixels (I chose this number because the squares are 200x200 pixels and it was necessary for the

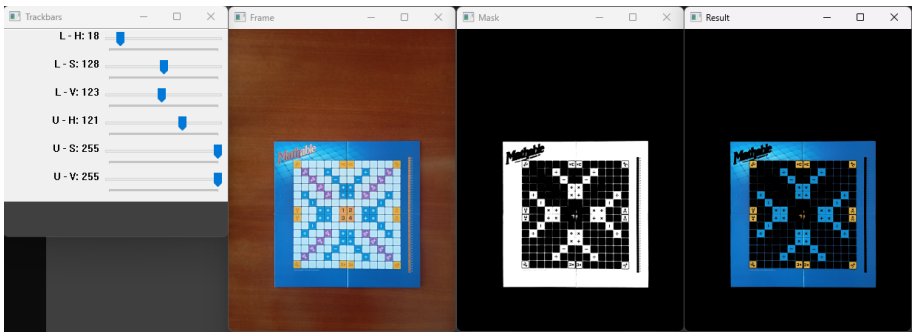

Figure 1: HSV

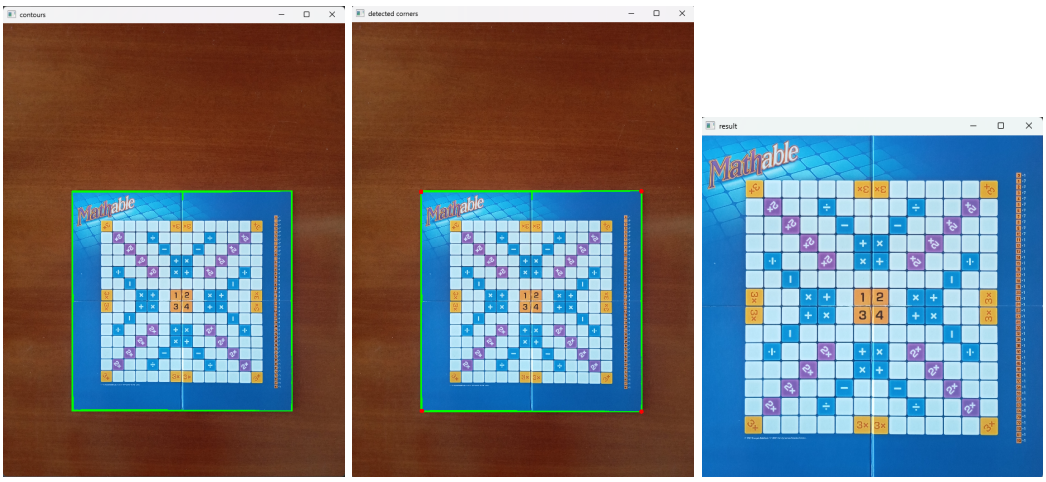

Figure 2: Contours, Corners and Result

Figure 3: Template gray, sharpened and thresholded

numbers on the templates and on the patches to be the same size), made them gray, applied a median blur, a Gaussian blur and then a threshold (see Figure 3).

```
def preprocess_image(img):
    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    img_m_blur = cv.medianBlur(img_gray, 5)
    img_g_blur = cv.GaussianBlur(img_m_blur, (0, 0), 5)
    img_sharpened = cv.addWeighted(img_m_blur, 1.2, img_g_blur, -0.8, 0)
    _, thresh = cv.threshold(img_sharpened, 30, 255, cv.THRESH_BINARY)
    kernel = np.ones((5, 5), np.uint8)
    thresh = cv.erode(thresh, kernel)
    return thresh
```

2.2 Template matching

After identifying the location of the new piece, I extracted the patch with the piece and used the same processing as for the templates: gray, median blur, Gaussian blur and threshold. I then used the matchTemplate() function from OpenCV in order to identify the template with the highest correlation. The number on that template is most likely the number on the piece.

```
def classify_number(patch):
    patch = preprocess_image(patch)
    max_corr = -np.inf
    index = -1
    for j in range(0, 46):
        img_template = cv.imread('templates_processed/' + str(j) + '.jpg')
        corr = cv.matchTemplate(patch, img_template, cv.TM_CCOEFF_NORMED)
        corr = np.max(corr)
        if corr > max_corr:
            max_corr = corr
            index = j
    return numbers[index]
```

3 Third task

The third task was to calculate the score after each round. To do this, I calculated the score for each move. After identifying the location and number of each new piece, I calculated how many points that move was worth. I first calculated how many equations the new number satisfied, and multiplied the number by the number of equations. I then checked if the position of the piece was a special one (x2 or x3), and multiplied the score by 2 or 3 if it was the case.