Sensors and Materials, Vol. 34, No. 7 (2022) 2821–2836
MYU Tokyo

S & M 3008

Automatic System for Detecting Student Attendance in College Classroom Based on Template Matching Method and Cellphone Storage Hanging Pocket

Weiping Lv,[1*] Lianyong Xiong,[1] Jinshan Xie,[1] and Cheng-Fu Yang[2,3**]

[1]School of Mathematics and Information Engineering, Longyan University, Fujian 364012, China
[2]Department of Chemical and Materials Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan
[3]Department of Aeronautical Engineering, Chaoyang University of Technology, Taichung 413, Taiwan

(Received March 29, 2022; accepted May 18, 2022)

Keywords: template matching method, OpenCV, image segmentation, image recognition, binarization, contour detection

Because teachers in colleges and universities generally do not know the number of students in their classes, they spend considerable time monitoring student attendance, reducing the teaching time. In this study, we investigated a new method for effectively monitoring the attendance of students in classes of colleges and universities. When students enter a classroom at our campus, they must store their cellphones in a pouch containing multiple pockets that is hung from a wall of the classroom. These cellphone storage hanging pockets (abbreviated to cellphone pockets) have become a necessary tool for students to store cellphones and not only improve the efficiency of studying in the classroom, but are also convenient for teachers to check student attendance at the start of classes. We investigated a template matching method for efficiently finding the attendance of students using cellphone pockets. We used images of cellphone pockets as the input images for template matching to find the grids of the cellphone pockets with no cellphones. This enabled teachers to use the serial numbers of the cellphone pocket grids with no cellphones to identify the absent students.

1. Introduction

An image is an entity obtained by observing objectives on various observation systems, and it can directly or indirectly act on the human eye to generate vision. In the transmission and exchange of information, an image is an important communication medium, and the color and grayscale are key factors that determine the expressiveness of an image.[1,2] In the human visual system, color is an important factor affecting visual effects. Since human eyes are much less sensitive to grayscale images than color images, for some grayscale images, such as IR images, scientific images, and magnetic resonance images, many researchers aim to transfer grayscale images to color images to enhance visual effects. These transfer techniques of image colorization

*Corresponding author: e-mail: 82008013@lyun.edu.cn
**Corresponding author: e-mail: cfyang@nuk.edu.tw
https://doi.org/10.18494/SAM3922

ISSN 0914-4935
https://myukk.org/

currently rely strongly on human interaction.[3,4] To convert a grayscale image into a color image, the traditional method is to use pseudo-color processing to map each of the grayscale levels of a black-and-white image to the assigned color.[5–7] That is, by establishing a one-to-one correspondence between a grayscale image and a color image, the grayscale image can be automatically converted into a color image. The advantages of this method are that it is fast and simple, and it does not change the information content of the processed image. However, the colorization effect is blunt and unnatural, and the application scope is limited.

Ruderman *et al.* proposed a method to convert a set of spectral natural images into long-, medium-, and short-wavelength cone spaces. This means that the different channels in the color space are orthogonal to each other, so that the color value of each channel is changed independently.[8,9] Reinhard and Pouli proposed a color transfer algorithm based on this anti-correlation $L\alpha\beta$ color space, by which the color of an image can be changed, and the color-changed image can retain its original shape and match the color of the reference image.[10] Welsh *et al.* subsequently proposed a method to improve this algorithm, enabling it to be used for grayscale images.[11] By this method, a color image can be processed as a reference image to colorize grayscale images and convert them into color images.

When students enter a classroom at our campus, they must store their cellphones in a pouch containing multiple pockets called a cellphone storage hanging pocket (cellphone pocket hereafter) that is hung from a wall of the classroom. An algorithm based on the algorithms mentioned above was proposed for the automatic detection of student attendance in the classrooms of a college or university by judging whether the cellphone pocket grids were empty or not from input images of the pocket. The proposed method was based on the template matching method and it was mainly used to compare the similarity between template images and actual images. The template matching method is commonly used with different sensing technologies. For example, Thanh *et al.* presented an enhanced template matching method that combined both orientation and spatial information in an effective and simple way, which was used to detect cars, humans, and maple leaves from images.[12] Xiao *et al.* presented an integrated method to analyze videos to monitor structural health and to enhance the accuracy of the vision-based structural responses. This method used template matching algorithms with a subpixel method.[13] Han proposed a template matching vision sensor system that easily detected different types of objects without prior training and also increased the reliability of template matching.[14] Chen and Zhang used a feature-weighted template matching method to design a banknote character identification algorithm.[15] In practical applications, the template matching method is simple to operate and can ensure a high recognition rate. The novelty of this paper is that we used the template matching method and took images of cellphone pockets in college and university classrooms as input images to match template images. Then, a system for the automatic detection of student attendance in college and university classrooms was investigated. Because the investigated system senses the cellphone numbers in the grids of the cellphone pocket, in this paper, we report an application of a smart sensing system and related technologies.

2. Processing in Automatic Detection of Attendance

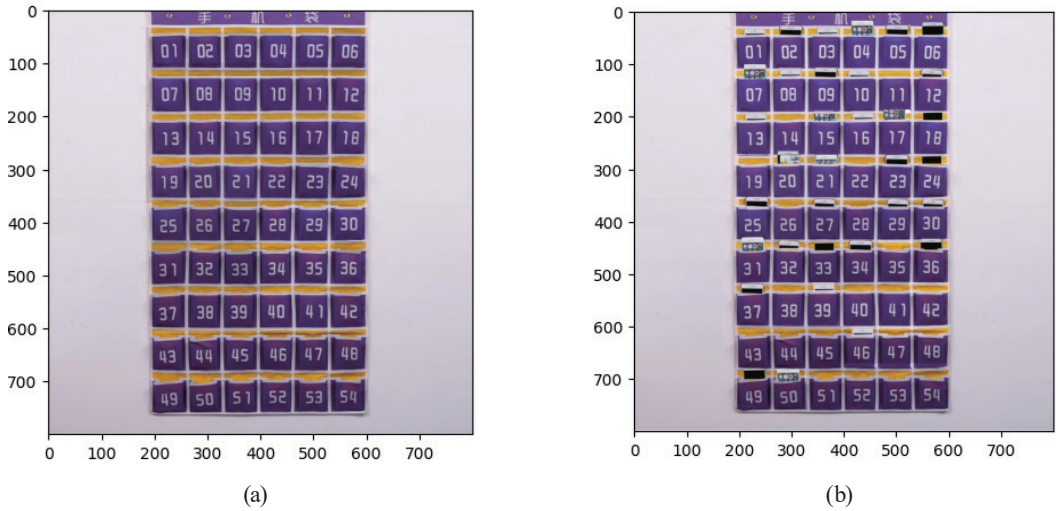
2.1 Pretreatment of cellphone pockets

2.1.1 Collection of images of cellphone pockets

There are many methods of obtaining the images of cellphone pockets, the most direct being to use a cellphone to take photographs or to download images from websites that sell cellphone pockets. To reduce the difficulty of preprocessing and obtain better template images, we used images of cellphone pockets downloaded from websites, thus avoiding the uneven illumination resulting from photographing cellphones. In this study, we assume that the maximum number of students in a class is 54; therefore, a cellphone pocket with 54 grids was chosen. Note that the collected images of cellphone pockets needed to be divided into a template image and an input image. We made a template for each grid of the cellphone pocket. To achieve a high accuracy rate after template matching, a score of least than 0.7 after template matching was necessary for the input images of the grids of the cellphone pocket for them to be stored in the cellphone. Owing to the uniqueness of the template image, it must be ensured that the input images and template images are of the same type of cellphone pocket to improve the success rate of template matching. The template image and input image of the cellphone pocket are respectively shown in Figs. 1(a) and 1(b). We found that these images meet the above requirements.

2.1.2 Grayscale processing

Grayscale images can characterize most of the features of RGB (red, green, blue) images with less data than that for images that do not use grayscale processing. Common grayscale



(a) (b)

Fig. 1. (Color online) (a) Template image and (b) input image.

transformation methods include the maximum grayscale method, average grayscale method, weighted average grayscale method, and Open Source Computer Vision Library (OpenCV) grayscale processing method. The maximum grayscale method uses the maximum value of the R, G, and B components of each pixel in a color image as the grayscale values of the image:

$$f(x, y) = \max(\mathrm{R}(x, y), \mathrm{G}(x, y), \mathrm{B}(x, y)). \tag{1}$$

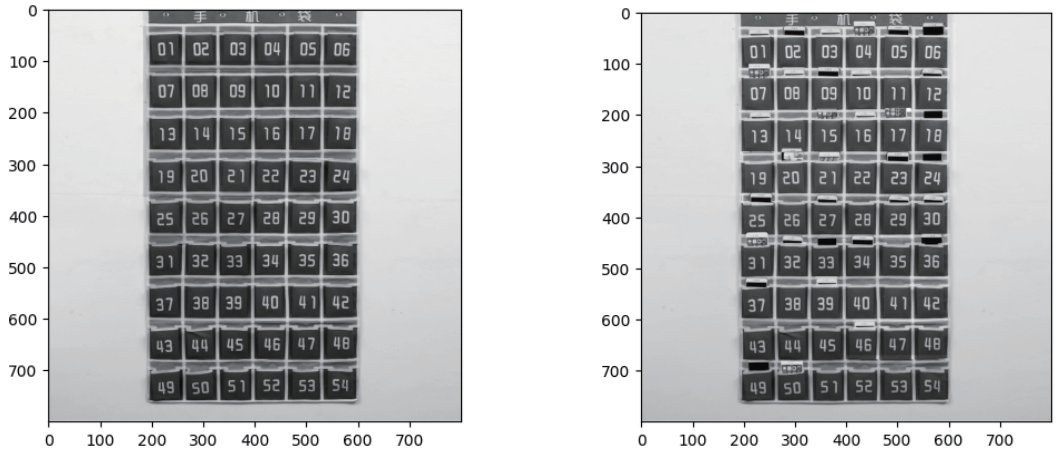
The average grayscale method uses the average value of the R, G, and B components of each pixel in a color image as the grayscale values of the image:

$$f(x, y) = (\mathrm{R}(x, y), \mathrm{G}(x, y), \mathrm{B}(x, y))/3. \tag{2}$$

The weighted average grayscale method assigns different weights to the R, G, and B components of each pixel in a color image to obtain more suitable grayscale values of the image. Since the human eye is most sensitive to green and least sensitive to blue, the G component has the highest weight and the B component has the lowest weight. The formula used is

$$f(x, y) = 0.30\mathrm{R}(x, y) + 0.59\mathrm{G}(x, y) + 0.11\mathrm{B}(x, y). \tag{3}$$

When OpenCV is used for grayscale processing and the imread() method is used to transfer an image, since the images read by OpenCV are in RGB format, it is necessary to convert the images into RGB format and then perform a grayscale transformation. The grayscale processing by OpenCV can generally add the parameter zero to imread() in the read images. It can also use cvtColor to perform grayscale processing to obtain a grayscale image. Figures 2(a) and 2(b) show the template matching image (denoted as LA) and input image LA, respectively.



(a) (b)

Fig. 2. (Color online) (a) Template grayscale image LA and (b) input grayscale image LA.

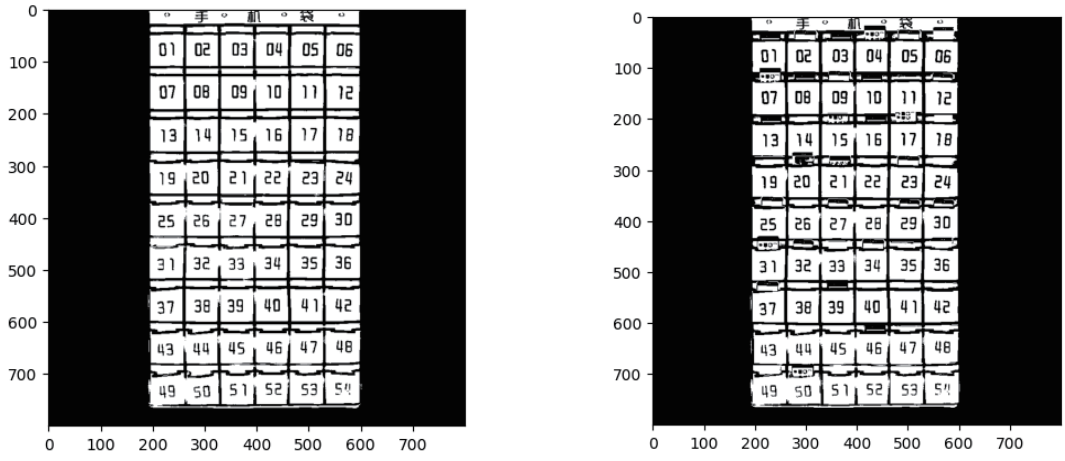
2.1.3 Image binarization

Image binarization is the technology of converting a grayscale image into a black-and-white image. This process essentially reduces the amount of information contained within an image from 256 shades of gray to two values, black and white, corresponding to a binary image. After the image is binarized, there are only two clean values in the entire image, making it very suitable for contour detection in this study. After calculating the contour, graphic processing can be performed, such as drawing the contour of the image, enhancing the sharpness of the edge of an image, emphasizing the contour of an image, and adding shadows to an image according to the contour. Therefore, this is a very important technology, and its operation with other functions can achieve many effects.

To further extract the reverse control area of the cellphone pocket from the image, the maximum class difference method (Otsu method) was used to binarize the grayscale image LA to obtain the binary image LB.[16] This is because the Otsu method can traverse all possible threshold values to obtain the best threshold values. However, this method is sensitive to noise due to the uneven illumination of images. Such noise distorts or deforms images of the cellphone pocket and affects the extraction of image data. The top-hat reconstruction method is a nonlinear filter based on a morphological operation that can perform noise suppression, feature extraction, and image segmentation.[17] Top-hat reconstruction can eliminate the above shortcomings of the Otsu method; thus, it is necessary to carry out top-hat reconstruction of the grayscale image LA and binary image LB. The transformation formula used in top-hat reconstruction is

$$\text{Top\_hat}(A) = A - \text{R}_A(A\theta B), \tag{4}$$

where $\text{R}_A(A\theta B)$ represents the result of the open operation. In addition, the gaps between the cellphone pocket grids must be used as boundaries in the following investigation; thus, the obtained LB image must also undergo a color-flipping step for subsequent segmentation. Figures 3(a) and 3(b) show the grayscale image LA and binary image LB, respectively.



(a) (b)

Fig. 3. (a) Grayscale image LA and (b) binary image LB.

2.1.4 Contour detection

The next step is to perform feature extraction on the binary image LB to identify the area of the cellphone pocket. Generally speaking, there are other foregrounds around the cellphone pocket, which will affect the feature extraction of the cellphone pocket. In addition, every grid of the cellphone pocket is connected to the others. To recognize the cellphone pocket image as a region, morphological operations must be performed on the image, such as a closing operation, expansion, and maximum filtering. By adjusting the relevant parameters, we can ensure that the region of the cellphone pocket becomes a single area. The basic principle of the closed operation of the morphology comprises the operations of swelling and then corrosion, which help to close small holes inside the foreground objects and remove small black spots on the objects. Figure 4 shows that black spots in the foreground are partially removed after the morphological operation, but the foreground does not become perfectly white. Because the cellphone pocket was successfully converted into a single area, no additional computing resources were necessary to make the entire region white.

The next step is performed for contour detection, where the principle is to hollow out the internal points. For example, if there are 3 × 3 rectangular points in the original image, then the point in the middle can be removed, so that the "circle of points" remains outside and forms outlines (B_1, B_3, and B_4 in Fig. 5). Therefore, the boundaries and connected domains in an image can form a topological graph.

The contour detection is used to detect the binarized edge image, and the four borders of the image constitute its frame. Generally, the frame of an image is defined by the zero pixels. The area occupied by the connected zero pixels in an image is called the zero-connected component, and the one-connected domain is obtained in the same way. If there is a zero-connected domain S containing the frame of the image, then S is called the background. The pixel in the ith row and jth column is represented by (i, j), and the pixel value is represented by $f\{i, j\}$. This image can

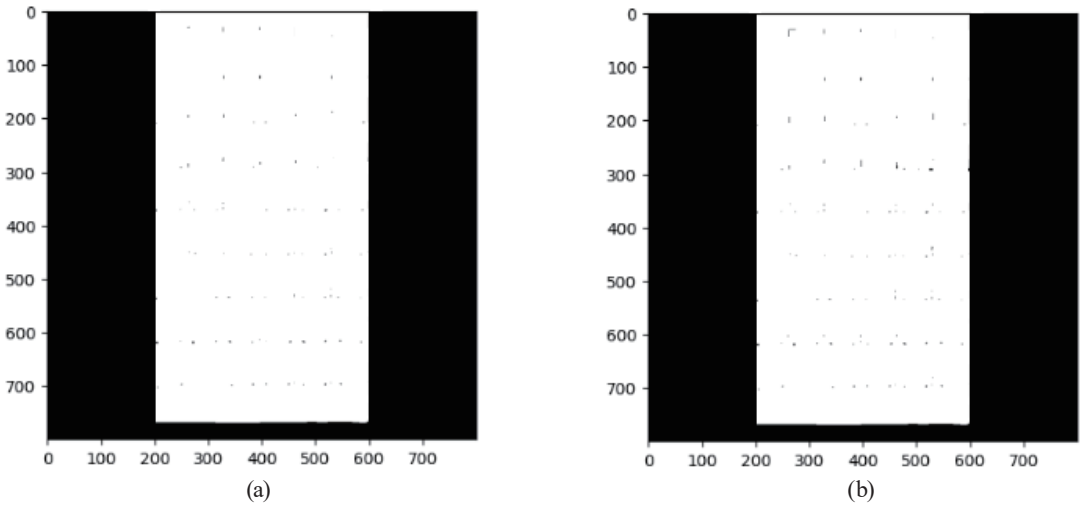


Fig. 4. (a) Template LB image after morphological operation and (b) input LB image after morphological operation.

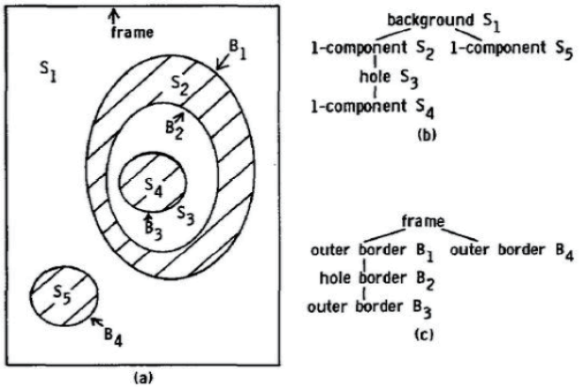


Fig. 5. Topological relationship of LB image.

then be represented by $F = \{f\{i, j\}\}$. When 4(8) is used to denote the connectivity of a pixel, if the pixel has a value of one, it means that 4(8) is connected; if the pixel has a value of zero, it means that 8(4) is connected. Next, it is necessary to find the boundary points to obtain the contour. In the case of 4(8) being connected, if a one-pixel point (i, j) has a zero-pixel point (p, q) in its eight-connected domains, then this one-pixel point is called a boundary point. The above descriptions are used to find the junction of the one-connected domain [the parent set of (i, j)] and the zero-connected domain [the parent set of (p, q)], and the one-pixel point on the boundary line is recognized as the boundary point.

By examining an image after the morphological operation, we found that only one contour exists. In OpenCV, the function cv2.findContours() is used to find the contour of an image. For further processing, we must also visualize the contour. The function cv2.drawContours() can draw contours in the target image. Because drawing a contour in the original image will change its characteristics, we use the img.copy() method to make a copy of the image and then draw the contour.

From Fig. 6, one can see that the identification of the contour is very accurate. As mentioned above, even if there are other contours, they will not affect the extraction of the contour of the cellphone pocket. To extract this contour, the cv2.boundingRect() function is first used to find the minimum rectangle of the contour. To simplify the processing of the cellphone pocket, the template LB image and input LB image are processed directly. This processing method has a similar effect to that of processing the original cellphone pocket. The cv2.boundingRect() function returns a list $[(x_1, y_1, w_1, h_1), (x_2, y_2, w_2, h_2), (x_3, y_3, w_3, h_3), \ldots, (x_i, y_i, w_i, h_i)]$, each of which is a tuple comprising the abscissa coordinate, ordinate coordinate, width, and height of the smallest rectangle of each contour.

The next step is to find the smallest rectangle of the contour of the cellphone pocket and extract it from the LB image. The basic idea of finding the smallest rectangle of the contour is to find its appropriate width and height. According to the LB image, the aspect ratio of the cellphone pocket is about 1:2, then we can define the aspect ratio as $\alpha = h_i/w_i$. We set $\alpha \in [1.8, 2.1]$ as the screening condition, and the image matrix is used to filter out the LB image of the contour of the cellphone pocket, as shown in Fig. 7.

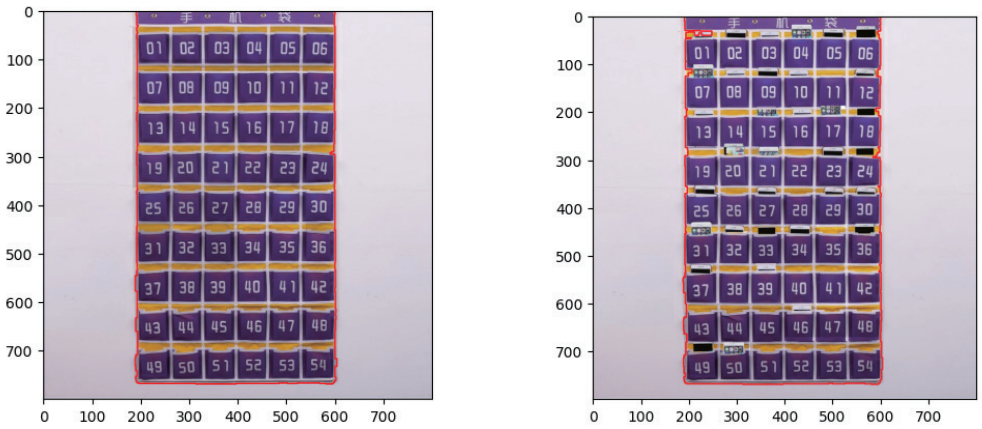


Fig. 6. (Color online) (a) Contour of the template image and (b) contour of the input image.

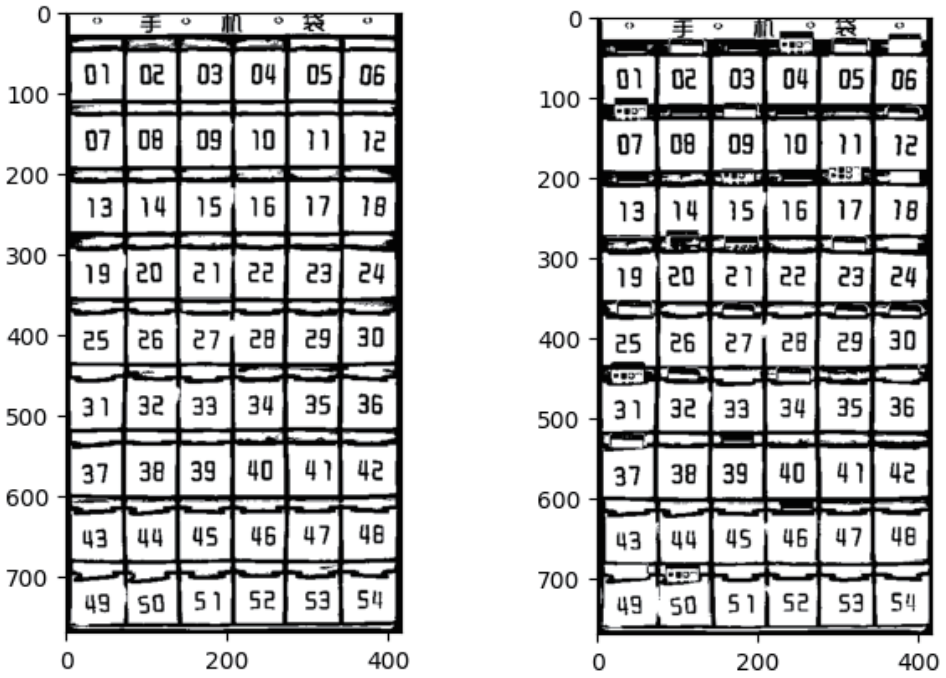


Fig. 7. (a) Template contour of the binarized grayscale image LB and (b) input contour of the binarized grayscale image LB.

3. Results and Discussion

3.1 Image segmentation

Image segmentation is the main problem in image recognition. To carry out image segmentation and obtain each cellphone pocket grid with a separately marked serial number (i.e., student number) as a template, we use bar graphs to process the binary image of the template to achieve image segmentation.

3.1.1 Statistical image with black spots in each row

In this step, the number of black spots in each line of the binarized greyscale image LB is counted. If the number of black spots in a line is much larger than that in surrounding lines, it can be seen as a dividing line of the cellphone pocket grid. In this way, we distinguish each line of the cellphone pocket more effectively than by using contour detection to extract images of the cellphone pocket grid. The key steps in counting the number of black spots in the binarized greyscale image LB are to find and record the number of black spots, draw the corresponding bar graph, and then segment each cellphone pocket grid using the peak of the bar graph. The key code is listed below:

```
Count the number of black values (0) in each row
    hd = []
    for row in range(rows):
        res = 0
        for col in range(cols):
            if threshold[row][col] == 0:
                res + = 1
        hd.append(res)
```

3.1.2 Plot black-spot bar graph

When the data in the barh() function of the matplotlib.pypolt model is called for drawing, the data in hd[ ] is taken as the target value, the coordinate point of each action of the LB image is used to plot the bar graph, and Fig. 8(a) is obtained. In Fig. 8(a), although the peaks and troughs can be distinguished, it is not easy to find the coordinates defining the boundary of the cellphone pocket grid. Therefore, the trough is subjected to a zeroing process. Figure 8(a) shows the trough in the bar graph without the zeroing process, and Fig. 8(b) shows the trough in the bar graph with the zeroing process. When hd[i] ≤ 200, we set hd[i] = 0 to obtain the results in Fig. 8(b).

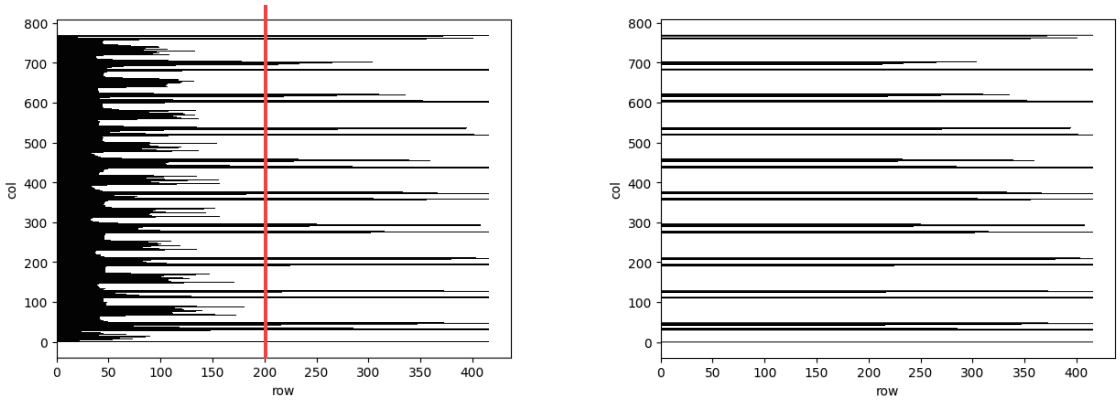
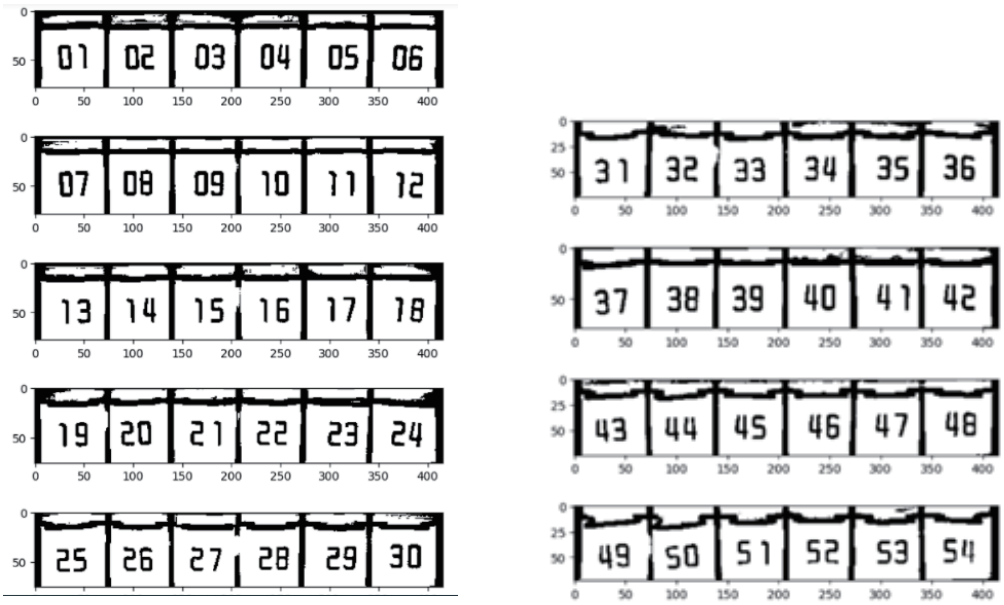


Fig. 8. (Color online) (a) Trough in the bar without zeroing process and (b) trough in the bar with zeroing process.

3.1.3 Horizontal divisions of cellphone pocket

As shown by the processed bar chart in Fig. 8(b), the black rectangles are next to each other, their heights in each row are similar, and they form a large rectangular bar. There is also a space between each large rectangular bar. These spaces can be roughly divided into two categories: very narrow spaces whose width is similar to that of a large rectangular bar (referred to as the small-space rectangular bar group) and very wide spaces with a width of about 100 pixels (referred to as the large-space rectangular bar group). By comparing Figs. 6(b) and 8(b), we can roughly analyze the information conveyed by these large rectangular bars. We found that the small-space rectangular bar group represents the entrance of each grid of the cellphone pocket, whereas the large-space rectangular bar group represents the rectangular frame where the number of the cellphone pocket grid is located. Figure 8(b) also shows some special rectangular bars, for example, a rectangular bar whose longitudinal axis close to zero is very thin and is a rectangular bar group with no small spaces. Therefore, it is neither the grid entrance of the cellphone pocket nor the rectangular frame where the number of the cellphone pocket is located.

Comparing Figs. 6(b) and 8(b), we conclude that the thin strip is the bottom frame of the cellphone pocket. In addition, there is a special small-space rectangular bar group close to the vertical coordinate of 780. Our analysis suggests that it represents the area of the text “mobile phone pocket”, which is located at the top of the cellphone pocket. The next step is to find the horizontal coordinates at which the grids of the cellphone pocket are divided, the result of which is shown in Fig. 9. The coordinates of the rectangular bars must satisfy the following: the upper side of the rectangular bar is empty (0) and the lower side is also empty (0). The two ordinates form the coordinates of a large rectangular bar, but these coordinates are not those of the segmentation images. For the segmentation images, the trough boundary of the bar can be used



(a) (b)

Fig. 9. Template image of the divided cellphone pocket., for Nos. (a) 1–30 and (b) 31–54.

as the coordinates. Note that it is necessary to filter the coordinates of the trough with width less than 50 pixels, because they cannot be the width of the cellphone pocket grid. The key codes and coordinates are as follows:

```
# find the interval that is not zero
    position = [] # record coordinates
    reg = []
    for i in range(rows-1): # traverse each row
        if hd[i] == 0 and hd[i+1]!= 0:
            reg.append(i)
        if hd[i]!= 0 and hd[i+1] == 0:
            reg.append(i)
        if len(reg) == 2:
            if (reg[1] - reg[0]) >= 50: # Filter the coordinates whose trough width is less than 50
pixels
                position.append(reg)
                reg = []
            else:
                reg = []
Boundary coordinates of the troughs: [[49, 109], [129, 190], [212, 272], [296, 354], [377, 435],
[459, 516], [538, 599], [622, 679], [703, 758]]
```

Next, we obtain the image segmentation coordinates of the trough boundary coordinates from the image LB of the template cellphone pocket shown in Fig. 6(b). Then, the cellphone pocket is divided horizontally with the dividing coordinate points as the boundary, and nine lines of grids are obtained from the cellphone pocket. This also ensures the complete interception of the entrance image of the grids on the cellphone pocket. The key code is described below:

```
# split into nine lines
Group = [] # Store the coordinates of each row of grids on the cellphone pocket
for i in position:
    group = threshold[i[0]-18:i[1],:] # Coordinate area shrinks
    Group.append(group)
    plt.imshow(group,cmap = 'gray')
plt.show()
```

3.1.4 Statistical image with black spots in each column

When the grids of the nine rows of the cellphone pocket are traversed, we count the black spots in each column. The key point is that we use the coordinates of each column to traverse the black spots of each row and count the records. Because the process is similar to that of statistical imaging for each line, little analysis is required in this step.

3.1.5 Plot black-spot bar charts

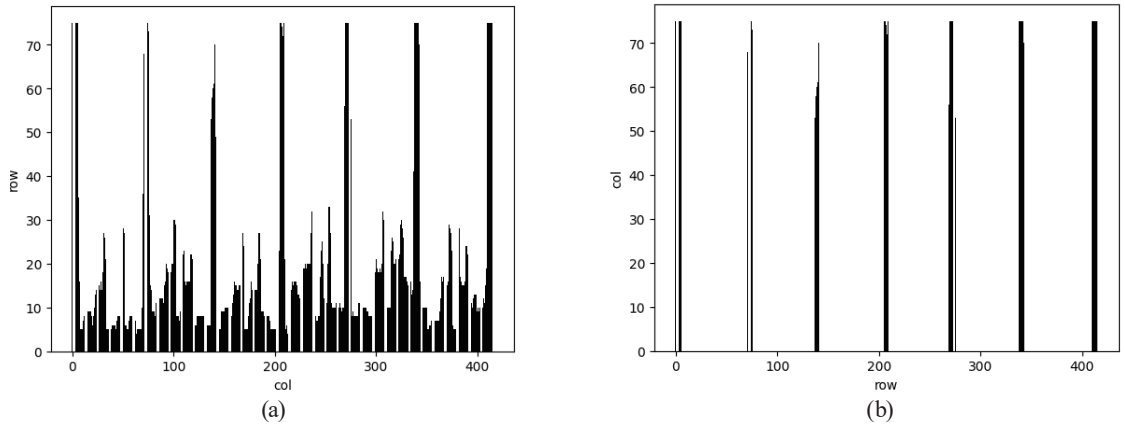
Next, we call the barh() function in the matplotlib.pypolt module to plot the black-spot bars. The method is similar to that in Sect. 3.1.2, and we obtain the bar with the trough not subjected to zeroing processing, as shown in Fig. 10(a). From Fig. 10(a), we find that although the peak and the trough can be distinguished, it is difficult to find the coordinates of the dividing lines of the cellphone pocket grids. Therefore, the trough is reset to zero. As shown in Fig. 10(b), the result of processing is a group of seven small-space rectangular bars. Comparing this result with Fig. 9, it can be seen that the large rectangular bars represent the division lines of each cellphone pocket grid.

3.1.6 Longitudinal divisions of cellphone pocket

As shown in Fig. 10(b), if each rectangular bar satisfies the two coordinates that are empty on the left and empty on the right, then the edge coordinates of binarized greyscale image LB are obtained, and they are recorded as the coordinates of division points. When the binarized greyscale image LB is used to divide the cellphone pocket horizontally with the dividing coordinates as the boundary, then we obtain six-column grids in the cellphone pocket (the process is similar to that in Sect. 3.1.3). Next, we traverse the cellphone pocket grids on each row to perform the above operations, and 54 trough boundary coordinates are obtained. The trough boundary coordinates are defined as the image segmentation coordinates and they are saved in the list "position", and the cellphone pocket is divided horizontally with the segmentation coordinate points as the boundary line. Some of the results are displayed in Fig. 11.

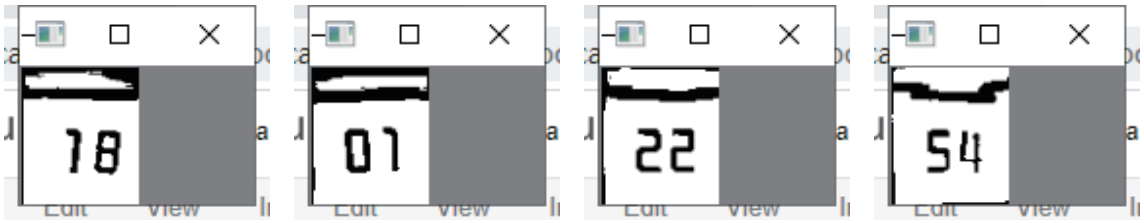
3.2 Save templates

Next, the obtained 54 grids of the cellphone pocket are saved and set as the templates, where image annotation is a key part of setting the template. Here, the method of loop traversal is used to realize this function, and the order is defined as



(a) (b)

Fig. 10. (a) Trough for the bar without zeroing processing and (b) trough for the bar with zeroing processing.



(a) (b) (c) (d)

Fig. 11. (Color online) Screenshots of some results. For Nos. (a) 18, (b) 01, (c) 22, and (d) 54.

$$\text{order} = i(\text{count}(col)) + j + 1, \tag{5}$$

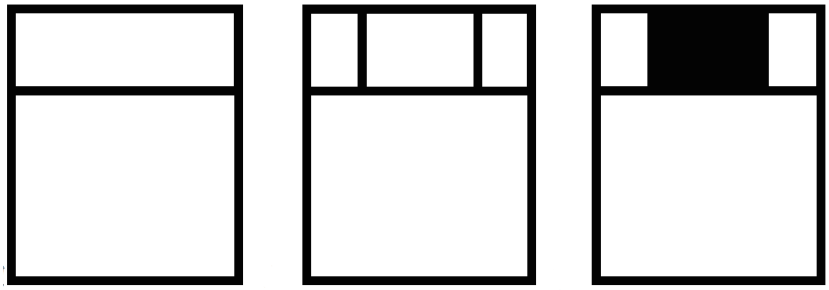
where i represents the serial number of each element in the group (i.e., the coordinate serial number of the cellphone pocket grid in each row), j represents the serial number of each element in the position (i.e., the coordinate serial number of each cellphone pocket grid), and count(*col*) indicates the number of cellphone pocket grids in each row (six in this study). The template images must also be of the same size. Here, we set the size of each template image to 60 × 70 pixels for the subsequent template matching. The key code is listed below:

```
for j, loc in enumerate(position):# loc represents image segmentation coordinate
    col_group = g[:, loc[0]:loc[1]] # all sizes are set as (60, 70)
    col_group = cv2.resize(col_group, (60,70))
    if not os.path.exists('template'):
        os.mkdir('template')
    cv_show('', col_group)
#       plt.imshow(col_group, cmap='gray')
#       plt.show()
    cv2.imwrite('template' + './img{}.jpg'.format(i × (len(position))+ (j+1)), col_group)
```

3.3 Template matching process

3.3.1 Preprocessing the input image

To finish the template matching process, the input image requires preprocessing, as described above, and image segmentation. The segmentation of the input images is basically the same as that of the template images. The principle of template matching technology is used to compare the similarity between the template image and the input image. However, the template matching process cannot be performed directly to segment the cellphone pocket grids (input image), and further processing is required. To improve the recognition accuracy, it is necessary to further process the input images of the divided grids. Figure 12(a) shows the normalized model of a template image, and Figs. 12(b) and 12(c) show the normalized models of the input images. From the images, we find that the input image in Fig. 12(b) will reduce the recognition success rate, making it necessary to convert the standardized model of Fig. 12(b) into the standardized model of Fig. 12(c) to improve the recognition accuracy. The process of code conversion is as follows:



(a) (b) (c)

Fig. 12. Comparison of the standardized models. (a) The normalized model of a template image, (b) and (c) the normalized models of the input images.

```
for r in range(rows):
for c in range(cols):
    if col_group[5:15,15:46].mean() >=100:
        if np.hstack((col_group[5:15,:10],col_group[5:15,49:-1])).mean() <= 180:
            col_group[r][c] = 0
```

3.3.2 Template matching process

The template matching process is used to find the positions of template images in a larger image. We use the basic principles of the template matching process and improve it to automatically evaluate student attendance in a college classroom based on the cellphone pocket. First, we slide the template images onto the input images and then compare the similarity between the images. Since our template image is of a single cellphone pocket grid rather than the whole cellphone pocket, the size of the template images is the same as that of the input images. Therefore, the comparison between a template image and an input image does not require an input image to be slid on a sample image; it is only necessary to directly compare the two images. We found that the similarity was very high. To reduce the similarity, we change the region of the template image and use the entrance of the cellphone pocket grid as the matching area of the template image, then the new template image is compared with the input image.

The matchTemplate method provides many different matching methods, and we use the TM_CCORR_NORMED method to calculate the normalized correlation; a normalized correlation closer to 1 suggests greater similarity between two objects. In this study, when the normalized correlation is closer to 1, it is more likely that there is no cellphone in the cellphone pocket of the input image, that is, the student is absent. Therefore, the loop traverse and matchTemplate method in OpenCV are used to achieve template matching between the template images corresponding to the cellphone pocket grid and the input images. The matching scores are stored in the SCORE list and are used as probabilities that the cellphone is not placed in the cellphone bag grids. After the matching score in the SCORE list is found, the relationship between the number of students and the number of cellphone pocket grids is considered. Under normal circumstances, absent students are a minority and the number of students in the class will not exceed 54. Therefore, we only need to find the serial number of a cellphone pocket with a high matching score and save it. When there are exactly 54 students in the class, no further

processing is required. When the number of students is less than 54, it is necessary to consider marking the excess cellphone pocket grids as empty instead of absent.

3.4 Absentee list

3.4.1 Student list

In this study, a class in a university having 50 students was used as an example to demonstrate the effectiveness of the automatic detection system for student attendance. In this class, excluding students on leave and transfer students, there are 47 students. The teacher can set the corresponding marks for the three excluded students (Nos. 43–45), so that when traversing the serial numbers (student numbers) of the cellphone pocket grids, these students are removed from the absentee list.

3.4.2 Absentee list

The SCORE list has already been obtained. This list holds the score of each cellphone pocket (the probability that the cellphone pocket cell has no cellphone). After traversing each element of the SCORE list, 0.71 is used as the judgment condition (the value can be adjusted as necessary). If an element is greater than 0.71, then one is added to its serial number, and the change is saved to the abLt list; because the serial number list starts at zero, it is necessary to add one to the serial number to ensure that the serial number of the list corresponds to that of the cellphone pocket grid. The key code is listed below:

```
# Find the numbers of absent students
abLt = []
for i, c in enumerate(SCORE):
    if c >= 0.71:
        abLt.append(i+1)
```

Through this process, the abLt list is obtained. Although one may think that the information saved in this list is the absentee list, we found that the abLt list also saves the serial numbers of students who have transferred or dropped out or are on leave. The empty elements include the transfer student and students on leave, and some of the numbers for the class are larger than 50. Therefore, we must remove these empty elements from the abLt list. The first step is to compare the elements in the abLt list with the largest student number in the class. Any elements in the abLt list that are greater than the largest student number must be removed. The second step is to directly remove the elements of the serial numbers marked above for transfer students and students on leave, after which the remaining elements in the abLt list are the absent list. The absent students found from the input images were [11, 14, 19, 22, 26, 28, 35, 38, 40, 41, 42, 47, 48]. By comparison with the student number list, we found that No. 21 is also absent. Thus, we obtained a recognition accuracy of 92.3% for the automatic detection system for student attendance.

4. Conclusions

We investigated a template matching method based on template matching of the input images of cellphone pockets in colleges and universities, so as to automatically monitor attendance in college classrooms and prevent students from missing classes. Our proposed method can also prevent the misjudgment of absences of students who have transferred school or been given permission for leave. To automatically monitor attendance, the teacher takes an image of a cellphone pocket containing cellphones as the input image. Then the uploaded database is used to find absent students and the results are returned. A class in a university having 50 students was used as an example to demonstrate the effectiveness of the automatic detection system for student attendance. Thirteen students were absent according to the automatic detection system, whereas the actual number of absent students was 14, corresponding to a high recognition accuracy of 92.3% for the automatic detection system.

Acknowledgments

This work was supported by projects under Nos. MOST 110-2622-E-390-002 and MOST 110-2221-E-390-020.

References

1 C. Kanan and G. W. Cottrell: PLoS ONE **7** (2012) e29740.
2 D. K. Seo, Y. H. Kim, Y. D. Eo, and W. Y. Park: Appl. Sci. **8** (2018) 1269.
3 V. G. Jacob and S. Gupta: 2009 16th IEEE Int. Conf. Image Processing (ICIP, 2009) 1653–1656.
4 S. Wan, Y. Xia, L. Qi, Y. H. Yang, and M. Atiquzzaman: IEEE Trans. Multimedia **22** (2020) 1756.
5 M. A. U. Khan, R. B. Khan, S. Bilal, A. Jamil, and M. A. Shah: Inf. Technol. J. **7** (2008) 210.
6 M. R. Khosravi, H. Rostami, G. R. Ahmadi, S. Mansouri, and A. Keshavarz: Int. J. Electr. Commun. Comput. Eng. **6** (2015) 324.
7 B. Selvapriya and B. Raghu: Int. J. Eng. Technol. **7** (2018) 954.
8 D. Ruderman, T. Cronin, and C. Chiao: J. Opt. Soc. Am. A **15** (1998) 2036.
9 S. Kumar and D. Singh: Int. J. Comput. Eng. Resear. **2** (2012) 1272.
10 E. Reinhard and T. Pouli: Colour Spaces for Colour Transfer, R. Schettini, S. Tominaga, and A. Trémeau, Eds., Computational Color Imaging, CCIW 2011, Lecture Notes in Computer Science, Vol. 6626 (Springer, Berlin, Heidelberg, 2011).
11 T. Welsh, M. Ashikhmin, and K. Mueller: ACM Trans. Graph. **211** (2002) 277.
12 N. D. Thanh, W. Li, and P. Ogunbona: 2009 9th Asian Conf. Computer Vision, Xi'an, Sep. 23–27 (2009) 193–202.
13 P. Xiao, Z. Y. Wu, R. Christenson, and S. Lobo-Aguilar: J. Civ. Struct. Health Monit. **10** (2020) 405.
14 Y. M. Han: Sensors **21** (2021) 8176.
15 G. B. Chen and G. C. Zhang: Microelectronics Comput. **30** (2013) 115 (in Chinese).
16 N. Otsu: IEEE Trans. Syst. Man Cybern. **9** (1979) 62.
17 J. W. Hsieh, S. H. Yu, and Y. S. Chen: J. Electron. Imaging **11** (2002) 507.