e-Yantra Robotics Competition Plus

(eYRC+ Pilot)

eYRCPlus-PS1#301

Team leader name	**Vishnu Chandrasenan**
College	**SIES Graduate School of Technology**
e-mail	**vishnu5594@gmail.com**
Date	**25-11-2015**

Image Processing (8)

Write down the answers to the following questions.

1. What is the resolution (size) of the test image assigned in the task?
2. What is the use of thresholding an image?

Answers:

1. The resolution of test_image1 is **1050 x 516 pixels** and test_image2 is **1050 x 511 pixels**.
2. Thresholding is an image segmentation technique which is used to:
    - Convert a gray scale image to binary image
    - Partition an image into a foreground and background
    - Isolate objects like text or other image data such as graphs, maps etc.
    - Be able to see what areas of an image consist of pixels whose values lie within a specified range, or *band* of intensities.

Consider the following illustrative example. We have a grayscale image of the eyes taken from a standard webcam (fig 1). This image can be thresholded to result in a binary image which contains only the required eye region, discarding all other irrelevant details (fig 2). Hence such examples of segmentation using thresholding can be used in image processing as a starting point in a series of other processing techniques.

Fig 1. Grayscale image of the eye which is the input

Fig.2 Binary thresholded output image

Explain the algorithm used to perform the task given in practice_test folder.

1. Differentiating foreground and background in the test image

The first step of the algorithm was to remove the black grid lines, thereby keeping only foreground elements (numbers) in the image. This was achieved as follows: a mask of the same size as test image was created, such that when 'bitwise or' is performed between mask and the input image, only the number part is left intact. All other unnecessary details in the image is discarded.

2. Contouring the digits in the image

The resultant image was contoured so that the digits are recognized with a green outline. The original input image with grids is now superimposed on the processed image. The grids and other details are obtained back, along with the presence of contoured numbers. Furthermore, we break the image into two grid sections: D1 and D2.

3. Number Detection in "Grid D1" by Template Matching

An array of images was created consisting of templates of numbers from 0 to 9. Thereafter, the 12 regions in D1 grid were cropped and stored in another array. At this stage, all the individual digit templates (0 to 9) were compared with individual regions in D1 (12 regions), resulting in a total 120 combinations. If a number template matches with good accuracy, a digit is detected. All the matched digits are stored in an array which is to be displayed.

4. Number Detection in "Grid D2" by Template Matching

In grid D2, there are 2 possible scenarios i.e. it can be a single digit number or a double digit number.

a)Detection for single digit scenario:

Initially 24 regions of D2 was cropped. Then the number of black pixels in every region was checked. If number of black pixels < 900, then it can be implied that the region contains a single digit. Subsequently, all regions were compared with template samples, and if match was detected, the number was stored in an array.

b)Detection for double digit scenario:

If single digit it not detected, the region was cropped into 2 halves such that one region contains the tens digit part and the other region contains the units digit part. Both the regions were respectively searched for template matches. The tens digit matches were stored in one array and the units digit part were stored in another array.

This process is explained in detail as follows:

i) Detection of tens digit:

Compare all templates (0 to 9) with all tens regions in D2 (24 regions) one by one (total 240 combinations). If a number template matches with the digit in the region with threshold > 0.9, detect match and store all numbers found in tens array.

ii) Detection of unit digit:

Compare all templates (0 to 9) with all unit regions in D2 (24 regions) one by one (total 240 combinations). If a number template matches with the digit in the region with threshold > 0.9, detect match and store all numbers found in units array.

5. Formation of 2 digit number and then display results

The array elements of tens array and units array were added such that,

- **Two digit number= 10*tens place + unit place**

The index of each region was found out and finally it was displayed along with the number. Thus the recognized number and its location in the grid square was found and displayed as output.

Software used (7)

Write down the answers to the following questions.

1. Write a function in python to open a color image and convert the image into grayscale. You are required to write a function *color_grayscale(filename,g)* which takes two arguments:
    a. filename: a color image (Test color image is in folder "Task1_Practice/test_images". Pick first image to perform the experiment.)
    b. g: an integer

Output of program should be a grayscale image if g = 1 and a color image otherwise.

Code:

```
import cv2

#Function to convert image to grayscale if g=1
def color_grayscale(filename,g):

    if g==1:
        image=cv2.cvtColor(filename,cv2.COLOR_BGR2GRAY)

    else:
        image=filename
    return(image)

img = cv2.imread('test_image1.jpg',1) # Load the color image
g=input("Enter value of g")           # accept value of g
image=color_grayscale(img,g)          # function call
cv2.imshow('Image',image)             # display output image

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Brief explanation: The *color_grayscale(filename,g)* function accepts two parameters. First parameter(filename) is the input image and second parameter is the integer g. The value of g is to be inserted on the Python Shell. If g=1 f the output will be a grayscale image. We are using OpenCV2 `cv2.COLOR_BGR2GRAY` command to convert our image to grayscale. If g is not equal to 1, we will display the original image as output.