15-112: Fundamentals of Programming and Computer Science, Fall 2019

Homework 4 Programming: Image Processing

Due: Tuesday, September 24, 2019 by 22:00

This programming homework is designed to get you more practise with processing images and applying the skills you have learnt so far, to solve real world problems.

Your submission will be made through the web interface of Autolab. In this homework, you will be writing several functions. Write all functions in the same file and call that file YourAndrewIDhw4.py. You should not have any test code in this file besides the function definitions plus any other helper functions and import statements you want to write. You should not have any main code that is executed. Your functions should be named according to the specifications given in the questions below. Again, if you want to write helper functions within the same file to help you organize your code, you are more than welcome to do so and you can name them whatever you want. You should submit this python file under lab4 option at:

`https://autolab.andrew.cmu.edu/courses/15112q-f19`

1 Check it out!

Whenever you buy something from the grocery store, the person at the checkout counter scans the code at the back of the product you are trying to purchase. This code is called a barcode, or more specifically, a UPC barcode, used to mark trade items. An image of a sample barcode is shown in Figure1:

Figure 1: A Sample UPC barcode - Source "How stuff works"

The UPC bar code consists of a series of vertical lines of varying width that together constitute a 12 digit number (UPC-A). To figure out how a UPC works, please read a very good explanation given at the link below - especially section 4 titled: "Can I decode the Bars?"

http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/upc.htm

Take the example given on the webpage and decode it manually using the directions given on the link above.

Your task, in this homework, is to take an image of a barcode and pragmatically (by writing a computer program) determine the numbers encoded in the barcode image. To assist you in this task, I am providing you with a library that allows all essential functions you need to process an image. Your program should load the image and then decode the barcode on the image. You can assume that the barcode image file exists in the same directory as your program and is indeed a barcode image file (You don't need to error check the barcode file). Your program should also use the "check digit" algorithm described on the "howstuffworks" website to determine whether your answer is valid or not. Following is a list of tasks (functions) that you will need to do to decode the barcode.

This homework needs the ImageWriter Library that can be downloaded from the Resrouces section of the course website:

http://www.qatar.cmu.edu/~srazak/courses/15112-f19/resources.php

The ImageWriter library uses another library called "opencv" for image processing. The opencv library is installed in the cluster computers in rooms 1032, 2035, and 1185. If you use a personal computer, make sure that you install this library by following the instructions given on the resource section of the course website. If you have "pip" installed, you should be able to install opencv by running the following command from a shell or command prompt:

```
pip3 install opencv-python
```

You can go to the website to generate barcode images of your choice. Please use UPC-A for barcode symbology and jpg as output format. I will be posting some sample barcodes as well.

http://www.barcoding.com/upc/

Task 1 (1 pt) *Write a function convertBlackWhite(pic) that takes a picture or image as input (You can assume that this picture is already loaded) and converts the colored image to a black and white image.*

Task 2 (6 pts) *Write a function called searchForStartPoint that takes a black and white picture and locates a the x and y axis of the point where the barcode starts. In this function, the y location is not as important, it can by anywhere vertically where there is the beginning of thr barcode. The x value has to be the exact horizontal pixel where the first bar is starting. The function should return the value of x and y as a list with index 0 as x and index 1 as y.*

Task 3 (6 pts) *Write a function called getUnitWidth that takes a a picture and the x and y axis of the start point of the first reference bar (reference bars are the first three bars in the code where each bar is one-unit-wide). The function should find the average width of next three bars (the reference bars) and return the average of their widths. The function should also return the x and y location of the* ***end of reference bars****. The return value should be a list in the format [average width, x ,y]. The x value should be the last pixel in the horizontal direction that is part of the last one-unit-bar.*

Task 4 (10 pts) *Write a function called getNextSequence that takes a a picture, the width of the unit bar, the x and y axis of the start point of the next data bar, as input arguments. The function should then return value the width of the next four bars in unit widths. The function should also return the x and y locations of the* ***end of the last bar****. This information should be packed in a list with format [width of bar1, width of bar2, width of bar3, width of bar4, x, y] - all widths should be in unit-widths (so if unit width is 3 pixels and a bar is 6 pixels that would equal 2 unit widths).*

Task 5 (5 pts) *Write a function called skipNextBar that takes the picture, and the x and y location of the current bar. This function should skip the current bar and return the x and y axis of the* ***last horizontal pixel of the current bar*** *as a list.*

Task 6 (10 pts) *Write a function called decodeSequence that takes a list of four widths of bars in unit-widths and returns the digit that this sequence represents. The return value of this function should be a single int.*

Task 7 (8 pts) *The last digit of the UPC code is called a check digit. This digit lets the scanner determine if it scanned the number correctly or not. Here is how the check digit is calculated for the other 11 digits, using the code 63938200039: (Source - HowStuffWorks)*

- *Add together the value of all of the digits in odd positions (digits 1,3,5, 7, 9 and 11). 6 + 9 + 8 + 0 + 0 + 9 = 32*
- *Multiply that number by 3. 32 * 3 = 96*
- *Add together the value of all of the digits in even positions (digits 2,4, 6, 8 and 10). 3 + 3 + 2 + 0 + 3 = 11*
- *Add this sum to the value in step 2. 96 + 11 = 107*
- *Take the number in Step 4. To create the check digit, determine the number that, when added to the number in step 4, is a multiple of 10. 107 + 3 = 110 The check digit is therefore 3.*

Write a function called verifyCode that takes a list of 12 integers and returns True if the check digit is valid, otherwise return False.

The following function puts together the functionality provided by your functions and decodes a barcode image passed to it. You are not allowed to change this code. If all your functions are working properly, this function will successfully decode the barcode and print the result.

```
def decodeBarcode(filename):
   # an empty list to hold the final barcode values
   barcode = []
   # load the image into variable pic
   pic = ImageWriter.loadPicture(filename)
   # convert the image to black and white
   convertBlackWhite(pic)
   # search for x and y location of first bar
   # note y is more flexible, x is not!
   start = searchForStartPoint(pic)
   x = start[0]
   y = start[1]
   # get unit widths from the reference bars at the beginning of the barcode
   unitInfoWithPoints = getUnitWidth(pic,x,y)
   unit = unitInfoWithPoints[0]
   x = unitInfoWithPoints[1]
   y = unitInfoWithPoints[2]
   # read the first six digits
   for i in range(0,6):
       # read the width of next four bars
       newSequence = getNextSequence(pic,unit,x+1,y)
       # decode the sequence to find the digit
       code = decodeSequence(newSequence[:4])
       # add the new digit to our list of digits
       barcode.append(code)
       x = newSequence[4]
       y = newSequence[5]
   # Skip the next five standard 1-1-1-1-1 bars
   for i in range(0,5):
       loc = skipNextBar(pic,x+1,y)
       x = loc[0]
       y = loc[1]
   # read the next six digits
   for i in range(0,6):
       newSequence = getNextSequence(pic,unit,x+1,y)
       code = decodeSequence(newSequence[:4])
       barcode.append(code)
       x = newSequence[4]
       y = newSequence[5]
   # verify that the code is valid and print result accordingly
   if (verifyCode(barcode)):
       print "valid barcode: ",str(barcode)
   else:
       print "ERROR: Invalid barcode: ",str(barcode)
```

Task 8 (-5 pts) *In this assignment, you will be graded on style. Follow the guidelines for HW3 for good style practices.*