24-678: Computer Vision for Engineers

Carnegie Mellon University

PS6

Due: **4/1/2022 (Fri) 5 PM @ Gradescope**

Issued: **3/23/2022 (Wed)**

Weight: **5% of total grade**

Note:

PS6-1 Part Identification and Classification

Factory automation is one of the major industrial applications of computer vision. Figure 1 shows a typical configuration of a vision-based part inspection system.

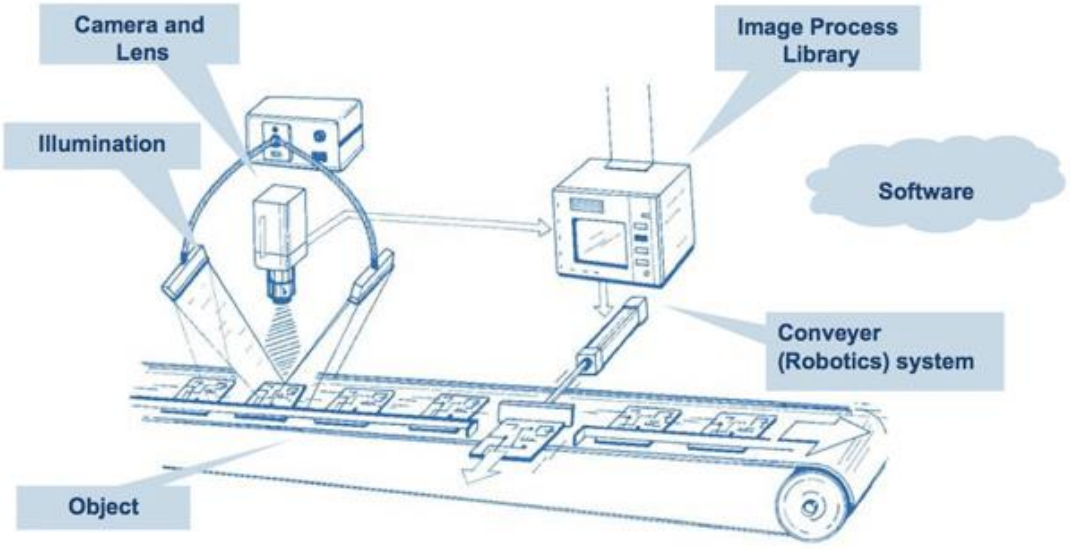
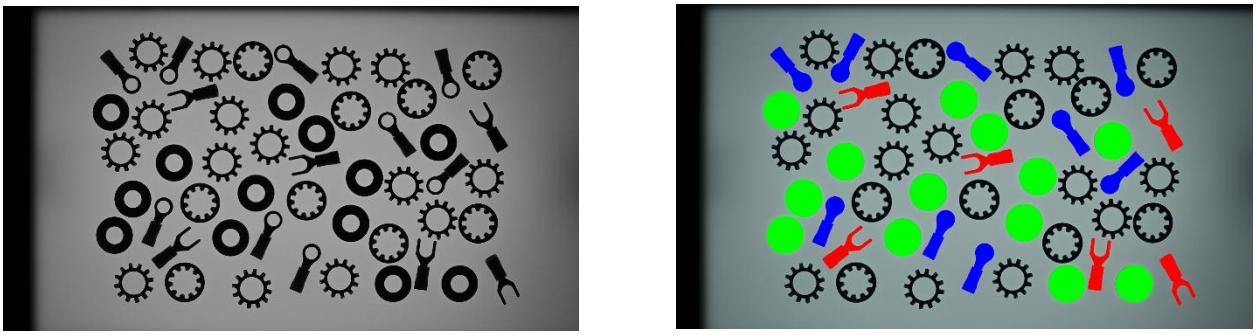


Figure 1: Typical part inspection system for factory automation

A sample OpenCV program shown in Appendix A takes as input an image of five different types of mechanical parts and identifies three of them by using geometric criteria. Defective parts are marked by different colors. Out of the following five types of mechanical parts, the program identifies the first three:

- Ring terminal
- Spade terminal
- Washer
- Internal-tooth lock washer
- External-tooth lock washer

Figure 2: Five types of mechanical parts



(a) Input (b) Output

Figure 3: Input and output of the program listed in Appendix A

In this problem set, you will study the sample program in Appendix A, understand how it works, and then modify the program so that all five types of mechanical parts are correctly identified. Paint the internal lock washers in purple and external lock washers in yellow.

Your program should take as input an image, “all-parts.png,” shown in Figure 3 (a) and generate a new file “all-parts-output.png.”

Submission

To prepare for the submission of your work on Gradescope, create:

(1) a folder called “ps6-1,” that contains the following files:

- source code file(s)
- output image created by your program:
  - all-parts-output.png
- "readme.txt" file that includes:
  - Operating system
  - IDE you used to write and run your code
  - The number of hours you spent to finish this problem

(2) a PDF file that contains the printouts and screenshots of all the files in the ps6-1 folder. Include the explanation of what modifications you made to detect all five types of mechanical parts. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

PS6-2 Detecting Defective Parts

matchShapes is a powerful OpenCV function that takes in two binary images (or contours) and finds the distance between them using Hu Moments (M. K. Hu, "Visual Pattern Recognition by Moment Invariants", IRE Trans. Info. Theory, vol. IT-8, pp.179–187, 1962). See some information about Hu Moments and how to use matchShapes on the web at: https://learnopencv.com/shape-matching-using-hu-moments-c-python/.

Using matchShapes, write a program that takes as input the image shown in Figure 5 (a), "spade-terminal.png," detects defective spade terminals and marks them by painting them in red as shown in Figure 5 (b). The program should save the output image as spade-terminal-output.png.

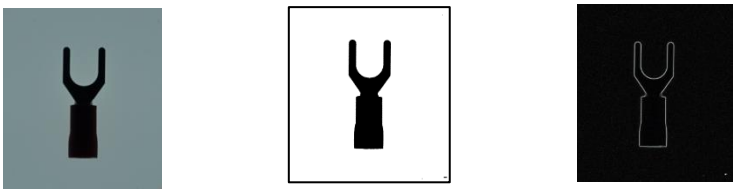


Figure 4: Template images (original, binary, and contour) of a spade terminal

(a) Input (b) Output

Figure 5: Input image of multiple space terminals and detected defective ones

Submission

To prepare for the submission of your work on Gradescope, create:

(3) a folder called "ps6-2," that contains the following files:

- source code file(s)
- output image created by your program:
  - spade-terminal-output.png
- "readme.txt" file that includes:
  - Operating system
  - IDE you used to write and run your code
  - The number of hours you spent to finish this problem

(4) a PDF file that contains the printouts and screenshots of all the files in the ps6-2 folder. Include the explanation of what thresholds you used to detect defect parts. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

Submit your work on Gradescope

Submit two files on Gradescope – replace “andrewid” with your own Andrew ID:

(1) **andrewid-ps6-files.zip** – this ZIP file should contain two folders, “ps6-1” and “ps6-2,” and all the files requested in PS6-1 and PS6-2.

Please make sure that files are organized in two separate folders in the ZIP file: “ps6-1” and “ps6-2.”

(2) **andrewid-ps6-report.pdf** – this PDF file serves as the report of your work, and it should contain the printouts and screenshots of all the files in the “ps6-1” folder and “ps6-2” folder. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

Please organize pages with section titles and captions to make the report easy to read.

Appendix A

```python
import cv2
import numpy as np
import argparse

# check size (bounding box) is square
def isSquare(siz):
    ratio = abs(siz[0] - siz[1]) / siz[0]
    #print (siz, ratio)
    if ratio < 0.1:
        return True
    else:
        return False

# chekc circle from the arc length ratio
def isCircle(cnt):
    (x,y),radius = cv2.minEnclosingCircle(cnt)
    len = cv2.arcLength(cnt, True)
    ratio = abs(len - np.pi * 2.0 * radius) / (np.pi * 2.0 * radius)
    #print(ratio)
    if ratio < 0.1:
        return True
    else:
        return False

if __name__ == "__main__":
    #
    parser = argparse.ArgumentParser(description='Hough Circles')
    parser.add_argument('-i', '--input', default = 'image/parts-all.png')

    args = parser.parse_args()
    # Read image
    img = cv2.imread(args.input)

    # Convert to gray-scale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Binary
    thr,dst = cv2.threshold(gray, 60, 255, cv2.THRESH_BINARY)

    # clean up
    for i in range(1):
        dst = cv2.erode(dst, None)
    for i in range(1):
        dst = cv2.dilate(dst, None)

    # find contours with hierachy
    cont, hier = cv2.findContours(dst, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    # each contoure
    for i in range(len(cont)):
        c = cont[i]
        h = hier[0,i]
        if h[2] == -1 and h[3] == 0:
            # no child and parent is image outer
            img = cv2.drawContours(img, cont, i, (0,0,255),-1)
        elif h[3] == 0 and hier[0,h[2]][2] == -1:
            # with child
            if isCircle(c):
                if isCircle(cont[h[2]]):
                    # double circle
                    img = cv2.drawContours(img, cont, i, (0,255,0),-1)
            else:
                # 1 child and shape bounding box is not squre
                if not isSquare(cv2.minAreaRect(c)[1]) and hier[0,h[2]][0] == -1 and hier[0,h[2]][1] == -1:
                    img = cv2.drawContours(img, cont, i, (255,0, 0),-1)


    cv2.namedWindow("Image",cv2.WINDOW_NORMAL)
    cv2.imshow("Image", img)
```