24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS6 Report
Due: Nov 10 2023

This file contains the following:
PS6-1 Part Identification and Classification

- all-parts-output.png
- readme.txt
- source code file(s) (attached to the end)

PS6-2 Detecting Defective Parts

- spade-terminal-output.png
- readme.txt
- source code file(s) (attached to the end)

Findings and discussion:
PS6-1

We are tasked to identify and label 5 types of mechanical parts: ring terminal, spade terminal, washer, internal lock washer, and external lock washer for this exercise. Some code was provided as the basic framework for parts detection.

The given code was missing the contour identification of internal lock washers and external look washers. To address this issue, I added two contour identification logic for specific part identification. For identifying internal lock washers, the program validates the contour having only the parent being circular and draws it in purple. For identifying external lock washers, the program validates the contour having the bounding box as square and draws it in yellow. I also increased the dilation iteration (4) to achieve better image results.

The results were satisfactory and the program was able to achieve the given tasks.

PS6-2

We are tasked to identify and label the defective spade terminal from the given image. The program first cleaned up the original image by performing erosion and dilation operations. To identify the defective spade terminal, I set the shape-matching threshold at 1.5 (higher is a stricter criterion for matching shapes) and labeled the defect in red. I also performed erosion and dilation operations to achieve better image results.

The results were satisfactory and the program was able to achieve the given tasks.

PS6-1 all parts original & output image

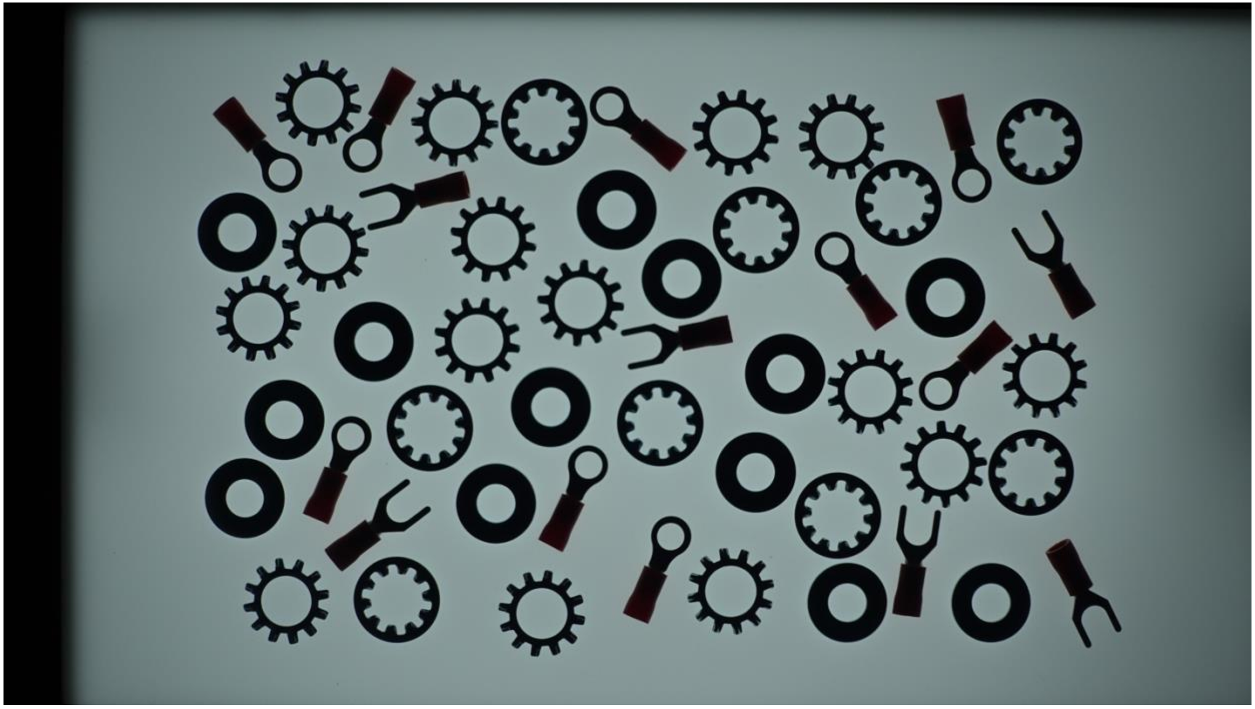


Figure 1. The given all parts image for part identification and classification.

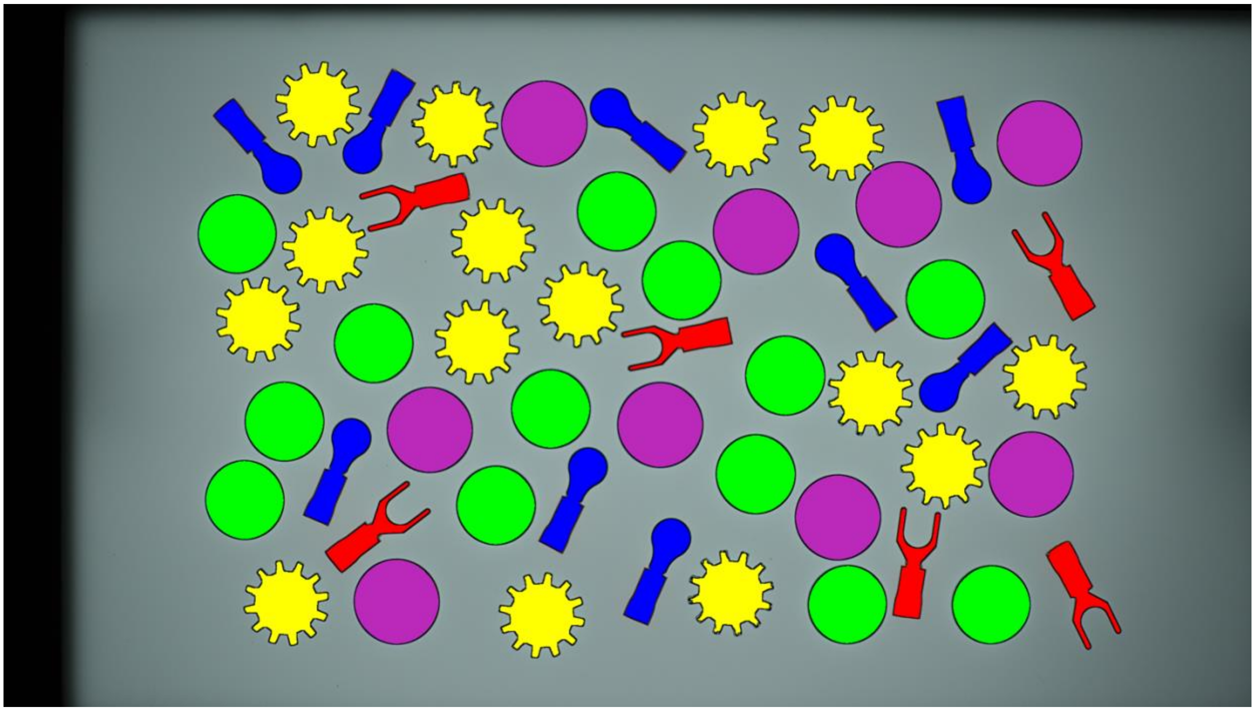
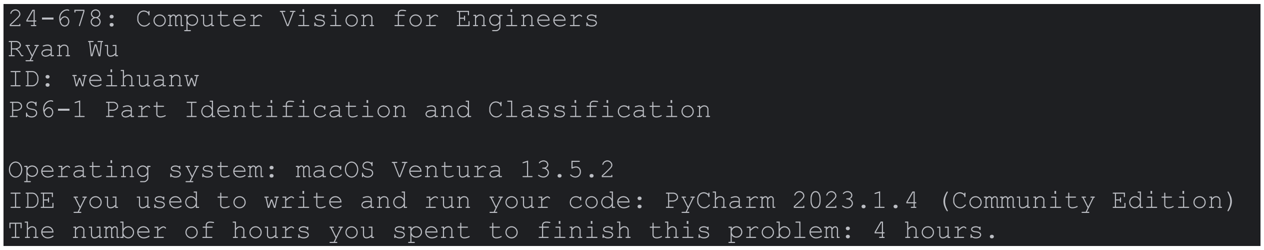


Figure 2. The output all parts image with different parts identified and color labeled.

PS6-1 readme.txt



```
24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS6-1 Part Identification and Classification

Operating system: macOS Ventura 13.5.2
IDE you used to write and run your code: PyCharm 2023.1.4 (Community Edition)
The number of hours you spent to finish this problem: 4 hours.
```

PS6-2 spade terminal original & output image

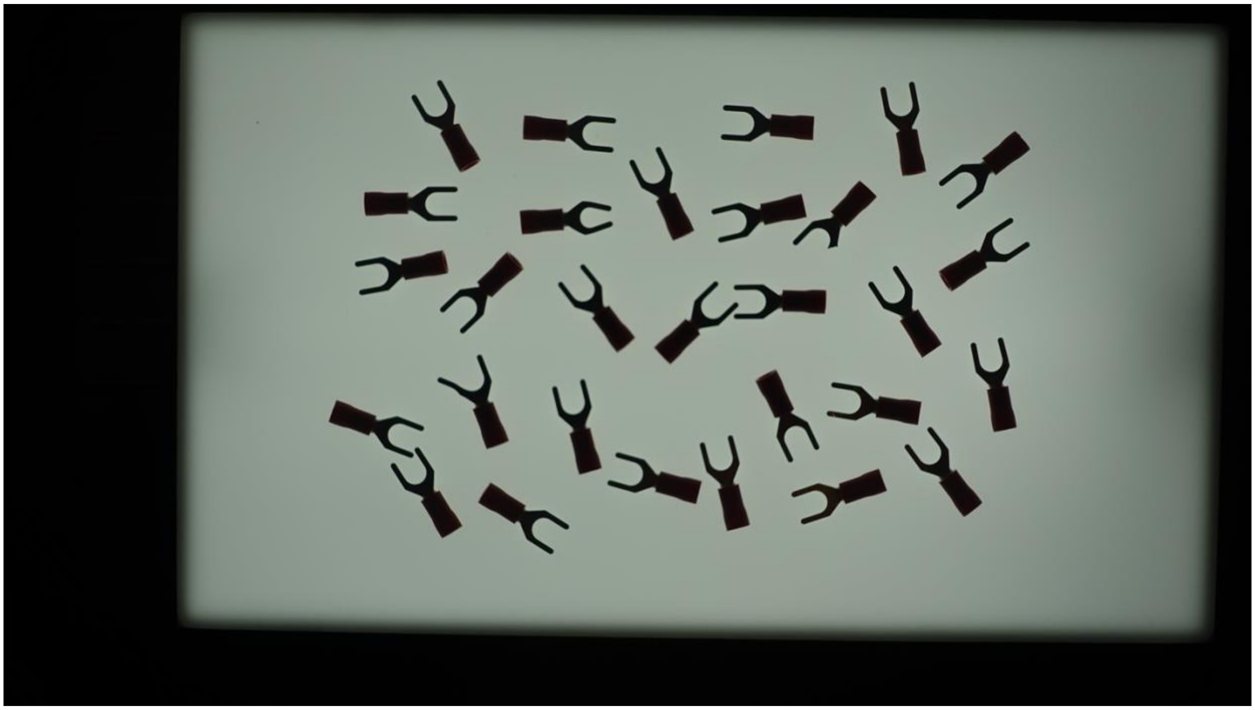


Figure 3. The given original image for defect detection.

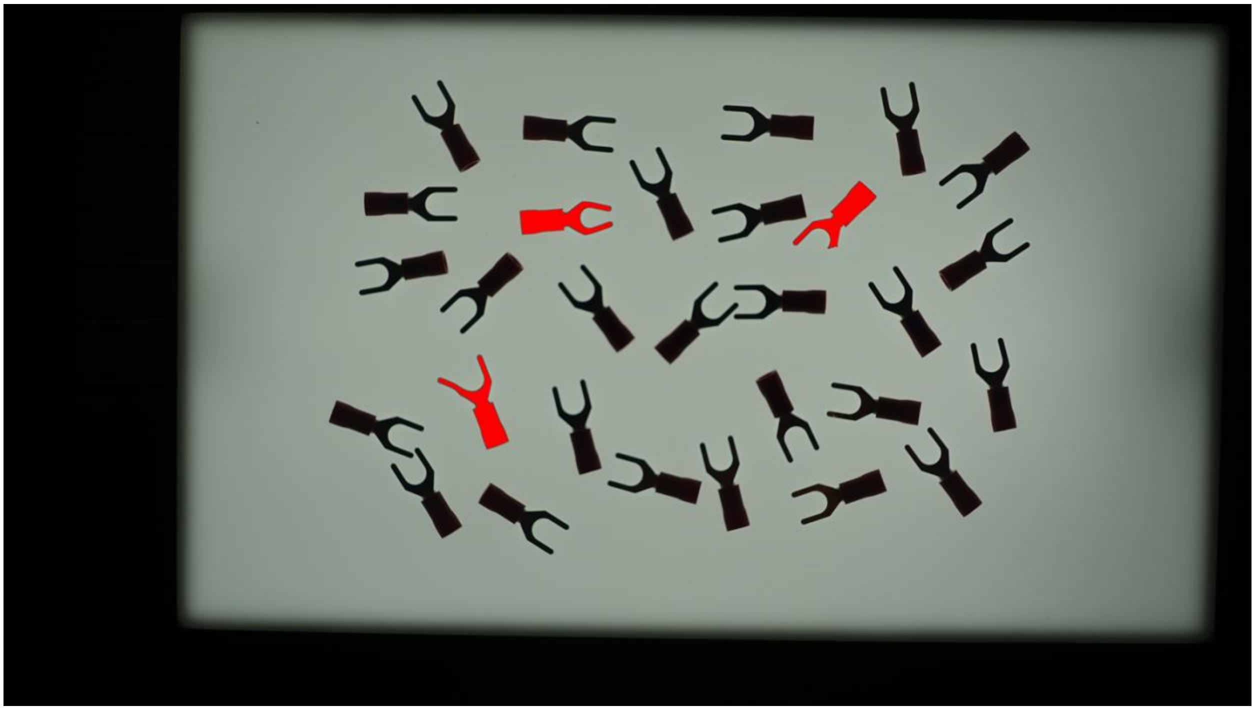
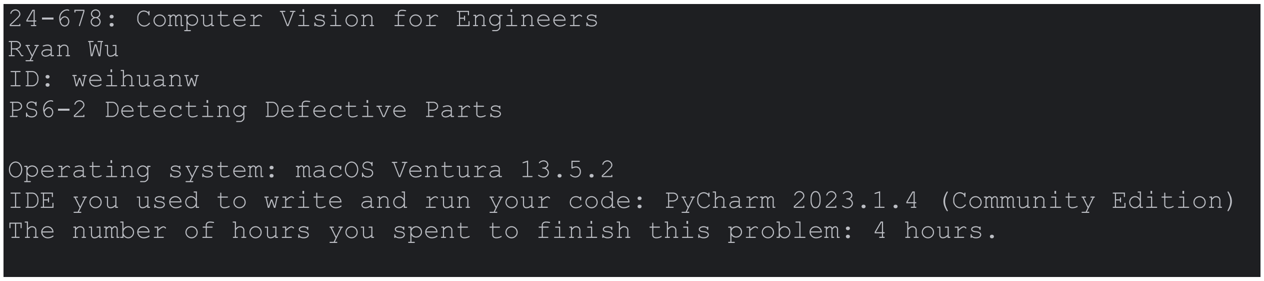


Figure 4. The output image with defects shown in red.

PS6-2 readme.txt



```
24-678: Computer Vision for Engineers
Ryan Wu
ID: weihuanw
PS6-2 Detecting Defective Parts

Operating system: macOS Ventura 13.5.2
IDE you used to write and run your code: PyCharm 2023.1.4 (Community Edition)
The number of hours you spent to finish this problem: 4 hours.
```

```python
# 24-678 Computer Vision for Engineers
# Ryan Wu (ID:weihuanw)
# PS06-1 Part Identification and Classification
# Due 11/10/2023 (Fri) 5 pm

# import the necessary packages
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

# check 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='all-parts.png')

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

    # Convert to grayscale
    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 (4):
        dst = cv2.dilate(dst, None)

    # find contours with hierarchy
    cont, hier = cv2.findContours(dst, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # filter out small contours based on area
    cont = [c for c in cont if cv2.contourArea(c) > 100]

    # each contour
    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:
                    # single circle
                    img = cv2.drawContours(img, cont, i, (187,41,187), -1)
            else:
                # 1 child and shape bounding box is not square
                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)
                # 2 children and shape bounding box is square
                elif isCircle(cont[h[2]]):
                    img = cv2.drawContours(img, cont, i, (0,255,255), -1)

    cv2.namedWindow('image', cv2.WINDOW_NORMAL)
    cv2.imshow('image', img)
    cv2.imwrite('all-parts-output.png', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```python
# 24-678 Computer Vision for Engineers
# Ryan Wu (ID:weihuanw)
# PS06-2 Detecting Defective Parts
# Due 11/10/2023 (Fri) 5 pm

# import the necessary packages
import cv2

# defect detection function
def detect_defect(image):
    # convert to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # convert to binary
    _, dst = cv2.threshold(gray_image, 60, 255, cv2.THRESH_BINARY)

    # dilation
    for contours in range(1):
        dst = cv2.erode(dst, None)

    # erosion
    for contours in range(2):
        dst = cv2.dilate(dst, None)

    # set a threshold for shape matching
    matching_threshold = 1.5

    # set a threshold for filtering out the edge
    max_contour_area = 50000

    # find contours
    cont, _ = cv2.findContours(dst, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # contour matching and draw contours
    for contours in range(len(cont)):
        c = cont[contours]
        match_contour = cv2.matchShapes(cont[3], c, cv2.CONTOURS_MATCH_I2, 0)
        if match_contour > matching_threshold and cv2.contourArea(c) < max_contour_area:
            image = cv2.drawContours(image, cont, contours, (0, 0, 255), -1)

    # display the output image
    cv2.imshow('spade-terminal-output image', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    # Save the output image
    cv2.imwrite('spade-terminal-output.png', image)

if __name__ == "__main__":
    input_image = cv2.imread("spade-terminal.png")
    detect_defect(input_image)



```