Compsci373 opencv type solution brief report

According the message that I received from Martin, I will use the opencv version of the solution as my extension.

> deadline, and I will count your experiments with OpenCV that you have already done as the extension part to the assignment. You would just need to document and reflect on what you have already done with OpenCV.
>
> I hope this is an acceptable compromise and helps you with your assignment clashes.
>
> kind regards
> Martin
>
>  **Martin Urschler**
>
> You can reply to this message in Canvas by replying directly to this email. If you need to include an attachment, please log in to Canvas and reply through the Inbox.

For the opencv type solution:

1.

```
import cv2
```

Firstly we need to import opencv package.

2.

```
def obtain_gray_image(src):
    gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
    return gray
```

This function for obtain grayscale image.

3.

```
def obtain_sobel_image(img):
    x = cv2.Sobel(img, cv2.CV_16S, 1, 0)
    y = cv2.Sobel(img, cv2.CV_16S, 0, 1)

    absX = cv2.convertScaleAbs(x)
    absY = cv2.convertScaleAbs(y)

    dst = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)
    return dst
```

This function for using the sobel operator to obtain the edge image

The first two of the lines to get the edges in the x and y directions

```
    dst = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)
```

And this line for getting the entire edge

4.

```
def obtain_mean_filter_image(img):
    img_blur = cv2.blur(img, (3, 3))
    img_blur = cv2.blur(img_blur, (4, 4))
    return img_blur
```

And this function for getting two average filtering.

5.

```
def obtain_threshold_image(img):
    ret, binary = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_TRIANGLE)
    return binary
```

This function for getting binarized image

6.

```
def obtain_contour_image(img):
    contours, hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    x = 0
    y = 0
    w = 0
    h = 0
    if len(contours) != 0:


        c = max(contours, key=cv2.contourArea)
        x, y, w, h = cv2.boundingRect(c)
    return x,y,w,h
```

This part for finding the largest connected contour, and return its coordinates, height and width

```
        c = max(contours, key=cv2.contourArea)
```

This line of code for finding the biggest countour (c) by the area

7.

```
def main():
    filename = "poster1small.png"
    src = cv2.imread(filename)

    img = obtain_gray_image(src)

    dst = obtain_sobel_image(img)

    blur = obtain_mean_filter_image(dst)

    binary = obtain_threshold_image(blur)

    x,y,w,h = obtain_contour_image(binary)
    if w != 0:
        cv2.rectangle(src, (x, y), (x + w, y + h), (0, 255, 0), 5)
    cv2.imshow("result image", src)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

And finally the main function help us show the result image.