**Midterm Report**

**DIGITAL IMAGE PROCESSING**

MIDTERM

Lecturer: **Mr. Trinh Hung Cuong**

Students: **Ho Vinh Tuong - 520K0091**

**Nguyen Pham Phu Thinh - 520V0012**

Course: **24**

**Midterm Report**

**DIGITAL IMAGE PROCESSING**

MIDTERM

Lecturer: **Mr. Trinh Hung Cuong**

Students: **Ho Vinh Tuong - 520K0091**

**Nguyen Pham Phu Thinh - 520V0012**

Course: **24**

# ACKNOWLEDGEMENT

We send our most sincere thanks to Mr. Trinh Hung Cuong for always supporting and accompanying us to complete the mid-term report of Digital Image Processing in the best way. We always hope that you will always be the teacher that accompanies and gives our students the best knowledge so that they can go out and have the most solid knowledge. Once again, thank you very much for your time.

The team would also like to thank the Faculty of Information Technology and the school for creating conditions both physically and mentally so that the group could complete the mid-term report in the best way.

We would like to thank all the teachers who have supported us in the best way. We sincerely thank you!

# THE REPORT IS COMPLETED AT TDT UNIVERSITY

I hereby declare that this is the product of our own project and under the guidance of Mr. Trinh Hung Cuong. The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

In addition, the project also uses a number of comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

If I find any fraud, I will take full responsibility for the content of my project. Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*HCMC, November 15th 2022*

*Authors*

*Ho Vinh Tuong*

*Nguyen Pham Phu Thinh*

# CONFIRMATION AND ASSESSMENT SECTION

**The evaluation of lecturer**

HCMC, November 15th, 2022

**The evaluation of examiner**

HCMC, November 15th, 2022

# SUMMATION

Contents

[ACKNOWLEDGEMENT 1](#_Toc119367672)

[THE REPORT IS COMPLETED AT TDT UNIVERSITY 2](#_Toc119367673)

[CONFIRMATION AND ASSESSMENT SECTION 3](#_Toc119367674)

[SUMMATION 4](#_Toc119367675)

[CHAPTER 1: METHODOLOGY OF SOLVING TASKS 37](#_Toc119367676)

[a. Extract each star 37](#_Toc119367677)

[b. Repaint each star 37](#_Toc119367678)

[c. Repaint all stars 38](#_Toc119367679)

[d. Repaint White borders of all stars to Black color 38](#_Toc119367680)

[e. Repaint the background to White color, and repaint all stars to Black color. 38](#_Toc119367681)

[f. Draw contours of all objects 39](#_Toc119367682)

[g. Draw rectangles surrounding each contour of all stars 39](#_Toc119367683)

[h. Sharpen the image, add the text 40](#_Toc119367684)

[CHAPTER 2: TASK RESULT 40](#_Toc119367685)

[a. Extract each star 40](#_Toc119367686)

[b. Repaint each star 42](#_Toc119367687)

[c. Repaint all stars 45](#_Toc119367688)

[d. Repaint White borders of all stars to Black color 45](#_Toc119367689)

[e. Repaint the background to White color, and repaint all stars to Black color. 45](#_Toc119367690)

[f. Draw contours of all objects 46](#_Toc119367691)

[g. Draw rectangles surrounding each contour of all stars 46](#_Toc119367692)

[h. Sharpen the image, add the text 46](#_Toc119367693)

[REFERENCE 46](#_Toc119367694)

# CHAPTER 1: METHODOLOGY OF SOLVING TASKS

## Extract each star

* Convert the input image to HSV color.
* Find mask range of HSV of the star’s color (lower and upper)
* Extract the star
* Reduce noise
* The result is a list of the extracted star and its mask range

Code:

def extractStar(img, bgrColor):

    # cvt to HSV img

    hsvImg = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

    # cvt color: BGR to HSV

    hsvColor = cv2.cvtColor(bgrColor, cv2.COLOR\_BGR2HSV)

    # calc lower and upper HSV

    maskColor = cv2.inRange(hsvImg, np.subtract(

        hsvColor, [[[10, 100, 100]]]), np.add(hsvColor, [[[10, 255, 255]]]))

    # extract

    res = cv2.bitwise\_and(img, img, mask=maskColor)

    # reduce noise

    res = cv2.morphologyEx(res, cv2.MORPH\_OPEN, (5, 5))

    # return pair of image of each star and its mask range

    return [res, maskColor]

Use the function:

starImgs = [extractStar(img, i) for i in colorList]

for i in range(len(starImgs)):

        # show(i[0])

        cv2.imwrite('star\_' + str(i+1) + ".jpg", starImgs[i][0])

## Repaint each star

* Convert the input image to HSV image.
* Find mask range of HSV of the star’s color (lower and upper)
* Replace the mask range by the difference HSV value of green and every single colors
* Convert the HSV image to BGR image.

Code:

def toGreen(img, color, diff):

    hsvImg = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

    hsvColor = cv2.cvtColor(color, cv2.COLOR\_BGR2HSV)

    mask = cv2.inRange(hsvImg, np.subtract(

        hsvColor, [[[10, 100, 100]]]), np.add(hsvColor, [[[10, 255, 255]]]))

    hsvImg[mask > 0] += diff

    hsvImg = cv2.cvtColor(hsvImg, cv2.COLOR\_HSV2BGR)

    return hsvImg

Run code:

greenStars = [toGreen(img, colorList[i], diffColor[i]) for i in range(6)]

for i in range(len(greenStars)):

    # show(i)

    cv2.imwrite('green\_star\_' + str(i+1) + ".jpg", greenStars[i])

## Repaint all stars

The same logic to the below function. However, I use loop to color all stars.

Code:

def toAllGreen(img, colorList, diffList):

    hsvImg = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

    for i in range(len(colorList)):

        # calc mask range

        hsvColor = cv2.cvtColor(colorList[i], cv2.COLOR\_BGR2HSV)

        maskColor = cv2.inRange(hsvImg, np.subtract(

            hsvColor, [[[10, 100, 100]]]), np.add(hsvColor, [[[10, 255, 255]]]))

        # replace range by relative diffColor

        hsvImg[maskColor > 0] += diffList[i]

    hsvImg = cv2.cvtColor(hsvImg, cv2.COLOR\_HSV2BGR)

    return hsvImg

Run code:

cv2.imwrite('all\_green\_stars.jpg', toAllGreen(img, colorList, diffColor))

## Repaint White borders of all stars to Black color

* Convert image to grayscale
* Use thresholding with THREST\_TOZERO\_INV
* Reduce noise by MORPH\_OPEN

Code:

def blackBorder():

    imgGray = cv2.imread('input1.jpg', 0)

    ret, th = cv2.threshold(imgGray, 220, 255, cv2.THRESH\_TOZERO\_INV)

    th = cv2.morphologyEx(th, cv2.MORPH\_OPEN, (3, 3))

    # show(th)

    return th

Run code:

cv2.imwrite('black\_borders.jpg', blackBorder())

## Repaint the background to White color, and repaint all stars to Black color.

* Mostly the same to the above function
* Use THRESH\_TOZERO\_INV
* Use THRESH\_BINARY

Code:

def blackStar():

    imgGray = cv2.imread('input1.jpg', 0)

    # src >= 190 ==> src = 0

    ret, th = cv2.threshold(imgGray, 190, 0, cv2.THRESH\_TOZERO\_INV)

    # scr > 180 => src=255, else src=0

    ret, th = cv2.threshold(th, 180, 255, cv2.THRESH\_BINARY)

    th = cv2.morphologyEx(th, cv2.MORPH\_OPEN, (5, 5))

    # show(th)

    return th

Run code:

cv2.imwrite('black\_stars.jpg', blackStar())

## Draw contours of all objects

* Copy the input image
* Convert the copied one to grayscale
* Use THRESH\_BINARY
* Create kernel 5x5
* Reduce noise
* Find contours
* Get x, y coordinates and their lengths
* Use drawContours

Code:

def contourStar(img):

    img1 = np.copy(img)

    imgGray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

    \_, th = cv2.threshold(imgGray, 220, 255, cv2.THRESH\_BINARY)

    kernel = np.ones((5, 5), np.uint8)

    # reduce noise

    th = cv2.morphologyEx(th, cv2.MORPH\_OPEN, kernel)

    th = cv2.morphologyEx(th, cv2.MORPH\_CLOSE, kernel)

    # find contours

    contours, hierarchy = cv2.findContours(

        th, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

    for cnt in contours:

        x, y, w, h = cv2.boundingRect(cnt)

        cv2.drawContours(img1, [cnt], -1, (0, 0, 255), 2)

    return img1

Run code:

cv2.imwrite('contour\_rec\_stars.jpg', contourRec(img))

## Draw rectangles surrounding each contour of all stars

* The same to the above function but using retangle to draw

Code:

def contourRec(img):

    img1 = np.copy(img)

    imgGray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

    \_, th = cv2.threshold(imgGray, 220, 255, cv2.THRESH\_BINARY)

    kernel = np.ones((5, 5), np.uint8)

    # reduce noise

    th = cv2.morphologyEx(th, cv2.MORPH\_OPEN, kernel)

    th = cv2.morphologyEx(th, cv2.MORPH\_CLOSE, kernel)

    # find contours

    contours, hierarchy = cv2.findContours(

        th, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

    for cnt in contours:

        x, y, w, h = cv2.boundingRect(cnt)

        cv2.rectangle(img1, (x, y), (x+w, y+h), (0, 255, 0), 2)

    return img1

Run code:

cv2.imwrite('sharped\_image.jpg', sharpen(img))

## Sharpen the image, add the text

* Create kernel
* Use filter2D

Code:

def sharpen(img):

    # smoothed = cv2.GaussianBlur(img, (9, 9), 10)

    # unsharped = cv2.addWeighted(img, 1.5, smoothed, -0.5, 0)

    # show(unsharped)

    kernel = np.array([[0, -1, 0],

                       [-1, 5, -1],

                       [0, -1, 0]])

    imgSharped = cv2.filter2D(img, ddepth=-1, kernel=kernel)

    imgSharped = cv2.putText(imgSharped, '520V0012', (346, 326),

                             cv2.FONT\_HERSHEY\_SIMPLEX, 2, color=(0, 0, 0), thickness=2)

    # show(imgSharped)

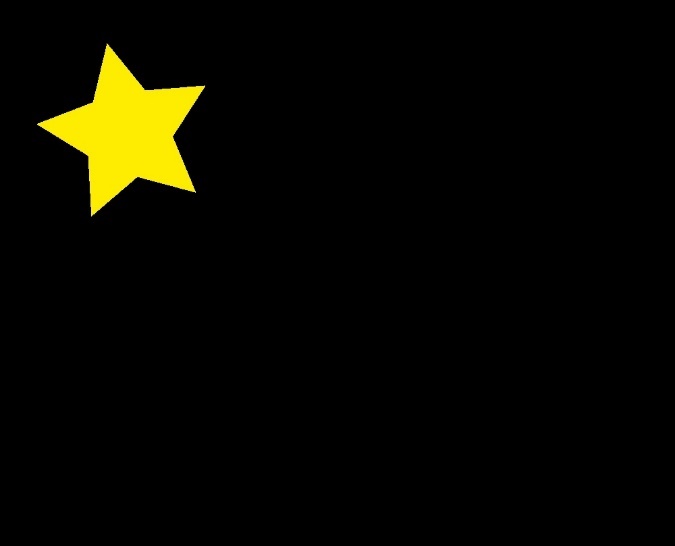
    return imgSharped

Run code:

cv2.imwrite('sharped\_image.jpg', sharpen(img))

# CHAPTER 2: TASK RESULT

## Extract each star







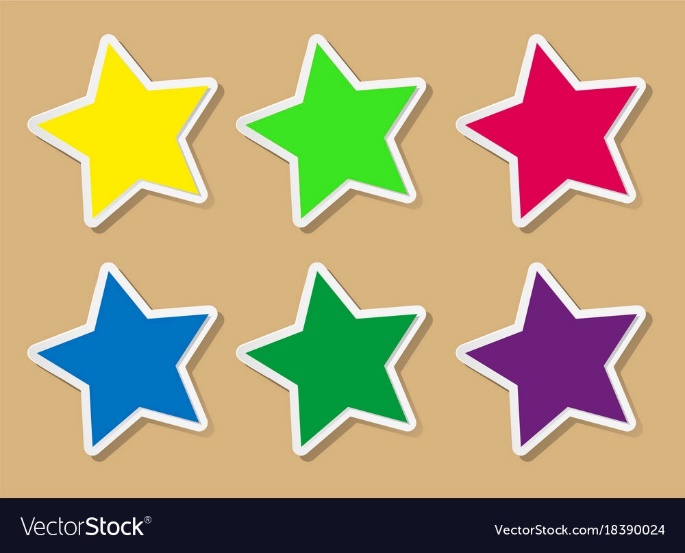


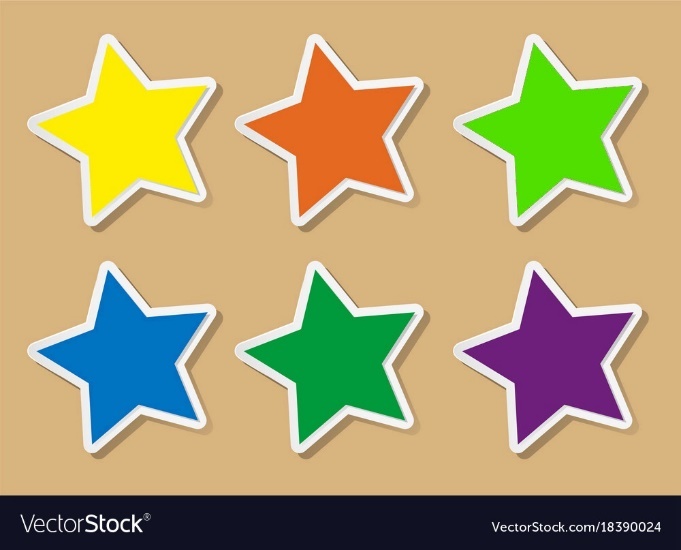




## Repaint each star













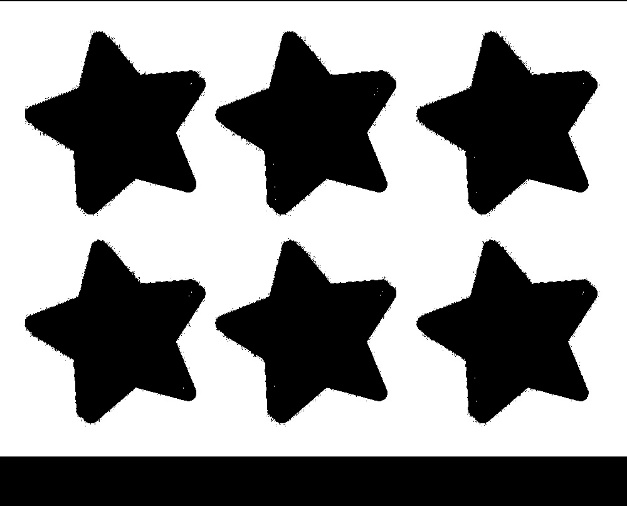
## Repaint all stars



## Repaint White borders of all stars to Black color



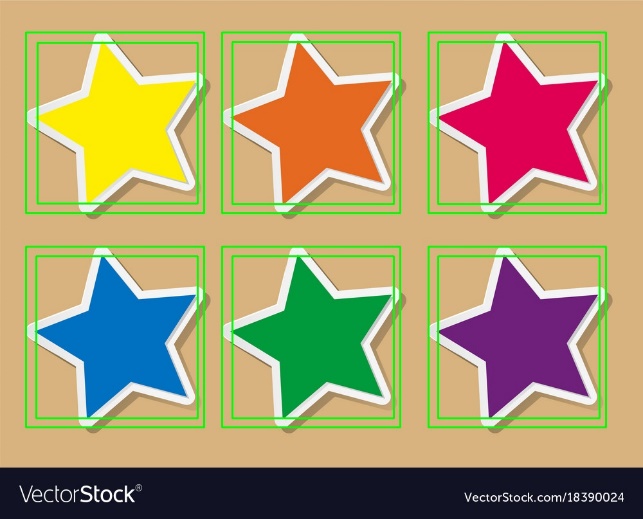
## Repaint the background to White color, and repaint all stars to Black color.



## Draw contours of all objects



## Draw rectangles surrounding each contour of all stars



## Sharpen the image, add the text



# REFERENCE

[How to check if an image contour is convex or not in OpenCV Python? (tutorialspoint.com)](https://www.tutorialspoint.com/how-to-check-if-an-image-contour-is-convex-or-not-in-opencv-python)

[OpenCV: OpenCV-Python Tutorials](https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html)