VIETNAM GENERAL CONFEDERATION OF LABORERS

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**Introduction to Digital Image Processing**

**Midterm Essay**

*Instructor*: **TS TRỊNH HÙNG CƯỜNG**

*Authors*: **PHẠM DUY KHÁNH – 522H0064**

**NGUYỄN HUỲNH ANH KHOA – 522H0046**

Class**: 505060**

Course**: 26**

**HO CHI MINH CITY, 2024**

VIETNAM GENERAL CONFEDERATION OF LABORERS

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**Introduction to Digital Image Processing**

**Midterm Essay**

Instructor: **TS TRỊNH HÙNG CƯỜNG**

Authors: **PHẠM DUY KHÁNH – 522H0064**

**NGUYỄN HUỲNH ANH KHOA – 522H0046**

Class**: 505060**

Cousre**: 26**

**HO CHI MINH CITY, 2024**

ACKNOWLEDGEMENT

First of all, we would like to thank Dr. Trịnh Hùng Cường for teaching and guiding us patiently so that we could complete the report without encountering much difficulty in terms of knowledge.

Next, we would like to express our gratitude to all the esteemed lecturers in the Faculty of Information Technology at Ton Duc Thang University for their contributions, knowledge, and support that they have provided to us throughout our learning process and the completion of the essay.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS Trịnh Hùng Cường;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Phạm Duy Khánh*

*Nguyễn Huỳnh Anh Khoa*

LECTURERS'S CONFIRMATION AND ASSESSMENT

SECTION

**The confirmation part of the instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh city, date month year

(Sign and write full name)

**The evaluation part of the teacher marks the essay**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh city, date month year

(Sign and write full name)

SUMMARY

Introduction to Digital Image Processing provide a fundemental knowledge about how to process an image. In this essay, we will dive into how to use OpenCV library to dominate an image and how to extract character from an image. By applying useful knowledge which I’ve learned in the course, we can use an appropriate approach to process a digital image.

TABLE OF CONTENTS

[ACKNOWLEDGEMENT i](#_Toc162523544)

[LECTURERS'S CONFIRMATION AND ASSESSMENT iii](#_Toc162523545)

[SECTION iii](#_Toc162523546)

[SUMMARY iv](#_Toc162523547)

[TABLE OF CONTENTS 1](#_Toc162523548)

[CHAPTER 1: Methodology of Solving Tasks 2](#_Toc162523549)

[1.1 First Task 2](#_Toc162523550)

[1.1.1 Introduction 2](#_Toc162523551)

[1.1.2 Image Loading and Color Conversion 2](#_Toc162523552)

[1.1.3 Color Dectection and Masking 3](#_Toc162523553)

[1.1.4 Image Thresholding and Morphological Operations 4](#_Toc162523554)

[1.1.5 Contour Detection and Filling 4](#_Toc162523555)

[1.2 Second Task 6](#_Toc162523556)

[1.2.1 Introduction 6](#_Toc162523557)

[1.2.2 Image Loading and Color Conversion 6](#_Toc162523558)

[1.2.3 Defining Regions of Interest 6](#_Toc162523559)

[1.2.4 Function to Edit the Image and Draw Rectangles 7](#_Toc162523560)

[1.2.5 Processing Each Region of Interest 7](#_Toc162523561)

[1.2.5.1 Kernel Initialization 9](#_Toc162523562)

[1.2.5.2 Closing Operation 9](#_Toc162523563)

[1.2.5.3 Conversion to Grayscale 9](#_Toc162523564)

[1.2.5.4 Thresholding 9](#_Toc162523565)

[1.2.5.5 Erosion 10](#_Toc162523566)

[1.2.5.6 Contour Detection 10](#_Toc162523567)

[1.2.5.7 Contour Filtering and Rectangle Drawing 10](#_Toc162523568)

[1.2.5.8 Image Return 10](#_Toc162523569)

[1.2.6 Saving the Output Image 11](#_Toc162523570)

[CHAPTER 2 – TASK RESULTS 12](#_Toc162523571)

[1.3 First Task 12](#_Toc162523572)

[1.3.1 Extract each star in the input image automatically. 12](#_Toc162523573)

[1.3.2 Repaint all stars. 17](#_Toc162523574)

[1.4 Second Task 18](#_Toc162523575)

[1.4.1 Draw rectangles surrounding each digit in the input image automatically. 18](#_Toc162523576)

CHAPTER 1: Methodology of Solving Tasks

* 1. First Task
     1. Introduction

Image processing plays an important role from computer vision to medical diagnostics. In methodology of solving tasks chapter, this we will explores the fundamental image processing techniques using Python programming language. The code loads an image input1.jpg and perfoms a load of works to detect colors, enhance contours, and highlight the shapes of stars. We delve into key concepts which we have learned in this course such as color detection, thresholding, and contour analysis. We will use 2 libraries such as OpenCV, NumPy called cv, and np respectively in our code to resolve tasks.

* + 1. Image Loading and Color Conversion

In the initial step, we will use cv.imread() twice to load image input1.jpg into variables as normal color and grayscale color image, ready for manipulation. Additionally, the Python script performs color space convertion to prepare for the following operations. Specifically, the image is transformed from the default BGR (Blue-Green-Red) color space to the HSV (Hue-Saturation-Value) color model using the cv.cvtColor() function.

A close-up of a computer code

Description automatically generated

The decision to transition from the BGR to the HSV color space to simplifies color analysis and manipulation, with specifying color ranges as hue, saturation, and value offers more way natural to think about a color in terms of hue and saturation than in terms of additive or subtractive color components, making it easier to capture colors in terms of hue, and saturation, which will enhances accuracy in color identification and histogram equalization.

* + 1. Color Dectection and Masking

After image loading and color conversion process, the script proceeds color detection, and masking operations to identify specific color within the image and isolate them.

From the last part of [official documentation](https://docs.opencv.org/3.1.0/df/d9d/tutorial_py_colorspaces.html) provided by Open-Source Computer Vision, “How to find HSV values to track?”. We pass the BGR value we want, and use cv2.cvtColor(BGR\_VALUE ,cv2.COLOR\_BGR2HSV) funtion to take lower bound and upper bound, those values will be passed to color\_dict\_HSV dictionary.

A number and numbers on a white background

Description automatically generated

The for loop iterates through each color name (key) in the color\_dict\_HSV dictionary. For each key, it will access, and convert the lower and upper bound of the HSV range for that color (key) to NumPy array. By using cv.inRange() function to create a binary mask, all pixels within the HSV range are set to white, and others to black. The cv.bitwise\_and() function, using the mask as a filter. Pixels with white values in the mask will be stored in the ballon variable to save into the result image using cv.imwrite() function. To remove noise from the image, I use morphological operation, by using np.ones() to create a 5x5 structuring element. The cv.morphologyEx() opens involves erosion followed by dilation to remove all small white objects on the foreground, while dilation recovers eroded ballon edges.

A computer code with text

Description automatically generated

* + 1. Image Thresholding and Morphological Operations

The Python code applies thresholding to the grayscale image using a predefined threshold value to separate into foreground and background regions based on pixel intensity. cv.threshold() function thresholding the grayscale image helps brighter regions by setting pixels above 225 to white and keeping others as their original values. As the same method to de-noise as above, we create a 5x5 kernel for dilation. The cv.dilate() function dilates the thresholded image using the kernel, expanding bright regions and making stars more noteworthy.

A close-up of a computer code

Description automatically generated

* + 1. Contour Detection and Filling

By finding contours within the dilated image, we can pinpoint the outlines of the stars by using cv.findContours().

A close-up of a computer code

Description automatically generated

The for loop iterates through each contour, filling them with white color on the original grayscale image, drawing a filled contour (solid shape) on the grayscale image, using white color to fill in the star regions.

A computer code with text

Description automatically generated

A final thresholding step inverts the grayscale image colors by using cv.threshold() with cv.THRESH\_BINARY\_INV as type. With this transformation, the previously white background becomes black, and the stars, originally filled with white, are now rendered in contrasting black, achieving the desired prominence. The final image, showcasing the extracted black stars against a clean white background, is then saved to the result image by cv.imwrite().

A close-up of a number

Description automatically generated

* 1. Second Task
     1. Introduction

The ability to automatically detect an element within an image is a crucial task with a large number of applications. In task 2, the main purpose is to detect digits in an image by drawing rectangles around them.

By using OpenCV and NumPy libraries, we can use thresholding, morphological operations, and contour detection, which are built-in function of those libraries. We can develop an algorithm which can resolve this task.

* + 1. Image Loading and Color Conversion

As we do in the task 1, we use cv.imread() twice to load the image input2.png into variables as normal color and grayscale color image, ready for manipulation.

A close-up of text

Description automatically generated

* + 1. Defining Regions of Interest

Because of complexity of some digits in the image, so we must define the regions of interest (ROIs) manually.

To handle this task, we specify the coordinates of bounding boxes that encapsulate each digit within the image.

A screenshot of a computer code

Description automatically generated

* + 1. Function to Edit the Image and Draw Rectangles

We use cv.adaptiveThreshold() function to apply the adaptive thresholding technique on the gray image. Adaptive thresholding is a method commonly employed to segment images into distinct regions based on variations in pixel intensity.

The cv.findContours() function will detect all connected regions within the thresholded image and return a list of contours (connected regions). We use cv.RETR\_EXTERNAL flag to retrive only the external contours, and exclude interal contours or holes within objects. The cv.CHAIN\_APPROX\_SIMPLE flag approximate the contours’s shapes to optimize the memory consumption, and reduce the complexity.

A close-up of a computer code

Description automatically generated

* + 1. Processing Each Region of Interest

We initial 2 variables which are minContourArea, and maxContourArea serve as thresholds to outline the acceptable range of contour areas. Contours without this range will be considered irrelevant and ignored in further steps.

The loop iterates to detect all contour within the image. The area enclosed of each contour is calculated by the cv.contourArea() function, and effectively quantifying the extent or size of the object represented by the contour.

A conditional statement is employed to evaluate whether the area falls in the pre-defined range by 2 of variables, minContourArea and maxContourArea.

Bounding rectangles are handled by cv.boundingRect() function. These rectangles tightly enclose the contours, providing a visual representation of the detected objects's spatial extent and orientation within the image. The coordinates of the top-left corner and the width and height of each rectangle are determined based on the contour's properties.

Green rectangles are drawn around the detected contours on the original image using the cv.rectangle() function.

A computer code with text

Description automatically generated

The detect\_and\_draw\_contours() function arranges a sequence of image processing to refine contour visibility, and present the objects within the image.

A screenshot of a computer code

Description automatically generated

* + - 1. Kernel Initialization

This initial stage requires the creation of a square-shaped kernel utilizing NumPy's np.ones() function. The dimensions of the kernel are dictated by the kernels parameter, outline its height and width.

* + - 1. Closing Operation

Subsequently, the cv.morphologyEx() function arranges a morphological closing operation on the input image. Morphological closing, a procedural amalgamation of dilation and erosion, endeavors to bridge minute gaps and refine the contours of objects present in the image.

* + - 1. Conversion to Grayscale

Following the closure operation, the resultant image undergoes conversion into grayscale via the cv.cvtColor() function. This conversion serves to streamline subsequent thresholding processes and mitigate computational intricacies.

* + - 1. Thresholding

The grayscale image is subjected to binary thresholding utilizing the cv.threshold() function. This transformation facilitates the conversion of the grayscale image into a binary counterpart, wherein pixels surpassing a specified intensity threshold are designated white, while those beneath the threshold are rendered black.

* + - 1. Erosion

Thereafter, the binary thresholded image undergoes an erosion operation employing the cv.erode() function. Erosion is instrumental in diminishing the size of foreground elements (white regions) within the binary image, thereby fostering the segregation of closely positioned objects and refining their boundaries.

* + - 1. Contour Detection

Contours are discerned within the eroded image via the cv.findContours() function. These contours, representing the perimeters of objects within the image, constitute an indispensable facet of object identification and delineation endeavors.

* + - 1. Contour Filtering and Rectangle Drawing

Within the iterative loop traversing the detected contours, each contour's area is calculated utilizing the cv.contourArea() function. Contours whose areas reside within a pre-defined range are deemed pertinent.

For qualifying contours, bounding rectangles are computed employing the cv.boundingRect() function. These rectangles enshroud the contours snugly, facilitating the visualization and analytical scrutiny of detected objects.

The delineated contours are accentuated by the drawing of green rectangles encompassing them on the original image via the cv.rectangle() function, thereby enriching the visual representation of identified objects.

* + - 1. Image Return

Finally, the function culminates in the provision of the modified image embellished with delineated rectangles, proffering it for subsequent processing or visual depiction.

* + 1. Saving the Output Image

The code snippet above demonstrates the invocation of the detect\_and\_draw\_contours function for each specified region of interest. Each function call includes distinct parameters tailored to the characteristics of the respective regions. These parameters dictate the behavior of contour detection and rectangle drawing within the function.

After processing all regions of interest, the modified image with drawn rectangles is saved to the specified file path using the cv.imwrite() function.

A screenshot of a computer program

Description automatically generated

CHAPTER 2 – TASK RESULTS

* 1. First Task
     1. Extract each star in the input image automatically.

A blue star on a black background

Description automatically generated

Figure 1.3.1: Blue star

A green star on a black background

Description automatically generated

Figure 1.3.2: Green star

A star in the sky

Description automatically generated

Figure 1.3.3: Orange star

A purple star on a black background

Description automatically generated

Figure 1.3.4: Purple star

A pink star on a black background

Description automatically generated

Figure 1.3.5: Red star

A yellow star on a black background

Description automatically generated

Figure 1.3.6: Yellow star

* + 1. Repaint all stars.

A group of stars on a white background

Description automatically generated

Figure 1.3.2: Repaint all stars.

* 1. Second Task
     1. Draw rectangles surrounding each digit in the input image automatically.

A close-up of numbers

Description automatically generated

Figure 1.4.1: Digit rectangles detection