

**Báo bài tập lớn xử lý ảnh**



**Sinh viên: Đào Văn Cường**

**Giáo viên hướng dẫn: thầy Phúc.**

Hà Nội

# **Lời nói đầu**

Ngày nay, công nghệ ngày càng phát triển, những công nghệ mới này giúp ích rất nhiều trong đời sống, công việc, học tập và nâng cao năng suất lao động. Một trong những công nghệ không thể thiếu là xử lý ảnh. Xử lý ảnh ứng dụng rất nhiều trong đời sống con người như: Giúp những bức ảnh chụp đẹp hơn, sống động hơn, ứng dụng trong nền nông nghiệp như tưới nước, diệt sâu bọ…, ứng dụng trong công nghiệp nhưng như robot tự động nâng cao năng suất giảm bớt lao động chân tay, ứng dụng trong giao thông như xe tự lái…

Trong những ứng dụng phổ biến đấy, ứng dụng nhận diện biển số xe cũng không kém phần quan trọng trong việc nhận diện biển số, quản lý xe trong các gara, bãi đỗ xe…

Với mục tiêu có thêm những kiến thức nền tảng trong xử lý ảnh, em chọn đề tài “Nhận diện biển số xe”.

Dù đã cố gắng tìm hiểu, nhưng do kiến thức có hạn nên bài báo cáo và phần demo còn nhiều thiếu sót. Kính mong nhận được những ý kiến từ thầy vào các bạn để em hoàn thiện hơn.

Em xin gửi lời cảm ơn thầy Phúc đã dạy một cách nhiệt tình để em có những kiến thức hoàn thiện bài tập lần này.

Em xin chân thành cảm ơn!

Hà Nội, ngày 17 tháng 10 năm 2018

Mục Lục

[**Lời nói đầu** 1](#_Toc528012444)

[**Chương 1: Tổng quan về hệ thống quản lý biển số** 4](#_Toc528012445)

[**Chương 2: Quá trình nhận diện biển số** 8](#_Toc528012446)

[**1.** **Tiền xử lý.** 8](#_Toc528012447)

[**2.** **Lấy những điểm ảnh trắng** 9](#_Toc528012448)

[**3.** **Lọc** 10](#_Toc528012449)

[**4.** **Dùng dilate để tăng sharp.** 11](#_Toc528012450)

[**5.** **Nhận dạng** 11](#_Toc528012451)

[**6.** **Tách lấy số** 12](#_Toc528012452)

[**Chương 3: Một số chức năng khác** 12](#_Toc528012453)

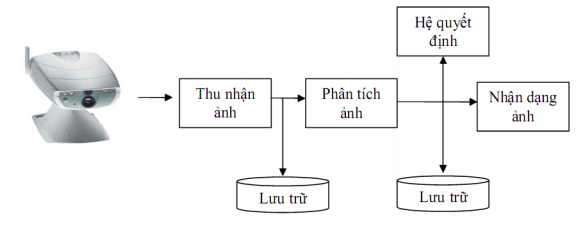
[**Lời kết** 15](#_Toc528012454)

# 

# **Chương 1: Tổng quan về hệ thống quản lý biển số**

1. **Tổng quan về xử lý ảnh số.**

Ngày nay kỹ thuật xử lí ảnh đã được ứng dụng rộng rãi ở rất nhiều lĩnh vực, trong sản xuất cũng như trong đời sống. Ví du các hệ thống xử lí ảnh vệ tinh để phân tích không gian vũ trụ, hệ thống thăm dò địa chất, hệ thống phân tích tế bào sinh học và gần gũi nhất với chúng ta là các phần mềm hiển thị và xử lí ảnh chuyên dụng như Photoshop, ACD See… Một hệ thống xử lý ảnh là hệ thống thực hiện các chức năng thu nhận ảnh đầu vào, thực hiện phép xử lý để tạo ảnh hoặc kết quả phân tích, nhận dạng ở đầu ra đáp ứng các yêu cầu và các ứng dụng cụ thể. Trong phạm vi đồ án, tôi xin giới hạn trong việc giới thiệu một hệ thống xử lý ảnh ứng dụng nhận dạng và ra quyết định trên thực tế.



Sơ đồ khối tổng quát của hệ thống này được cho trong hình 1.1, trong đó gồm ba khối chức năng cơ bản: • Khối thu nhận ảnh: thực hiện chức năng thu nhận ảnh và thực hiện quá trình số hóa (lưu giữ theo định dạng yêu cầu). • Khối phân tích ảnh: trước hết hệ thống tiến hành bước tiền xử lý ảnh với mục đích tăng cường, cải thiện chất lượng ảnh, làm nổi các đặc trưng cơ bản của ảnh hay làm cho ảnh gần giống nhất với trạng thái gốc. Sau đó, là quá trình

phân tích ảnh và trích chọn đặc trưng của ảnh ví dụ như biên, điểm gấp khúc, điểm kết thúc, điểm chữ thập… • Khối nhận dạng: dựa vào các đặc trưng đã thu nhận từ quá trình phân tích ảnh trước đó thực hiện quá trình nhận dạng, đưa ra các quyết định ứng với các ứng dụng cụ thể.

1. Một số khái niệm và các vấn đề cơ bản trong xử lý ảnh số.

* Một số khái niệm.
* Phần tử ảnh (Pixel – Picture Element)

Ảnh trong thực tế là một ảnh liên tục về không gian và giá trị độ sáng. Để có thể xử lý ảnh bằng máy tính cần thiết phải số hóa ảnh. Trong quá trình số hóa, người ta biến đổi tín hiệu liên tục sang tín hiệu rời rạc thông qua quá trình lấy mẫu (rời rạc hóa về không gian) và lượng hóa thành phần giá trị. Trong quá trình này người ta sử dụng khái niệm Pixel để biểu diễn các phần tử của bức ảnh. Ở đây, cũng cần phân biệt khái niệm pixel hay đề cập đến trong các hệ thống đồ họa máy tính. Để tránh nhầm lẫn ta tạm thời gọi khái niệm pixel này là pixel thiết bị.

* Ảnh màu (Color Image)

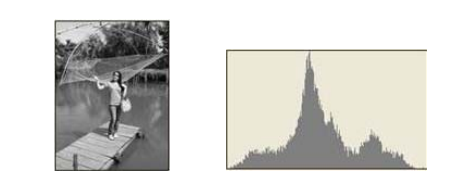
Ảnh màu chứa thông tin màu cho mỗi phần tử ảnh. Thông thường giá trị màu này dựa trên các không gian màu (color space) trong đó không gian màu thườngđược dùng là RGB tương ứng với ba kênh màu đỏ (Red) – xanh lá cây (Green) – xanh da trời (Blue). Tùy thuộc vào số bit, được sử dụng để lưư trữ màu ta có số lượng màu khác nhau, ví dụ 8 bit, 16 bit, 24 bit (True Color). Nếu ta sử dụng nhỏ hơn 24 bit để lưu trữ màu thì ta phải có 1 bảng Palette màu, nó tương tự như một bảng Lookup Table cho phép ánh xạ giữa một vị trí trong bảng với một tổ hợp của không gian màu RGB. Ví dụ như sử dụng 8 bit tương ứng với 256 màu thì ta phải có bảng ánh xạ 256 màu đó tương ứng với 256 tổ hợp Red – Green – Blue.

* Ảnh đa mức xám

Mức xám (Gray level) là kết quả sự mã hóa tương ứng một cường độ sáng của mỗi điểm ảnh với một giá trị số - kết quả của quá trình lượng tử hóa. Ảnh đa mức xám là ảnh có sự chuyển dần mức xám từ trắng sang đen. Thực tế, một giá trị mức xám chính là sự tổ hợp của ba giá trị RGB (Red- Green – Blue). Thông thường mỗi điểm ảnh trong bức ảnh đa xám thường được mã hóa 8 bit, tương ứng với 256 mức xám.

* Lược đồ mức xám

Lược đồ này (Hình 1.2) đơn giản cho ta biết tần suất xuất hiện của mỗi điểm ảnh ( pixel ) trong một bức ảnh ứng với mức xám tương ứng. Ví dụ, một ảnh đa mức xám sử dụng 8 bit, có 256 mức xám từ o tới 255. Lược đồ mức xám sẽ có trục hoành chạy từ 0 tới 255 và trục tung chính là tổng số điểm ảnh có mức xám tương ứng. Biểu đồ này tuy đơn giản nhưng có nhiều ứng dụng trong các bài toán giãn độ tương phản và phân ngưỡng ảnh ( biến từ ảnh mức xám sang ảnh nhị phân).



* Ảnh nhị phân

Ảnh nhị phân là ảnh mà mỗi điểm ảnh ảnh chỉ nhận một trong hai giá trị nhị phân.

* Biểu diễn ảnh

Trong biểu diễn ảnh, người ta thường dùng các phần tử đặc trưng của ảnh là pixel. Nhìn chung có thể xem một hàm hai biến chứa thông tin như biểu diễn một ảnh.

Một số mô hình thường được dùng để biểu diễn ảnh như: mô hình toán học, mô hình thống kê,v.v....

* Tăng cường ảnh - Khôi phục ảnh.

Tăng cường ảnh là bước quan trọng, tạo tiền đề cho xử lý ảnh. Nó gồm một loạt các kỹ thuật như: lọc độ tương phản, khử nhiễu, nổi màu… Khôi phục ảnh nhằm loại bỏ các suy giảm trong ảnh.

* Biến đổi ảnh.

Thuật ngữ biến đổi ảnh thường được dùng để nói tới một lớp các ma trận đơn vị và các kỹ thuật dùng để biến đổi ảnh. Cũng như các tín hiệu một chiều được biểu diễn bởi một chuỗi các hàm cơ sở. Có nhiều loại biến đổi được dùng như

• Biến đổi Fourier [1]

• Biến đổi KL (Karhuman Loeve )… [2]

* Phân tích ảnh.

Phân tích ảnh liên quan đến việc xác định các độ đo định lượng của ảnh để đưa ra một mô tả đầy đủ về ảnh. Một số kỹ thuật hay dùng là dò biên, dán nhãn vùng liên thông, phân vùng ảnh…

* Nhận dạng ảnh.

Nhận dạng là quá trình liên quan đến các mô tả đối tượng mà người ta muốn đặc tả nó. Quá trình nhận dạng thường đi sau quá trình trích chọn các đặc tính chủ yếu của đối tượng. Có hai kiểu mô tả đối tượng • Mô tả tham số (Nhận dạng theo tham số) [3] • Mô tả theo cấu trúc (Nhận dạng theo cấu trúc) [4]

# **Chương 2: Quá trình nhận diện biển số**

Tài nguyên: Một số ảnh lấy trên mạng chụp mặt trước của xe. Môt số ảnh chụp nghiêng nên nhận dạng không chính xác.

Ngôn ngữ lập trình: python sử dụng thư viện opencv và numpy.

**Các bước nhận dạng:**

1. **Tiền xử lý.**

* Load image sử dụng imread

Mục tiêu: Chuyển từ ảnh không gian 3 chiều về ảnh không gian một chiều để dễ dàng xử lý.

ở đây chúng ta sẽ chuyển từ không gian 3 chiều RGB sang không gian 1 chiều gray.

Một trong các công thức phổ biến đó là:

**Y = 0.2126R + 0.7152G + 0.0722B**

Trong đó:

* Y: ma trận xám cần tìm
* R: ma trận xám đỏ của ảnh
* G: ma trận xám lục của ảnh
* B: ma trận xám lam của ảnh

def grayImage(self):

self.image=cv.imread(self.in\_path)

self.grayscaled = cv.cvtColor(self.image,cv.COLOR\_BGR2GRAY)

if self.in\_path!="":

minc = Image.fromarray(self.grayscaled)

mincol = ImageTk.PhotoImage(minc)

label3 = Label(self.parent, image=mincol)

label3.image = mincol

label3.place(x=5, y=5)



Ảnh trước và sau khi xử lý

* Tiếp theo chúng ta sẽ dãn độ tương phản.

def morphologyEx(self):

kernel = np.ones((5,10),np.uint8)

self.closing = cv.morphologyEx(self.grayscaled, cv.MORPH\_BLACKHAT, kernel)

if self.in\_path!="":

minc = Image.fromarray(self.closing)

mincol = ImageTk.PhotoImage(minc)

label3 = Label(self.parent, image=mincol)

label3.image = mincol

label3.place(x=5, y=5)



1. **Lấy những điểm ảnh trắng**

Ảnh đen trắng, như tên gọi, chỉ có hai màu đen với giá trị là 0 và trắng với giá trị 255. Bài toán đặt ra là làm thế nào để biến đổi ảnh xám (hay nói cách khác là một ma trận 2 chiều với giá trị mỗi ô trong khoảng 0-255) về thành một ma trận 2 chiều với giá trị mỗi ô là 0 hoặc 255.

Cách giải quyết là chọn một ngưỡng (threshold) để xác định đâu là điểm ảnh đen và đâu là điểm ảnh trắng. Nếu giá trị trên ảnh xám lớn hơn ngưỡng threshold, đấy là điểm ảnh trắng và ngược lại.

**Gọi ảnh xám là Gray, ảnh đen trắng cần xác định là BW, tọa độ các pixel trên hình là (x,y), ta có  
- BW(x,y) = 255 nếu Gray(x,y) > threshold  
- BW(x,y) = 0 nếu Gray(x,y) <= threshold**

Ở trong bài toán này, chúng ta lấy ngưỡng là 100 để lấy được những vùng màu trắng đặc biệt là biển số.

def normalize(self):

# cv.normalize(self.closing,self.closing,0,255,cv.NORM\_MINMAX)

# cv.imshow("closing",self.closing)

retval,self.threshold=cv.threshold(self.closing,100,255,cv.THRESH\_BINARY)

cv.imshow('normalize',self.threshold)



1. **Lọc**

Tiếp theo, làm thế nào để lấy được vùng là biển sổ. Như chúng ta đã thấy trong hình, vùng biển số mật độ điểm màu trắng khá nhiều.

Thế nên, điều cần làm là lọc bớt những vùng có ít mật độ điểm trắng.

Bằng cách đếm những pixel trắng trong vùng chữ nhật mà chúng ta chọn. Ở đây chọn vùng 16 \* 8. Cho vùng này chạy hết ảnh. Nếu có những vùng nhỏ hơn số pixel trắng mà chúng ta đặt trước thì chúng ta sẽ set những vùng này về 0 tức là màu đen.

def normalize(self):

# cv.normalize(self.closing,self.closing,0,255,cv.NORM\_MINMAX)

# cv.imshow("closing",self.closing)

retval,self.threshold=cv.threshold(self.closing,100,255,cv.THRESH\_BINARY)

cv.imshow('normalize',self.threshold)



Ảnh sau khi đã lọc

1. **Dùng dilate để tăng sharp.**

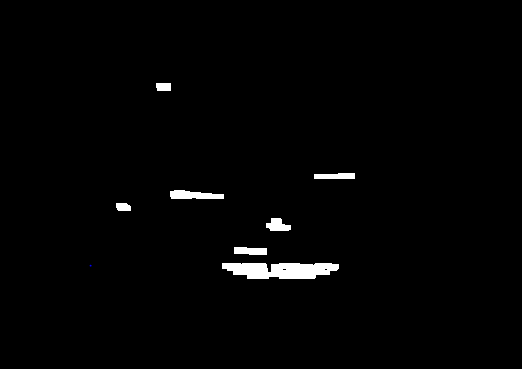
Tiếp theo, chúng ta sẽ lấy vùng biển số bằng cách dãn nở những điểm ảnh màu trắng có giá trị 255 để nối những điểm ảnh màu trắng với nhau. Sau đó chúng ta sẽ có được những vùng có thể là biển số.

def lam\_nhoe(self):

kernel = np.ones((5,10),np.uint8)

self.image2=cv.dilate(self.threshold,kernel,2)

cv.imshow("image2",self.image2)



Hình sau khi đã được xử lý.

Ở hình trên chúng ta tìm được 7 vùng có thể là biển số.

1. **Nhận dạng**

Sau đây, chúng ta sẽ nhận dạng những vùng này bằng cách xác định chiều dài chiều rộng của vùng đấy xem có đúng là biến số hay không.

Trước tiên, chúng ta phải tìm được ra những vùng đó bằng cách tìm những vùng là màu trắng. sử dụng hàm findContours cho cả image đó chúng ta sẽ tìm ra được 7 vùng là biển số.

Chúng ta sẽ duyệt lần lượt 7 contours này. Nếu đúng trường hợp cần xét chúng ta sẽ dừng lại.

def nhanDang(self):

w,h,chanel=self.image.shape

kernel = np.ones((5,10),np.uint8)

size = w, h, chanel

im2, contours, hierarchy = cv.findContours(self.image2,cv.RETR\_LIST,cv.CHAIN\_APPROX\_SIMPLE)

for i in range(0,len(contours)):

x,y,w,h = cv.boundingRect(contours[i])

if float(w/h)>2 and float(h/w)<4:

s=w\*h

if(w>80 and w<250 and h>15 and h<40):

cv.rectangle(self.image,(x,y),(x+w,y+h+5),(0,255,0),1)

self.imgBienSo=self.image[y:y+h+5,x:x+w]

# cv.imshow('image',self.image)

cv.imshow("imgBienSo",self.imgBienSo)



ảnh đã nhận dạng được

1. **Tách lấy số**

Cuối cùng là tìm những phần là số ở ảnh chúng ta tìm được.

Chúng ta tiếp tục lấy biên của những số đã tìm được.

Lấy ngưỡng của ảnh để thu về được những số.

Cuối cùng là findcontours để lấy ra được những số này.

def tachBienSo(self):

grayscaled = cv.cvtColor(self.imgBienSo,cv.COLOR\_BGR2GRAY)

kernel = np.ones((5,10),np.uint8)

w,h,chanel=self.imgBienSo.shape

size = w, h, chanel

closing = cv.morphologyEx(grayscaled, cv.MORPH\_BLACKHAT, kernel)

cv.normalize(self.closing,self.closing,0,255,cv.NORM\_MINMAX)

retval,th=cv.threshold(closing,100,255,cv.THRESH\_BINARY)

cv.imshow('th',th)

# cv.imshow("image bien so grayscaled",grayscaled)

im2, contours, hierarchy = cv.findContours(th,cv.RETR\_LIST,cv.CHAIN\_APPROX\_NONE)

print(contours)

for i in range(0,len(contours)):

x,y,w,h = cv.boundingRect(contours[i])

if h>w:

if(h>5 and w>5):

img=[]

img=th[y:y+h,x:x+w]

cv.imshow('imso'+str(i),img)

**Chương 3: Một số chức năng khác**

Sau đây em sẽ trình bay một số chức năng khác của xử lý ảnh như:

* Add

Ở đây thực chất là công 2 ma trận với nhau, điều kiện là ảnh phải cũng kích thước như thế thì 2 ma trận sẽ cùng kích thước và chúng ta sẽ cộng 2 ảnh được với nhau.

**Python:**cv.**Add**(src1, src2, dst, mask=None) → None

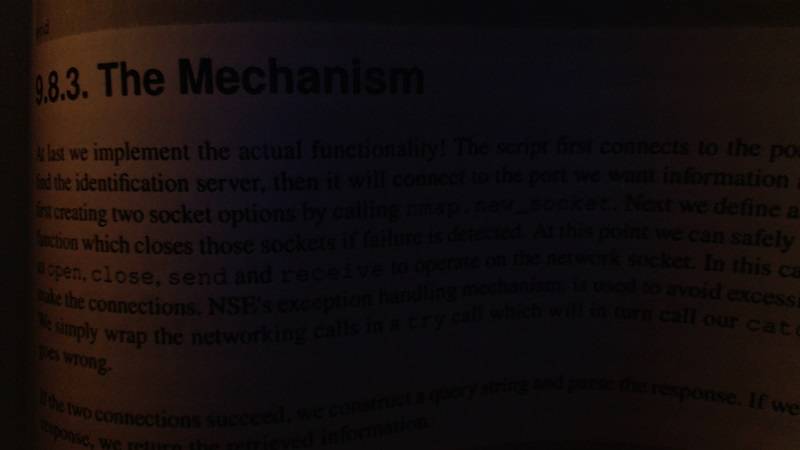
Các thông số như:

* src1 – mảng đầu tiên
* src2 – mảng thứ 2
* dst – mảng đầu ra

Tổng của hai mảng khi cả hai mảng đầu vào có cùng kích thước và cùng một số kênh:

\texttt{dst}(I) =  \texttt{saturate} ( \texttt{src1}(I) +  \texttt{src2}(I)) \quad \texttt{if mask}(I) \ne0

Ví dụ: dst = sc1 + sc2.



Ảnh chưa xử lý



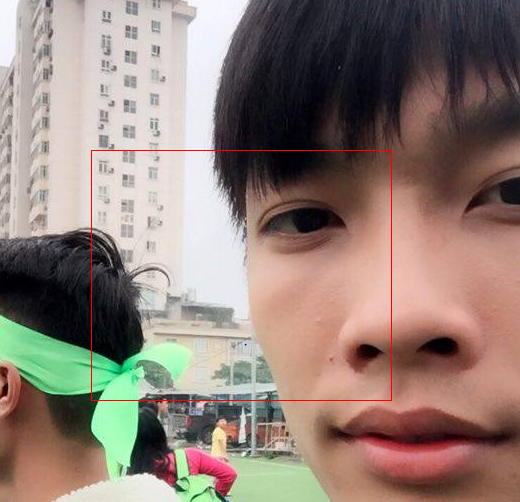
ảnh sau khi đã xử lý

* Vẽ rectangle

cv2.**rectangle**(img, pt1, pt2, color[, thickness[, lineType[, shift]]])

Các thông số:

* Img – image
* Ptl – Vị trí điểm vẽ
* Pt2 – số pixel sẽ vẽ
* Color - màu



**Lời kết**

Sau một số bước thì chúng ta đã lấy được ảnh của những số trong biển số.

Vì còn làm một số bước thô sơ và ảnh không chính xác nên chúng ta thu về một số kết quả sai.

Chúng ta cần có thời gian phân tích biển số, trích chọn hợp lý và có những ảnh có chất lượng cao hơn để xử lý một cách chính xác hơn.

Xin cảm ơn thầy đã dạy nhiệt tình và rất thú vị để em có thể hoàn thành tốt bài tập lớn lần này.