**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**



**BÁO CÁO**

**ĐỒ ÁN LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG**

**Đề tài:**

**XÂY DỰNG ỨNG DỤNG OPENCV KẾT HỢP VỚI REACTNATIVE**

GVHD: Th.s Huỳnh Tuấn Anh

Sinh viên thực hiện:

Nguyễn Quý Năng 14520568

TP. Hồ Chí Minh, ngày 19 tháng 12 năm 2018

# **Lời Cảm Ơn**

Đầu tiên, nhóm chúng em xin gởi lời cảm ơn chân thành đến tập thể quý Thầy Cô Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM và quý Thầy Cô khoa Công Nghệ Phần Mềm đã giúp cho nhóm chúng em có những kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Trong thời gian một học kỳ thực hiện đề tài, nhóm chúng em đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới. Từ đó, nhóm chúng em vận dụng tối đa những gì đã thu thập được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, trong quá trình thực hiện, nhóm chúng em không tránh khỏi những thiếu sót. Chính vì vậy, nhóm chúng em rất mong nhận được những sự góp ý từ phía các Thầy Cô nhằm hoàn thiện những kiến thức mà nhóm chúng em đã học tập và là hành trang để nhóm chúng em thực hiện tiếp các đề tài khác trong tương lai.

Nhóm em xin chân thành cảm ơn quý Thầy Cô!

# **NHẬN XÉT CỦA GIẢNG VIÊN**

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

………………………………………………………………………………………………..........................................................................................................................................................................................

..............................................................................................................................................

**Mục lục**

[**Lời Cảm Ơn** 2](#_Toc503942144)

[**NHẬN XÉT CỦA GIẢNG VIÊN** 4](#_Toc503942146)

[CHƯƠNG 1: TỔNG QUAN 7](#_Toc503942147)

[1.1. Giới thiệu đề tài 7](#_Toc503942148)

[1.1.1. Mô tả đề tài 7](#_Toc503942149)

[1.1.2. Môi trường cài đặt 7](#_Toc503942150)

[1.2. Mục tiêu cần đạt được 7](#_Toc503942151)

[CHƯƠNG 2: XÂY DƯNG ỨNG DỤNG NHẬN DIÊN MÃ CARD ĐIỆN THOẠI BẰNG OPENCV VÀ MÔ HÌNH SVM 16](#_Toc503942160)

[**3.1 Sơ đồ thực hiện** 16](#_Toc503942161)

[**3.2 Dữ liệu huấn luyện.** 17](#_Toc503942162)

[**3.3 Tiền xử lý ảnh** 19](#_Toc503942163)

[**3.4 Xác định ký tự** 23](#_Toc503942164)

[**3.5 Phân tách ký tự** 24](#_Toc503942165)

[**3.6 Nhận dạng ký tự bằng SVM** 25](#_Toc503942166)

[3.6.1 Huấn luyện dữ liệu 25](#_Toc503942167)

[3.6.2 Trích chọn đặc trưng 25](#_Toc503942168)

[CHƯƠNG 4: HOÀN THÀNH ỨNG DỤNG 27](#_Toc503942169)

[**4.1 Giao diện của phần mềm** 27](#_Toc503942170)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 28](#_Toc503942171)

[**5.1 Kết luận** 28](#_Toc503942172)

[**5.2 Ưu nhược điểm của phần mềm** 29](#_Toc503942173)

[5.2.1 Ưu điểm. 29](#_Toc503942174)

[5.2.2 Nhược điểm. 29](#_Toc503942175)

[**5.3 Hướng phát triển** 29](#_Toc503942176)

[CHƯƠNG 6 : PHỤ LỤC 29](#_Toc503942177)

[1. Cài đặt Opencv vào android studio để lập trình : 29](#_Toc503942178)

[2. Code training dữ liệu mô hình SVM bằng C++ Visual Studio: 33](#_Toc503942179)

# CHƯƠNG 1: TỔNG QUAN

## Giới thiệu đề tài

### Mô tả đề tài

Ngày nay, smartphone đã phát triển vượt bậc và xuất hiện rộng rãi trong xã hội chúng ta, kéo theo đó là nhu cầu nghe, gọi, nhắn tin, liên lạc của mọi người ngày càng tăng. Để thực hiện điều đó, hầu hết mọi người phải mua card điện thoại và nhập 12-13 con số để nạp tiền vào trong điện thoại. Nắm bắt được vấn đề đó, nhóm đã làm một ứng dụng quét mã số thẻ card trên điện thoại để người dùng thuận tiện hơn trong việc nạp card. Ứng dụng giúp người dùng có thể nạp tiền nhanh hơn, và chính xác hơn rất nhiều. Điều đó mang lại cho mọi người sự thoải mái, tiết kiệm thời gian cũng như công sức.

Để làm được điều đó, nhóm đã thao khảo một số thư viện mã nguồn mở để giúp cho việc nhận diện các chữ số mã thẻ, và chọn OpenCV làm mã nguồn mở chính thức. Điều đó sẽ giúp cho chức năng nhận diện chính xác và linh hoạt hơn.

### Môi trường cài đặt

* Điện thoại Android có camera.

## Mục tiêu cần đạt được

Ứng dụng nạp card đáp ứng nhu cầu nạp tiền vào điện thoại của người dùng và thỏa mãn các chức năng sau:

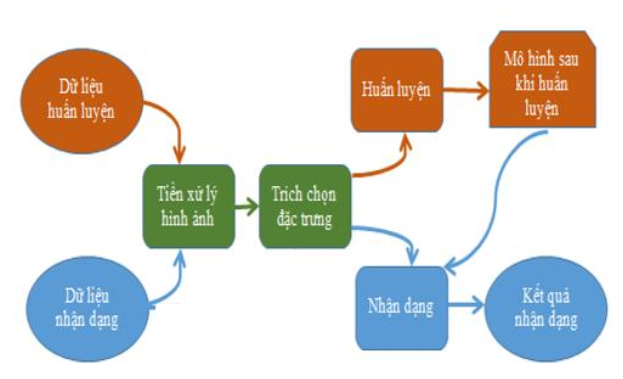
* Quét mã số thẻ cào
* Tự động thực hiện cuộc gọi để nạp tiền
* Chia sẻ mã số thẻ cho người thân, bạn bè

# CHƯƠNG 2: XÂY DƯNG ỨNG DỤNG NHẬN DIÊN MÃ CARD ĐIỆN THOẠI BẰNG OPENCV VÀ MÔ HÌNH SVM

Trong vấn đề nhận dạng ký tự số bằng thuật toán SVM ta có những bƣớc cơ bản  
như sau: tiền xử lý hình ảnh, phân tách ký tự, huấn luyện dữ liệu sau đó đưa dữ liệu  
huấn luyện và dữ liệu cần nhận dạng vào thuật toán SVM để nhận dạng. Bước trướ sẽ  
làm tiền để để thực hiện bƣớc sau, kết quả của các bước cũng sẽ phụ thuộc lẫn nhau. Ta  
sẽ lần lượt thực hiện từng bước, từng quá trình thì bài toán sẽ trở nên đơn giản và dễ  
hiểu hơn rất nhiều.

## **3.1 Sơ đồ thực hiện**

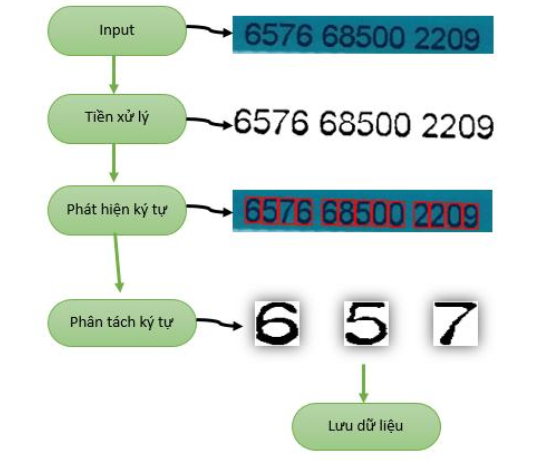
Một quá trình nhận dạng thì gồm 3 bước cơ bản: chuẩn bị bộ dữ liệu, huấn luyện  
và nhận dạng. Điều đó sẽ đƣợc thể hiện rõ trong hình 2.1.  
Đầu vào: Các yêu cầu dữ liệu đầu vào là các file ảnh đuọc chụp lại từ card điện  
thoại (quét bằng Camera của các thiết bị Android). Để việc tiến hành nhận dạng đƣợc  
chính xác, các file ảnh này đã qua giai đoạn tiền xử lý : xử lý ảnh nhiễu, ảnh đa sắc, nhị  
phân hóa, phân tách ra từng ký tự,...  
Đầu ra mong muốn: là một file text chứa các ký tự số tương ứng mà đầu vào đưa  
vào.  
Sơ đồ thực hiện:



***Hình 3.1*** *: Quy trình nhận dạng số*

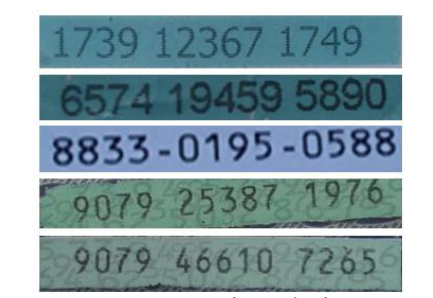
## **3.2 Dữ liệu huấn luyện.**

Dữ liệu huấn luyện là bộ cơ sở dữ liệu ta đưa vào trong quá trình huấn luyện, nó  
gồm các ký tự số từ 0 tới 9. Để xây dựng được bộ dữ liệu huấn luyện sao cho chính xác  
và phù hợp với bài toán của ta yêu cầu thì ta phải trải qua các bước xử lý hình ảnh: tiền  
xử lý ảnh, phát hiện ký tự và phân tách ký tự.



***Hình 3.2****: Quy trình chuẩn bị dữ liệu huấn luyện*

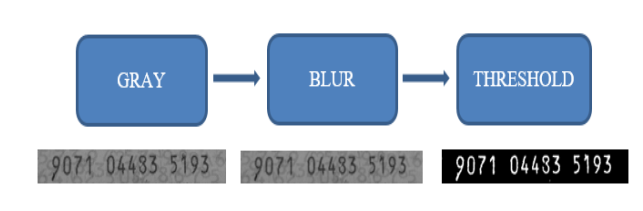
Dữ liệu huấn luyện của ta đó là các ký tự số từ 0 tới 9, để có được bộ dữ liệu đó  
ta phải tiến hành quá trình thu thập dữ liệu. Cụ thể đối với bài toán này ta cần phải sưu  
tầm các chữ số trong các loại card, trong bước này ta có thể chụp hình lại cái card chứa  
số, sau đó cắt thủ công vùng chứa số hoặc cho quét trên thiết bị Android sau đó lưu  
xuống trong bộ nhớ thiết bị.  
dưới đây là một số hình ảnh card dùng để làm dữ liệu huấn luyện. Khi thu thập  
dữ liệu ta cần lấy nhiều mẫu ký tự số phong phú, từ phông nền khác nhau, ký tự số của  
các nhà mạng khác nhau như Viettel, Mobile Phone, Vina Phone,… các nét ký tự số  
khác nhau và các ký tự số cũng khác nhau. Ví dụ trong trường hợp số 1 có gạch chân và  
số 1 không có gạch chân, số 4 kín và số 4 hở,... Việc thu thập dữ liệu càng đa dạng thì  
bài toán nhận dạng sẽ có kết quả càng tốt.



***Hình 3.3****: Hình ảnh đầu vào để huấn luyện*

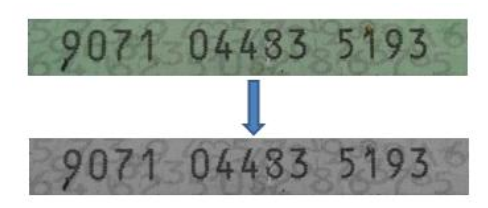
## **3.3 Tiền xử lý ảnh**

Để phát hiện được ký tự và phân tách ra ta phải thực hiện quá trình tiền xử lý:  
chuyển thành hình xám (Gray), làm mờ (GaussianBlur), nhị phân hóa (Threshold).  
Ngoài ra ta có thể sử dụng thêm nhiều kỹ thuật xử lý hình ảnh khác để giúp quá trình  
tiền xử lý hình ảnh có kết quả chạy tốt hơn.



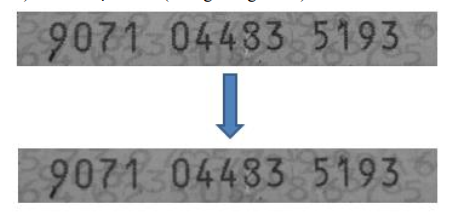
***Hình 3.4*** *: Quy trình tiền xử lý ảnh*

Hình ảnh của ta ban đầu đưa vào là hình ảnh có hệ màu RGB là hệ màu có sự kết  
hợp của 3 màu cơ bản đỏ (Red), xanh lục (Green), xanh lơ (Blue) được biểu diễn giá trị  
từ 0 đến 255. Ta cần phải chuyển hình ảnh về ảnh xám (Gray) để tiện lợi cho quá trình  
phát hiện được vùng chứa số. Ảnh Gray là ảnh chỉ được biểu diễn các giá trị nằm trong  
khoảng [0, 255]. Trong OpenCV hỗ trợ chuyển đổi không gian màu bằng hàm:  
***cvtColor*** *(imageSrc, imageDst, code)*Trong đó: imageSrc là ảnh gốc, imageDst là ảnh sau khi chuyển màu, code là mã  
chuyển đổi không gian màu, cụ thể trong bài toán của ta sẽ sử dụng không gian màu  
xám trong OpenCV: BGRA2GRAY



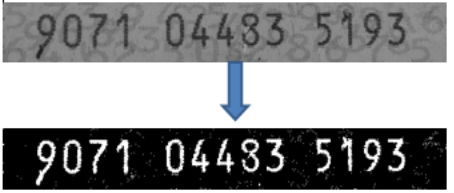
***Hình 3.5*** *: Chuyển xám*

Sau khi nhị chuyễn xám hình ảnh ta tiếp tục dùng hàm GausianBlur.  
GaussianBlur là cách làm mờ một ảnh bằng hàm Gaussian. Phƣơng pháp này được ứng  
dụng một cách rộng rãi và hiệu quả trong các phần mềm xử lý đồ họa. Nó cũng là công  
cụ phổ biến để thực hiện quá trình tiền xử lý (preprocessing) hình ảnh dùng làm dữ liệu  
đầu vào tốt cho các phân tích cao cấp hơn nhƣ trong Computer Vision, hoặc cho các  
giải thuật được thực hiện trong một tỉ lệ khác của hình được cho. Nó có thể giúp làm  
giảm nhiễu (Noise) và mức độ chi tiết (không mong muốn) của hình ảnh.



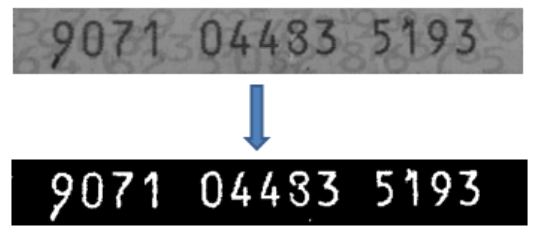
***Hình 3.6*** *: Gaussian Blur*

Quá trình làm mờ hình ảnh rất quan trọng, nó quyết định rất nhiều vào kết quả  
cuối cùng của bài toán. Ta có thể xem hình ảnh 2.7 để thấy được độ quan trọng của  
hàm này trong bài toán.



***Hình 3.7:*** *Tiền xử lý ảnh không làm mờ*

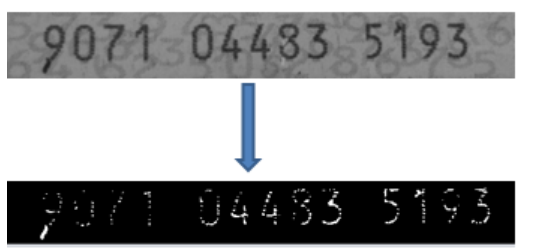
Đến bƣớc tiền xử lý ảnh với Threshold. Ảnh nhị phân là ảnh mà giá trị của các  
điểm ảnh chỉ được biểu diễn bằng hai giá trị là 0 (đen) và 255 (trắng) tương ứng với 0  
và 1. Vì giá trị của điểm ảnh được biểu diễn bởi 2 giá trị là 0 hoặc 1, nên một điểm ảnh  
được biểu diễn bằng 1 bit nên ảnh có kích thước rất nhỏ. Có rất nhiều phương pháp nhị  
phân hóa cho hình ảnh, để biết phương pháp nào tốt nhất thì ta sẽ thử nghiệm trong bài  
toán của mình và từ đó tìm ra phương pháp phù hợp nhất. Trong quá trình thử nghiệm  
trong bài toán có rất nhiều phương pháp tìm ngưỡng, dựa trên kết quả thực nghiệm thì  
ta dùng phương pháp sau có hiệu quả nhất: ta sẽ cắt một khung hình nhỏ (khung hình  
này bao gồm cả nét số và nền, cách thực hiện ta sẽ tham khảo thêm phần phụ lục) để  
tính điểm có giá trị pixel nhỏ nhất (MinVal), sau đó dùng hàm InRange để cho những  
khu vực nằm trong khoảng giá trị nhờ công thức (\*) thì đó sẽ là số.  
*{ (MinVal – thresholdValue) , (MinVal – thresholdValue)}* (\*)  
Trong đó: thresholdValue là một sai số có giá trị là 45.



***Hình 3.8*** *: Nhị phân hóa*

Ta chọn sai số nhị phân hóa là 45 sở dĩ như vậy có thể mẫu khung hình ta đã cắt  
ra để tính giá trị MinVal chưa phải là giá trị tốt nhất của toàn bộ hình ảnh, thế nên ta sẽ  
cộng trừ thêm sai số là 45 cho kết quả tốt nhất. Ta chọn sai số 45 chứ không phải số  
khác vì trong quá trình thử nghiệm 45 là sai số có kết quả tốt nhất.

Dưới đây là hình ảnh khi ta dùng sai số là 30 Ta có thể thấy với sai số là 30 và  
khoảng giá trị nhị phân nằm trong khoảng {[MinVal – 30] , [MinVal +30]} sẽ làm ký tự  
số đứt nét khá nhiều, điều này sẽ sẽ gây khó khăn trong bƣớc phân tách ký tự và nhận  
dạng.

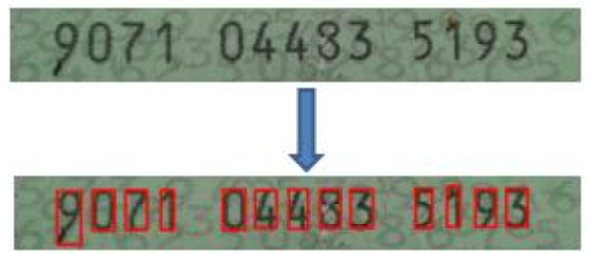


***Hình 3.9*** *: Nhị phân hóa sai số 30*

Giai đoạn tiền xử lý hình ảnh rất quan trọng, vì nó sẽ quyết định việc xác định và  
cắt các ký tự. Nếu thực hiện giai đoạn này không tốt thì sẽ gây ra rất nhiều phiền toái  
như không khoanh được các ký tự mình mong muốn, cắt thiếu ký tự,...  
lưu ý: các bước tiền xử lý hình ảnh của hình ảnh đưa vào huấn luyện phải giống  
với tiền xử lý hình ảnh của các hình ảnh đưa vào nhận dạng để quá trình nhận sẽ có độ  
chính xác cao nhất.

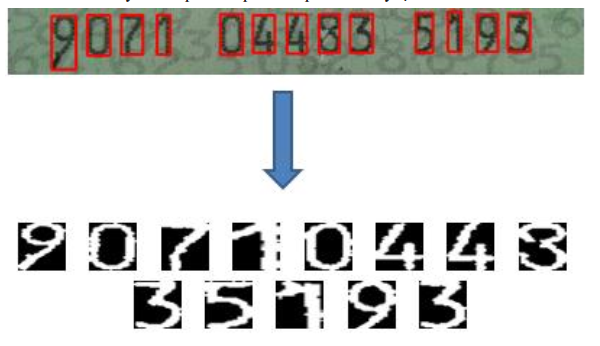
## **3.4 Xác định ký tự**

Sau khi thực hiện xong các kỹ thuật để xử lý hình ảnh ta sẽ tiếp tục bước xác  
định các ký tự.  
Ta dùng hàm *Contours* để xác định ký tự số. Contours là đường bao kết nối tất cả  
các điểm liền kề nhau có cùng màu sắc hoặc độ tương phản. Chính vì đặc tính này,  
contours thường được dùng trong xác định vật thể, nhận dạng, ...  
Ý tưởng của thuật toán này để khoanh ký tự là nó sẽ tìm cận trên, cận dưới, cận  
trái, cận phải. Cận trên: quét ảnh theo chiều ngang từ trên xuống dưới. Đến khi nào gặp  
điểm đen đầu tiên thì dừng lại đó là điểm cận trên. Cận dưới: quét ảnh theo chiều ngang  
từ dưới lên trên. Đến khi nào gặp điểm đen đầu tiên thì dừng lại đó là điểm cận dưới.  
Cận trái: quét ảnh theo chiều dọc từ trái sang phải. Đến khi nào gặp điểm đen đầu tiên  
thì dừng lại đó là điểm cận trái. Cận phải: quét ảnh theo chiều dọc từ phải sang trái. Đến  
khi nào gặp điểm đen đầu tiên thì dừng lại đó là điểm cận phải. Sau khi tìm được 4 điểm  
cận. Ta cắt ảnh theo 4 tọa độ vừa tìm được thì sẽ thu đƣợc vùng chỉ chứa ký tự.  
Trong trường hợp này chúng ta cũng sẽ dùng thuật toán xác định contours làm  
nền tảng để xác định các ký tự số. Trong OpenCV thuật toán tìm contours hoạt động  
dựa trên các chi tiết có màu trắng, trên nền màu đen. Ta dùng hàm *FindContours* trong  
OpenCV để khoanh vùng số.



## **3.5 Phân tách ký tự**

Từ các hình đã cắt vùng chứa số, ta tiến hành cắt từng chữ số sau đó lưu trữ dạng  
hình ảnh, đưa các số cùng loại vào cùng một thư mục và đặt tên cho thư mục chính là  
giá trị số đó. Ví dụ: ta đưa các hình ảnh chứa số 1 vào cùng một thư mục hình ảnh, sau  
đó đặt tên thư mục là 1, tương tự cho các số còn lại. Sau khi làm xong các bước đó, ta  
sẽ có 10 thư mục hình ảnh tương ứng với 10 ký tự số. Trong OpenCV ta dùng hàm  
*BoundingRect* để phân tách ký tự.



## **3.6 Nhận dạng ký tự bằng SVM**

### 3.6.1 Huấn luyện dữ liệu

Huấn luyện SVM là việc giải bài toán quy hoạch toàn phương SVM. Các phương  
pháp số giải bài toán quy hoạch này yêu cầu phải lưu trữ một ma trận có kích thước  
bằng bình phương của số lượng mẫu huấn luyện. Trong những bài toán thực tế, điều này  
là không khả thi vì thông thƣờng kích thƣớc của tập dữ liệu huấn luyện thƣờng rất lớn  
(có thể lên tới hàng chục nghìn mẫu). Nhiều thuật toán khác nhau được phát triển để  
giải quyết vấn đề nêu trên. Những thuật toán này dựa trên việc phân rã tập dữ liệu huấn  
luyện thành những nhóm dữ liệu. Điều đó có nghĩa là bài toán quy hoạch toàn phương

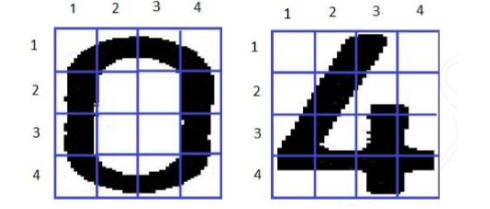
Với kích thước nhỏ hơn. Sau đó, những thuật toán này kiểm tra các điều kiện KKT  
(Karush-Kuhn-Tucker) để xác định phương án tối ưu.  
Một số thuật toán huấn luyện dựa vào tính chất: nếu trong tập dữ liệu huấn luyện  
của bài toán quy hoạch toàn phương con cần giải ở mỗi bước có ít nhất một mẫu vi  
phạm các điều kiện KKT, thì sau khi giải bài toán náy, hàm mục tiêu sẽ tăng. Như vậy,  
một chuỗi các bài toán quy hoạch toàn phương con với ít nhất một mẫu vi phạm các  
điều kiện KKT được đảm bảo hội tụ đến một phương án tối ưu. Do đó, ta có thể duy trì  
một tập dữ liệu làm việc đủ lớn có kích thước cố định và tại mỗi bước huấn luyện, ta  
loại bỏ và thêm vào cùng một số lượng mẫu.



### 3.6.2 Trích chọn đặc trưng

Quay lại bài toán nhận dạng tự mã card ta đang xét, làm sao để nhận dạng được  
các kí tự này dựa trên mô hình SVM? Trước hết cần phải nhận thấy rằng SVM là một  
bộ máy phân loại dữ liệu, muốn sử dụng được nó ta cần phải có dữ liệu, dữ liệu đối với  
các kí tự mà ta cần nhận dạng ở đây chính là các đặc trưng trong ảnh của kí tự đó. Giả  
sử ta cần phân loại 10 lớp dữ liệu (tương ứng với 10 kí tự trong card điện thoại), với  
mỗi lớp dữ liệu, ta tính toán đƣợc 10 vector đặc trưng (mỗi lớp 10 mẫu) và mỗi vector  
đặc trưng tương ứng với các đặc trưng trong một ảnh.  
Khi đó ta sẽ đưa vào bộ huấn luyện SVM toàn bộ dữ liệu này, sau đó với một  
ảnh bất kì, ta sẽ tính toán một vector đặc trƣng của ảnh đó, mô hình SVM sẽ xem xét  
xem dữ liệu này (tức vector đặc trưng này) thuộc vào lớp nào trong số những lớp mà nó  
đã được huấn luyện. Tính toán đặc trƣng trong ảnh. Đặc trưng trong ảnh là những đặc  
điểm riêng biệt giúp phân biệt ảnh này với ảnh khác. Việc xem xét nhƣ thế nào là các  
đặc trưng trong ảnh là một việc không có quy ước trước, nó phụ thuộc vào cách nghiên  
cứu, cài đặt của từng trường hợp cụ thể và vẫn đang được nghiên cứu để đưa ra những  
phương pháp tốt nhất.  
Trong phần này ta sẽ tính toán các vector đặc trưng dựa trên ý tưởng của phương  
pháp Haar, ta có thể áp dụng các phương pháp tính toán đặc trưng khác trong ảnh hay  
hơn, hiệu quả hơn. Giả sử ta có hai kí tự như hình , nhìn bằng mắt thường ta có thể

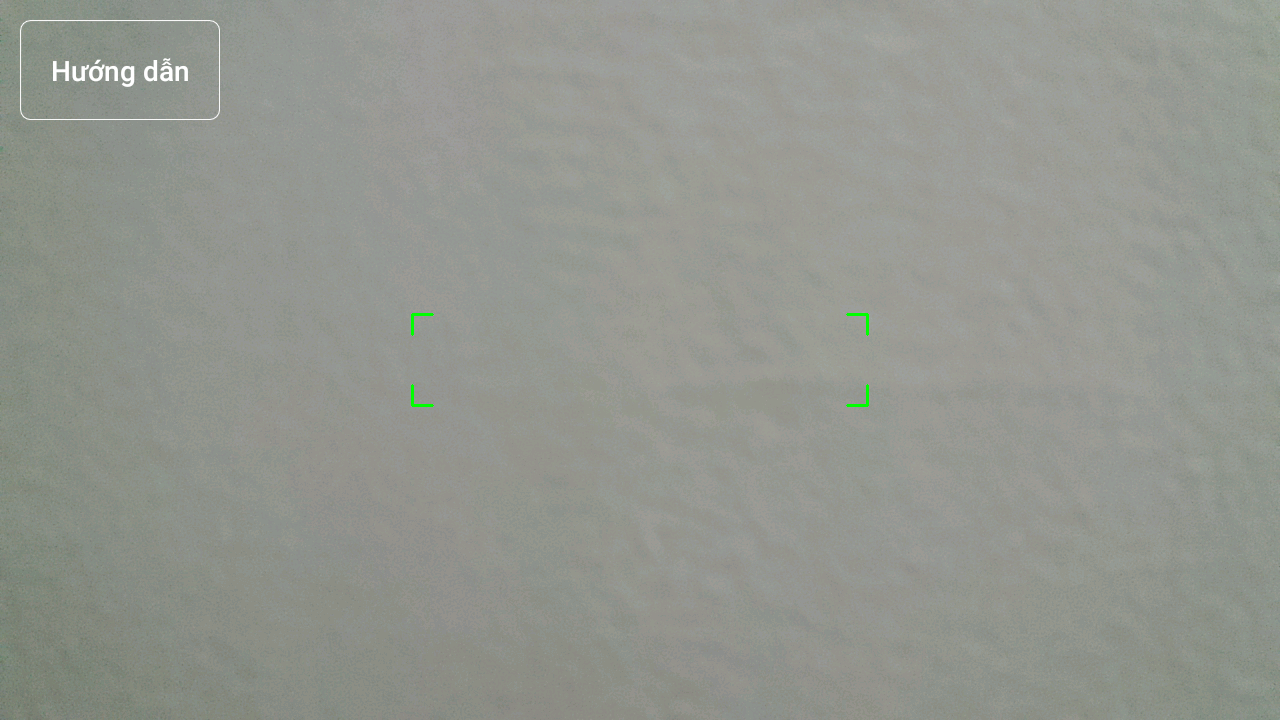
dễ dàng phân biệt được hai kí tự 0 và 4, tuy nhiên làm sao để máy tính phân biệt được  
hai kí tự này? Bây giờ ta sẽ đưa nhỏ hai ký tự này về cùng một kích thước, chia nhỏ ký  
tự thành 16 ô nhỏ khác nhau nhƣ hình dưới đây:

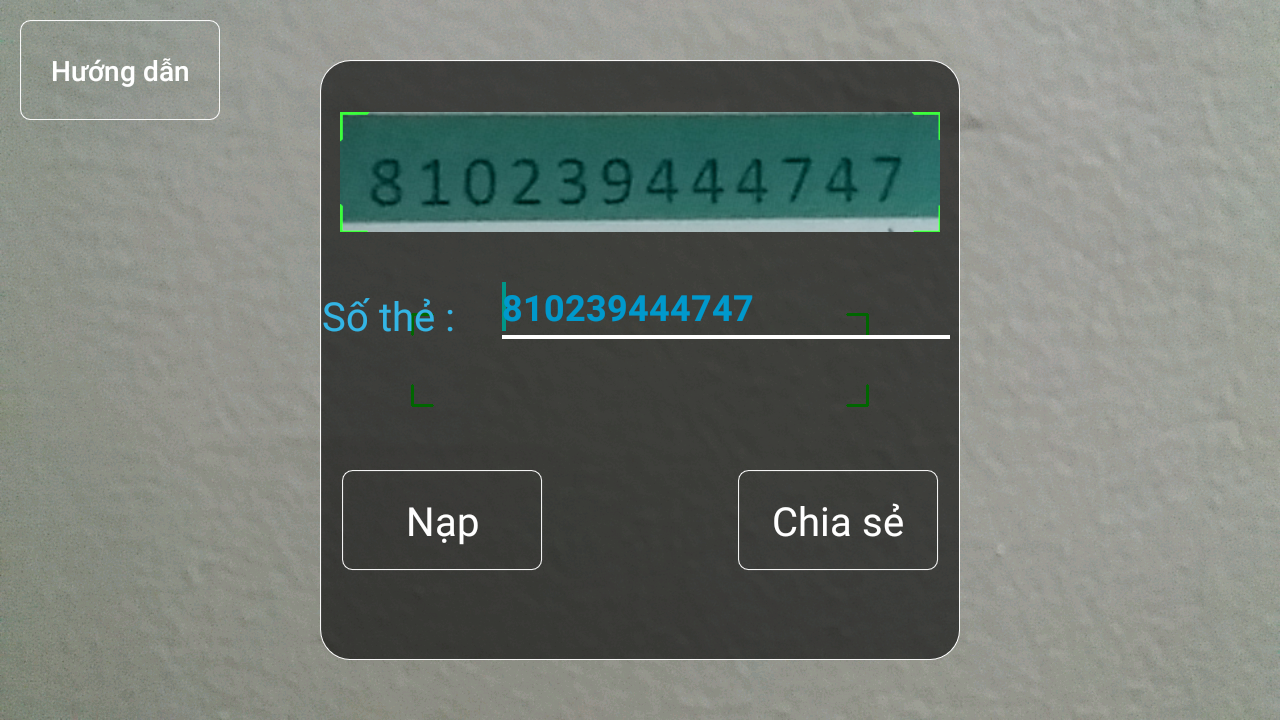


Ta nhận thấy rằng nếu tính tổng các pixel đen trong các ô của hai bức ảnh thì số  
0 và số 4 có thể phân biệt dựa vào các ô (1,1), (1, 4), (2, 2), (3,3),… tại những ô đó,  
tổng số các điểm ảnh đen là khác nhau hoàn toàn. Tính toán số điểm ảnh đen của 16 ô  
vuông này ta có thể thu đƣợc 16 đặc trưng của một ảnh, 16 đặc trưng này đủ để phân  
biệt kí tự 0 và 4.  
Tuy nhiên, với 10 kí tự ta cần phải tính toán nhiều hơn các đặc trưng, các đặc  
trưng không nhất thiết phải là 0 (tức không có điểm ảnh đen nào) hoặc 1 (tức là toàn số  
điểm ảnh đen trong ô) mà có thể là một tỉ lệ tương đối nào đó. Từ 16 đặc trƣng cơ bản  
trên, ta kết hợp chúng lại để tạo ra những đặc trƣng khác, chẳng hạn như lấy tổng các  
đặc trưng trên đường chéo (1,1) + (2,2) + (3,3) + (4,4) hoặc tổng các đặc trưng xung  
quanh đường biên của ảnh … số đặc trương càng lớn thì việc phân loại các lớp càng ít bị  
sai, có nghĩa là xác suất nhận dạng càng lớn.

# CHƯƠNG 4: HOÀN THÀNH ỨNG DỤNG

**4.1 Giao diện của phần mềm**

* + Màn hình chính : là một camera
* Khung màu xanh : là khung để cho ngươi sử dụng phần mềm đưa phần mã thẻ cào vừa với khung để nhận diện mã thẻ cào
* Nút hướng dẫn : sẽ hiển thị hướng dẫn sử dụng khi người dùng bấm vào
  + Màn hình hiển thị khi bắt được mã số thẻ cào :



Khi nhận diện được mã số thẻ thì phần mềm sẽ hiển thị một dialog gồm :

* Nút nạp : có chức năng nạp
* Nút chia sẻ : có chức năng chia sẻ mã thẻ cào qua các ứng dụng khác.
* EditText: hiển thị mã số thẻ cào nhận diện được. Khi click vào sẽ cho phép bạn chỉnh sửa được.
* Image : là phần cắt ra từ vùng mã số thẻ cào . Cho phép người dùng đối chiếu với kết quả nhận diện được để có thể chỉnh sửa.

# CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## **5.1 Kết luận**

Thuận lợi và khó khăn trong việc làm đô án

* **Thuận lợi.**
* Sự quyết tâm cao của thanh viên trong nhóm.
* Bản thân mỗi thành viên luôn luôn cố gắng và đã có kinh nghiệm trong việc lập trình trực quan.
* Nguồn tài liệu phong phú : google , stackoverflow…..
* **Khó khăn:**
* Việc phối hợp nhóm đôi lúc gặp khó khăn khi không hiểu ý nhau
* Nguồn tại liệu quá nhiều dẫn đến khó khăn trong việc chọn lọc
* Việc tìm hiểu opencv là một khó khăn rất lớn khi hoàn toàn chưa biết về nó
* Khó khăn trong việc đọc tài liệu tiếng anh.
* **Kết quả đạt được**
* Hoàn thành được một ứng dụng khá hoàn chỉnh
* Đạt được tốc độ nhận dạng như mong muốn
* Có kinh nghiệm trong việc làm và quản lí nhóm

## **5.2 Ưu nhược điểm của phần mềm**

### 5.2.1 Ưu điểm.

* Phần mềm nhận diện tốt. Tốc độ nhận dạng nhanh
* Giải quyết được bài toán nạp mã thẻ cào nhanh hơn so với cách nạp truyền thống**.**

### 5.2.2 Nhược điểm.

* Đôi lúc nhận diện chưa được chính xác do việc rung lắc
* Nhận diện không được ở nơi thiếu sáng

## **5.3 Hướng phát triển**

* Bổ sung thêm chức năng bật đèn flash để có thể nhận diện ở nơi thiếu sang
* Bổ sung thêm chức năng kiểm tra tài khoản chính và tài khoản phụ
* Tăng thêm sự chính xác khi nhận diện
* Bổ sung thêm chức năng nhận diện mã thẻ cào giấy in.

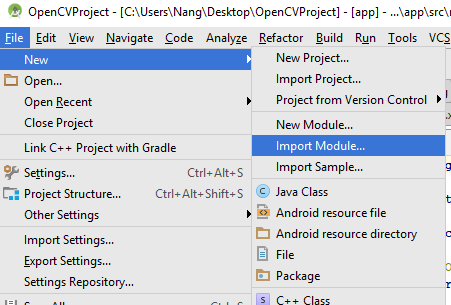
# CHƯƠNG 6 : PHỤ LỤC

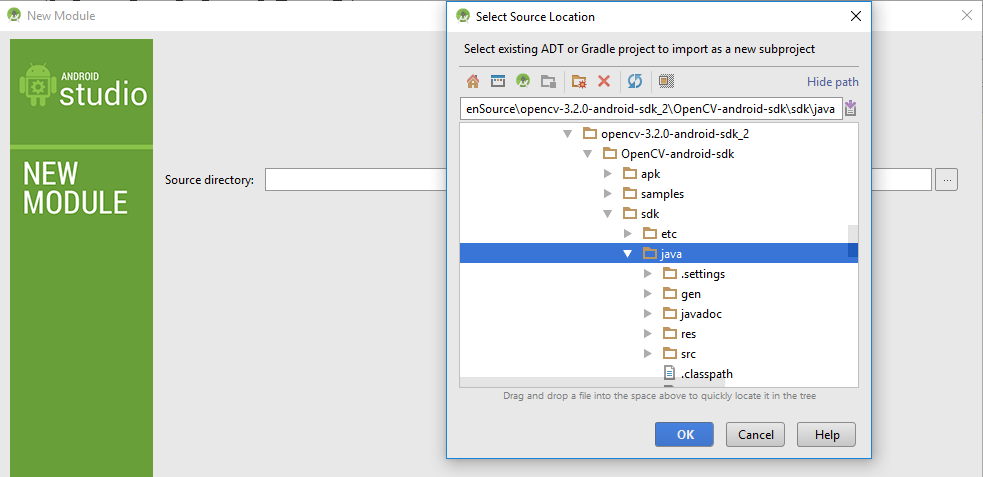
## 1. Cài đặt Opencv vào android studio để lập trình :

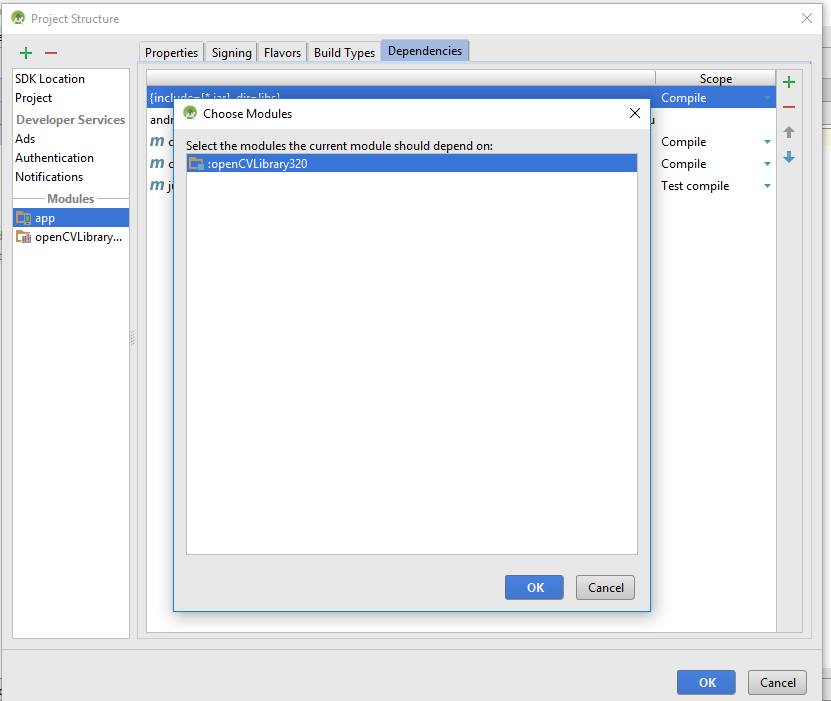
* **Hướng dẫn cài đặt :**
* **Chuẩn bị :**
* Đã cài đặt android studio.
* Thư viện Opencv cho android.
* NDK cho android.

**Bước 1** : Tạo project android bình thường. Sau đó import module opencv vào android:

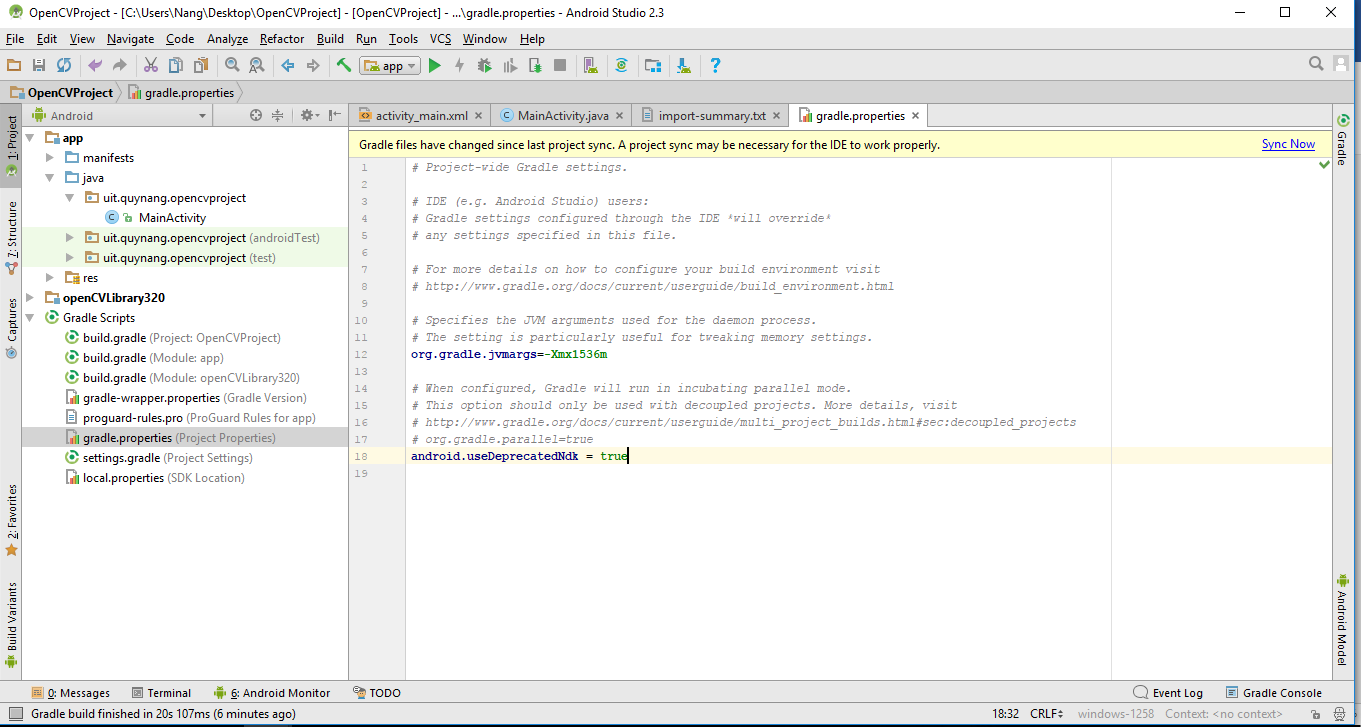
File 🡪 New 🡪 Import Module 🡪 Trỏ đến thư mục OpencvAndoird đã tải về 🡪 SDK 🡪 Java

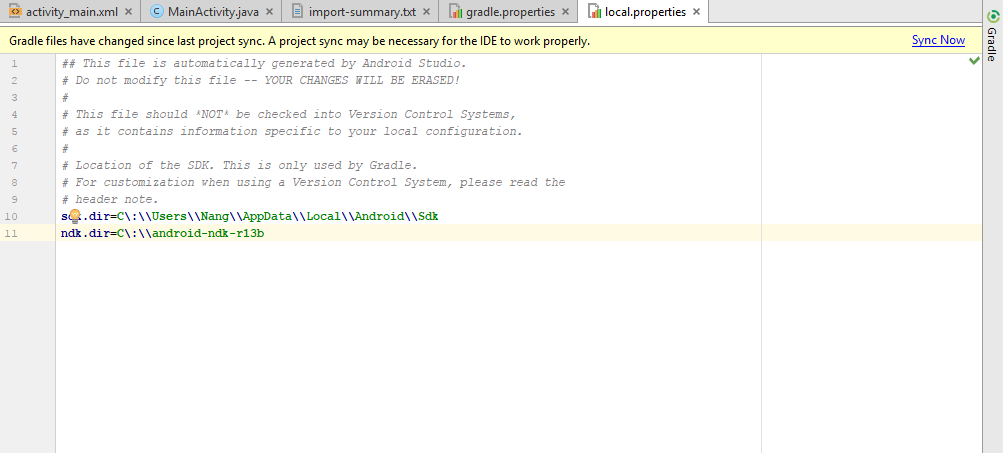




**Bước 2** : chuột phải vào App 🡪 Open Module Setting 🡪 App -> Dependencies 🡪dấu cộng 🡪 thêm thư viện opencv

**Bước 3 :** Vô file gradle.properties. Thêm dòng android.useDeprecatedNdk = true



Thêm đường dẫn thư mục ndk cho android như ở bước chuẩn bị vào file local.properties(SDK Location). Đường dẫn của máy tính bạn

**Bước 4 :** Vào build.gradle(Moudule: OpencvLibrary) chỉnh lại compileSdkVersion = compileSdkVersion trong file build.gradle(Moudule: app)

**Bước 5:** Tạo thư mục jniLibs sau đó copy SDK -> native ->libs( của thư mục opencv) vào thư mục vưa tạo.

* Như vậy chúng ta đã tạo xong một project android với thư viện opencv

## 2. Code training dữ liệu mô hình SVM bằng C++ Visual Studio:

#include <opencv2\imgproc.hpp>

#include <opencv2\core.hpp>

#include <opencv2\highgui\highgui.hpp>

#include <opencv2\ml\ml.hpp>

#include <iostream>

#include"dirent.h"

using namespace cv;

using namespace std;

using namespace ml;

// nhan dang dua vao dac trung pixel

int count\_pixel(Mat img, bool black\_pixel = true)

{

int black = 0;

int white = 0;

for (int i = 0; i < img.rows; ++i)

for (int j = 0; j < img.cols; ++j)

{

if (img.at<uchar>(i, j) == 0)

black++;

else

white++;

}

if (black\_pixel)

return black;

else

return white;

}

vector<float> calculate\_feature(Mat src)

{

Mat img;

if (src.channels() == 3)

{

cvtColor(src, img, CV\_BGR2GRAY);

threshold(img, img, 100, 255, CV\_THRESH\_BINARY);

}

else

{

threshold(src, img, 100, 255, CV\_THRESH\_BINARY);}

vector<float> r;

vector<int> cell\_pixel;

resize(img, img, Size(40, 40));

int h = img.rows / 4;

int w = img.cols / 4;

int S = count\_pixel(img);

int T = img.cols \* img.rows;

for (int i = 0; i < img.rows; i += h)

{

for (int j = 0; j < img.cols; j += w)

{

Mat cell = img(Rect(i, j, h, w));

int s = count\_pixel(cell);

float f = (float)s / S;

r.push\_back(f);

}

}

for (int i = 0; i < 16; i += 4)

{

float f = r[i] + r[i + 1] + r[i + 2] + r[i + 3];

r.push\_back(f);

}

for (int i = 0; i < 4; ++i)

{

float f = r[i] + r[i + 4] + r[i + 8] + r[i + 12];

r.push\_back(f);

}

r.push\_back(r[0] + r[5] + r[10] + r[15]);

r.push\_back(r[3] + r[6] + r[9] + r[12]);

r.push\_back(r[0] + r[1] + r[4] + r[5]);

r.push\_back(r[2] + r[3] + r[6] + r[7]);

r.push\_back(r[8] + r[9] + r[12] + r[13]);

r.push\_back(r[10] + r[11] + r[14] + r[15]);

r.push\_back(r[5] + r[6] + r[9] + r[10]);

r.push\_back(r[0] + r[1] + r[2] + r[3] + r[4] + r[7] + r[8] + r[11] +

r[12] + r[13] + r[14] + r[15]);

return r;

}

vector<string> list\_folder(string path)

{

vector<string> folders;

DIR \*dir = opendir(path.c\_str());

struct dirent \*entry;

while ((entry = readdir(dir)) != NULL)

{

if ((strcmp(entry->d\_name, ".") != 0) && (strcmp(entry->d\_name,

"..") != 0))

{

string folder\_path = path + "/" + string(entry->d\_name);folders.push\_back(folder\_path);

}

}

closedir(dir);

return folders;

}

vector<string> list\_file(string folder\_path)

{

vector<string> files;

DIR \*dir = opendir(folder\_path.c\_str());

struct dirent \*entry;

while ((entry = readdir(dir)) != NULL)

{

if ((strcmp(entry->d\_name, ".") != 0) && (strcmp(entry->d\_name,

"..") != 0))

{

string file\_path = folder\_path + "/" + string(entry-

>d\_name);

files.push\_back(file\_path);

}

}

closedir(dir);

return files;

}

void main()

{

int number\_of\_class = 10;

int number\_of\_sample = 70;

int number\_of\_feature = 32;

Ptr<SVM> svm = SVM::create();

svm->setType(SVM::C\_SVC);

svm->setKernel(SVM::LINEAR);

svm->setTermCriteria(TermCriteria(TermCriteria::MAX\_ITER, 700, 1e-6));

Mat label(1, 700, CV\_32SC1);

Mat data(700, 32, CV\_32FC1);

int index = 0;

vector<string> folders =

list\_folder("C:\\Users\\trandung\\Desktop\\Numbers");

if (number\_of\_class != folders.size() || number\_of\_sample <= 0 ||

number\_of\_class <= 0)

{

cout << "error";

return;

}

Mat src;

for (size\_t i = 0; i < folders.size(); ++i)

{

vector<string> files = list\_file(folders.at(i));

if (files.size() <= 0 || files.size() != number\_of\_sample)

{

cout << "error";return;

}

string folder\_path = folders.at(i);

string label\_folder = folder\_path.substr(folder\_path.length() -

1);

for (size\_t j = 0; j < files.size(); ++j)

{

src = imread(files.at(j));

if (src.empty()) {

cout << "error";

return;

}

vector<float> feature = calculate\_feature(src);

for (size\_t t = 0; t < feature.size(); ++t)

data.at<float>(index, t) = feature.at(t);

label.at<int>(0, index) = i;

index++;

}

}

cout << data;

svm->train(data, ROW\_SAMPLE, label);

svm->save("D:\\trainingchuan.xml");

waitKey();

getchar();