

**BỘ GIÁO DỤC ĐÀO TẠO**

**BỘ QUỐC PHÒNG**  
**HỌC VIỆN KỸ THUẬT QUÂN SỰ**



**BÁO CÁO BÀI TẬP LỚN MÔN XỬ LÝ ẢNH**

**ỨNG DỤNG ĐỌC THẺ ĐIỆN THOẠI TRÊN HỆ ĐIỀU HÀNH  
WINDOWS PHONE**

Danh sách thành viên trong nhóm

**Trương Tiến Phúc**

**TH11A**

**Dương Hồ Minh Tú**

**TH11A**

**Hà Nội - 2016**

## MỤC LỤC

1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone, mục tiêu đặt ra .....	2
1.1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone .....	2
1.2. Mục tiêu đặt ra .....	2
2. Hướng tiếp cận .....	2
2.1. Giải pháp thực hiện .....	2
2.1.1. Phương pháp sẽ sử dụng .....	2
2.1.2. Các phương pháp đã có, so sánh Tesseract OCR với ABBY Reader .....	2
2.2. Kỹ thuật xử lý ảnh áp dụng vào bài toán .....	3
2.2.1. Các bước tiền xử lý ảnh trước khi đưa vào nhận dạng .....	3
3. Thực nghiệm, đánh giá chất lượng sản phẩm .....	10
3.1. Giao diện ứng dụng .....	10
3.2. Các mẫu dữ liệu và kết quả thu được .....	12
4. Những điểm ưu và hạn chế của ứng dụng, hướng phát triển .....	13
4.1. Đánh giá .....	13
4.2. Hướng phát triển .....	13
Phụ lục .....	13

## **1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone, mục tiêu đặt ra**

### **1.1. Giới thiệu bài toán đọc thẻ điện thoại trên hệ điều hành windows phone**

Thông thường khi nạp thẻ cho điện thoại bạn phải thực hiện lệnh \*100\*mã thẻ # rồi ấn phím gọi mới nạp được, tuy nhiên, như vậy cũng mất khá nhiều thời gian cho việc thực hiện nạp thẻ và quan trọng có thể bạn ấn nhầm, chưa kể đối với người già việc đọc các số trên thẻ gặp rất nhiều khó khăn. Nhưng khi sử dụng ứng dụng đọc thẻ điện thoại, bạn hoàn toàn giải quyết được điều đó, khi cần nạp thẻ, bạn chỉ cần một vài thao tác đơn giản là có thể nạp thẻ điện thoại thành công.

### **1.2. Mục tiêu đặt ra**

Mục tiêu ứng dụng đặt ra có thể đọc hầu hết các loại thẻ điện thoại với độ chính xác cao nhất.

## **2. Hướng tiếp cận**

### **2.1. Giải pháp thực hiện**

#### *2.1.1. Phương pháp sẽ sử dụng*

Bước 1: Tiền xử lý ảnh thu nhận được trước khi đưa vào nhận dạng.

- Phân vùng ảnh có chứa mã số thẻ (cắt ảnh).
- Chuyển sang ảnh mức xám.
- Làm mịn ảnh.
- Tăng cường độ sắc nét của ảnh.
- Nhị phân hóa ảnh bằng thuật toán Otsu.

Bước 2: Sử dụng thuật toán nhận dạng từ thư viện Tesseract.

#### *2.1.2. Các phương pháp đã có, so sánh Tesseract OCR với ABBY Reader*

##### **a. Giới thiệu ABBYY FineReader**

ABBYY FineReader là một hệ thống nhận dạng ký tự quang học (OCR) chuyển đổi các tài liệu đã được quét, tài liệu PDF và các tệp hình ảnh (bao gồm ảnh kỹ thuật số) sang định dạng có thể chỉnh sửa.

Nhờ công nghệ nhận dạng tài liệu thích ứng (ADRT®) của ABBYY, ABBYY FineReader có thể phân tích và xử lý toàn bộ tài liệu, thay vì từng trang trong một lúc. Phương pháp này giữ lại cấu trúc của tài liệu nguồn, bao

gồm định dạng, siêu liên kết, địa chỉ email, đầu trang và chân trang, chú thích hình ảnh và bảng, số trang và ghi chú cuối trang.

ABBYY FineReader có thể nhận dạng văn bản được viết bằng bất kỳ ngôn ngữ nào trong số 190 ngôn ngữ được hỗ trợ hoặc kết hợp những ngôn ngữ này. Trong số các ngôn ngữ được hỗ trợ có Tiếng Ả Rập, Tiếng Việt, Tiếng Hàn, Tiếng Trung, Tiếng Nhật, Tiếng Thái và Tiếng Do Thái. ABBYY FineReader có thể tự động phát hiện ngôn ngữ của tài liệu.

#### b. So sánh Tesseract với ABBY LineReader

Từ kết quả so sánh giữa Tesseract OCR và ABBYY FineReader OCR của dự án IMPACT - được hỗ trợ bởi Cộng đồng Châu Âu theo Chương trình làm việc ICT FP7. Dự án này phối hợp thực hiện bởi Thư viện Quốc gia Hà Lan<sup>1</sup>.

**Kết luận:** trong điều kiện lý tưởng Tesseract tỏ ra hiệu quả hơn so với FineReader.

## 2.2. Kỹ thuật xử lý ảnh áp dụng vào bài toán

### 2.2.1. Các bước tiền xử lý ảnh trước khi đưa vào nhận dạng

Mục tiêu thực hiện các bước tiền xử lý là thực hiện các phương pháp xử lý ảnh từ đó nâng cao chất lượng ảnh đầu vào. Đối với bài toán nhận dạng văn bản nói chung và bài toán nhận dạng kí tự số nói riêng, ảnh số được tạo ra từ những kí tự với thiết bị quét không phải lúc nào cũng cho hình ảnh tốt nhất, vì ảnh quét có chữ mờ hay mất nét, có thể có nhiễu. Trong mục này, ta xem xét hai kỹ thuật nâng cao chất lượng ảnh số phục vụ cho quá trình nhận dạng là làm mịn ảnh và tăng cường sắc nét ảnh

Trong thực tế, ảnh thu nhận ban đầu là ảnh chứa nhiều màu hơn là trắng và đen. Vì vậy để có thể thực hiện được quá trình phân tích và nhận dạng, cần phải chuyển chúng thành ảnh nhị phân trong đó mỗi điểm ảnh được biểu diễn bởi một trong 2 giá trị là 0 hoặc 255. Đầu tiên, ảnh màu nhận vào sẽ được chuyển thành ảnh xám với các mức xám có giá trị từ 0 đến 255 dựa trên ba giá trị red, green, blue của ảnh đầu vào.

---

<sup>1</sup> Improving Access to Text IMPACT: [https://lib.psnec.pl/Content/358/PSNC\\_Tesseract-FineReader-report.pdf](https://lib.psnec.pl/Content/358/PSNC_Tesseract-FineReader-report.pdf)

Phương trình chuyển đổi ảnh màu sang ảnh xám:

$$\text{GreyPixel} = \text{Red} * 0.299 + \text{Green} * 0.587 + \text{Blue} * 0.114$$

Nhằm tương cường chất lượng ảnh, nhóm thực hiện làm mịn ảnh và tăng độ sắc nét bức ảnh trước khi nhị phân hóa ảnh bằng thuật toán Otsu.

**Mịn ảnh** được thực hiện dựa trên bộ lọc trơn (Smoothing filter) nhằm loại nhiễu, bước này dùng trong quá trình tiền xử lý (Pre-processing) khi phải giảm bớt một số chi tiết không cần thiết của một đối tượng nào đó trong ảnh. Nhóm sử dụng bộ lọc Gaussian:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \sigma = 0.5$$

Trái ngược với bộ lọc mịn ảnh, bộ lọc **tăng cường độ sắc nét** (Sharpening filter) để nhấn mạnh hay cải thiện chi tiết bị mờ của đối tượng đang xét trong ảnh văn bản, ví dụ như dấu của các chữ cái không rõ ràng. Qua những bộ lọc loại này, bức ảnh màu tối có mức sáng trung bình của toàn bộ điểm ảnh được tăng cường. Ma trận của một loại bộ lọc tăng cường độ sắc nét ảnh thường sử dụng có các hệ số như sau:

$$M = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Ảnh sau khi đã được nâng cao chất lượng, so sánh mức xám của từng điểm với một ngưỡng cho trước để quyết định điểm đó sẽ là 0 hoặc 255, giá trị 0 biểu diễn cho màu đen và 255 biểu diễn cho màu trắng. Một trong các phương pháp được đưa ra là giải thuật Otsu để tìm ra ngưỡng thích hợp đối với mỗi ảnh nhận vào.

Cho trước ảnh đa mức xám có L mức sáng, ký hiệu  $p(i)$  là tần suất xuất hiện của mức sáng thứ  $i$  và các trọng số tần suất  $\omega_0(t), \omega_1(t)$  dựa theo ngưỡng  $t$  như sau:

$$\begin{cases} \omega_0(t) = \sum_{i=0}^{t-1} p(i) \\ \omega_1(t) = \sum_{i=t}^{L-1} p(i) \end{cases}$$

Các hàm thuộc:

$$\begin{cases} \mu_0(t) = \sum_{i=0}^{t-1} \frac{i \cdot p(i)}{\omega_0(t)} \\ \mu_1(t) = \sum_{i=t}^{L-1} \frac{i \cdot p(i)}{\omega_1(t)} \\ \mu_T(t) = \sum_{i=0}^{L-1} i \cdot p(i) \end{cases}$$

Từ đó, suy ra sự liên hệ giữa những trọng số và các hàm thuộc:

$$\begin{cases} \omega_0(t)\mu_0(t) + \omega_1(t)\mu_1(t) = \mu_T(t) \\ \omega_0(t) + \omega_1(t) = 1 \end{cases}$$

Otsu định nghĩa giá trị:

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)(\mu_0(t) - \mu_1(t))^2, \forall t \leq L - 1$$

và Otsu cho rằng ngưỡng thích hợp là giá trị lớn nhất trong số các giá trị  $\sigma_b^2(t)$ . Như vậy, ngưỡng được chọn phụ thuộc theo đặc trưng mức sáng

trong ảnh số, và do vậy ngưỡng của giải thuật Otsu được xem là ngưỡng thích nghi (Adaptive thresholding).

Một cách tiếp cận với giải thuật **Otsu** mà nhóm thực hiện trong bài toán này là:

Để giảm bớt quá trình tính toán, ta thực hiện tính toán các giá trị mức xám trong khoảng giá trị của ma trận histogram với công thức:

$P(u, v) = \sum_{i=u}^V P_i$  (1) và tính giá trị trung bình trong khoảng giá trị của ma trận histogram với công thức:  $S(u, v) = \sum_{i=u}^V i \cdot P_i$  (2)

Với giá trị  $u=1$ , Phương trình (1) và (2) được viết lại như sau:

$$\begin{aligned} P(1, v+1) &= P(1, v) + p_{v+1} \text{ với } P(1, 0) = 0, (3) \\ S(1, v+1) &= S(1, v) + (v+1)p_{v+1} \text{ với } S(1, 0) = 0, (4) \end{aligned}$$

Với  $p_{v+1}$  là xác suất của mức xám  $v+1$ .

Từ (1), (2), (3), (4) ta có được

$$\begin{aligned} P(u, v) &= P(1, v) - P(1, u-1), (5) \\ S(u, v) &= S(1, v) - S(1, u-1), (6) \end{aligned}$$

Như vậy,

Từ (5),  $\omega_k$  được viết lại như sau:

$$\omega_k = P(1, t_k) - P(1, t_{k-1}) = P(t_{k-1} + 1, t_k)$$

Tương tự,  $\mu_k$  được viết lại như sau:

$$\mu_k = S(1, t_k) - S(1, t_{k-1}) = S(t_{k-1} + 1, t_k)$$

Cuối cùng ta được công thức:

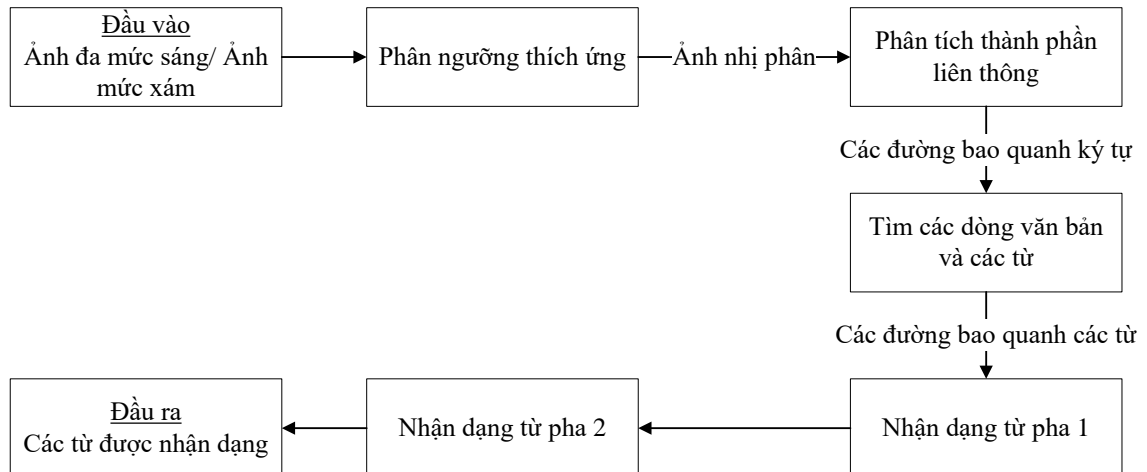
$$\sigma_b^2(t) = H(t_{i-1} + 1, t_i) = \frac{S(t_{i-1} + 1, t_i)^2}{P(t_{i-1} + 1, t_i)}.$$

### 2.2.2. Thuật toán nhận dạng ký tự OCR của Tesseract

#### a. Giới thiệu Tesseract

Tesseract là một công cụ OCR mã nguồn mở được nghiên cứu và phát triển bởi HP trong giai đoạn 1984-1994. Nó được biết như là một phần mềm thêm vào cho dòng sản phẩm máy quét của HP. Trong giai đoạn này, nó vẫn còn rất sơ khai và chỉ được dùng để cải thiện chất lượng của các bản in. Kể từ năm 2006, nó đã được phát triển bởi Google.

#### b. Kiến trúc tổng thể của thuật toán nhận dạng ký tự trong Tesseract



*Kiến trúc tổng thể của Tesseract*

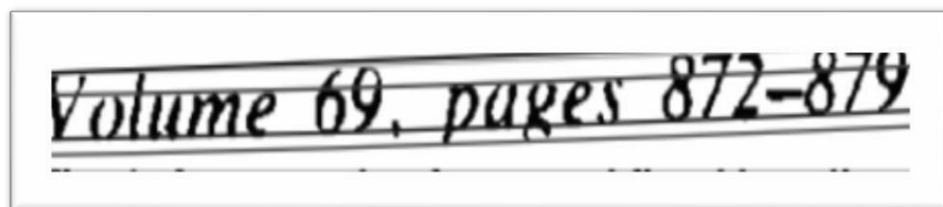


Tạo ngưỡng thích nghi giúp loại bỏ các yếu tố nền của hình ảnh (ví dụ như ánh sáng, bóng, ...) và giúp phân tích các pixel thành ảnh nhị phân. Giai đoạn tiếp theo là ảnh nhị phân được đưa vào bộ Phân tích thành phần liên thông (Connected component analysis) để tìm ra hình dạng phác thảo của những thành phần liên thông. Đây là một tiến trình phức tạp mất nhiều thời gian nhưng cần có để tách ra các ký tự có trong hình. Sau đó, nhận dạng được tiến hành qua một quá trình với hai lần nhận dạng. Lần thứ nhất: nhận ra lần lượt từng từ. Mỗi từ có nghĩa là đạt yêu cầu và được thông qua và được lưu vào dữ liệu. Lần thứ hai, khi phân loại thích ứng, công cụ sẽ nhận dạng lại các từ không được nhận dạng tốt ở lần trước đó.

### c. Xác định dòng và từ

**Xác định dòng:** thực hiện các chức năng như xác định dòng chặn dưới và chặn trên, đối với mỗi dòng thì cắt gọn từ trước khi xác định vùng của mỗi ký tự, ngoài ra cần nhận dạng khoảng cách giữa chữ và số. Lọc dãy dòng không chỉ tìm dãy ký tự trong từng dòng mà còn phát hiện các ký tự có độ cao chênh lệch trong dòng như ký tự chấm câu, ký tự dấu và nhiều... Tuy nhiên, nếu ảnh số chứa các dòng có độ nghiêng hoặc cong thì giải thuật trở nên phức tạp. Để giúp giảm bớt mất thông tin khi nhận dạng ảnh nghiêng thì áp dụng giải thuật biến đổi Hough tìm góc nghiêng để đưa ảnh số trở lại vị trí thông thường. Trong trường hợp dòng có độ cong nào đó thì phải thiết lập các dòng cơ sở.

**Thiết lập dòng cơ sở:** Khi dòng văn bản được tìm thấy, các dòng cơ sở được thiết lập chính xác hơn bằng cách sử dụng một đường có tên là spline toàn phương (là dòng mà được kết hợp từ nhiều đoạn). Nó giúp Tesseract xử lý các trang có đường cơ sở là đường cong. Các dòng cơ sở được thiết lập bằng cách phân vùng các blobs thành các nhóm có thể thay thế thích hợp liên tục trong đường cơ sở thẳng ban đầu.



*Đường cơ sở hình cong*

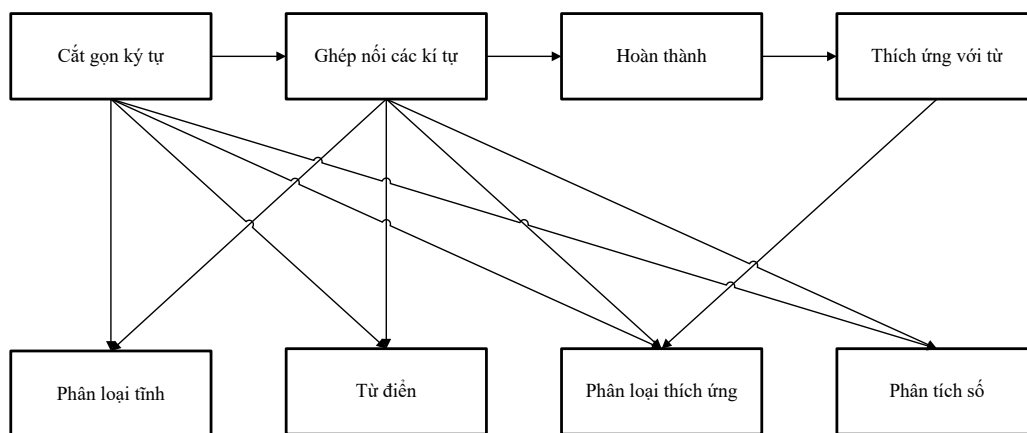
**Cắt gọn từ** sẽ xác định xem có các ký tự liên nhau trong một từ hay không. Nếu có nó sẽ cắt nhỏ các ký tự ra thành các ký tự riêng lẻ.



*Cắt các ký tự liên nhau*

**Nhận dạng khoảng cách giữa chữ và số** xác định khoảng cách giữa các số hoặc giữa các chữ là một vấn đề khá phức tạp. Tesseract giải quyết những vấn đề này bằng cách đo khoảng cách trong một phạm vi hạn chế theo chiều dọc giữa dòng cơ sở và dòng trung bình.

**Nhận dạng từ** là quá trình phân tích một từ được chia ra thành các ký tự như thế nào.



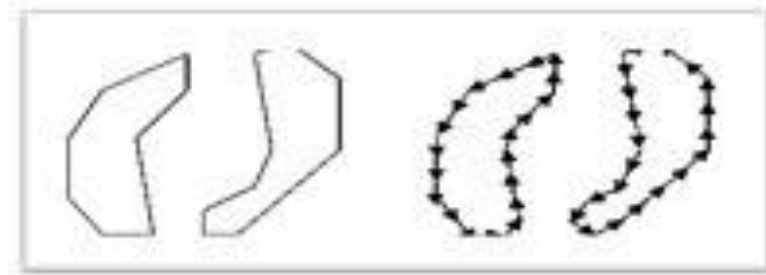
*Sơ đồ nhận dạng từ*

Mỗi ký tự cần nhận dạng có những đặc trưng riêng, có khoảng 50 tới 100 đặc trưng điển hình trong mỗi ký tự. Mỗi đặc trưng chứa 3 tham số là hoành độ, tung độ, và góc xoay. Trong khi đó mỗi ký tự mẫu có từ 10 tới 20 đặc trưng, mỗi đặc trưng có 4 tham số là hoành độ, tung độ, góc xoay, độ dài.

Vấn bản luôn tồn tại độ dư thừa ký tự và từ vựng, vậy chức năng phân loại ký tự tạo ra danh sách rút gọn chứa các ký tự mà ký tự đối sánh có thể

trùng khớp. Các lớp ký tự mẫu sinh ra các lớp véc tơ bit tương ứng với những đặc trưng của từng ký tự.

**Những đặc trưng của ký tự nhận dạng** (Features of character) được đối sánh với lớp véc tơ bit của ký tự mẫu, và tính toán sự khác nhau giữa các đặc trưng của chúng. Bên cạnh đó có tham số thứ hai là độ dài của ký tự nhận dạng.



*Các đặc trưng ký tự được nhận dạng*

Hệ số đánh giá đối sánh là tích hai tham số trên, cặp đối sánh nào có hệ số nhỏ nhất thì xem như chúng là tương tự nhau.

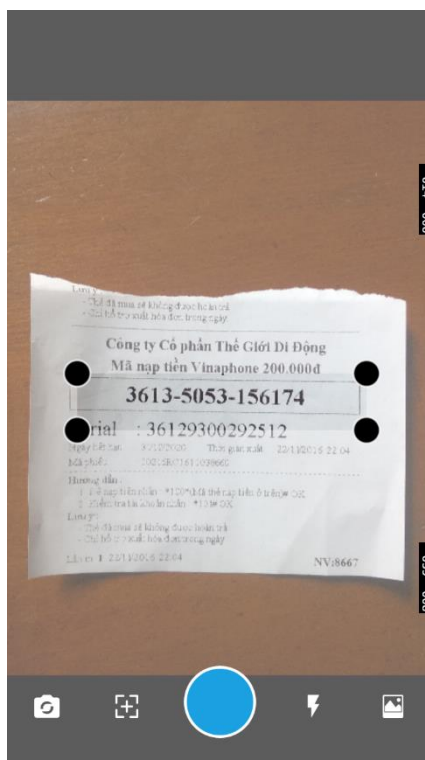
**Chức năng phân loại tĩnh** (Static Classifier) phù hợp với các ký tự có phong chữ bất kỳ, nhưng nó chủ yếu được dùng để nhận dạng các ký tự riêng như ký tự chú giải, dấu ngăn cách hay kết thúc câu... trong khi chức năng phân loại thích ứng (Adaptive Classifier) dùng để nhận dạng các ký tự theo phong chữ chuẩn.

### 3. Thực nghiệm, đánh giá chất lượng sản phẩm

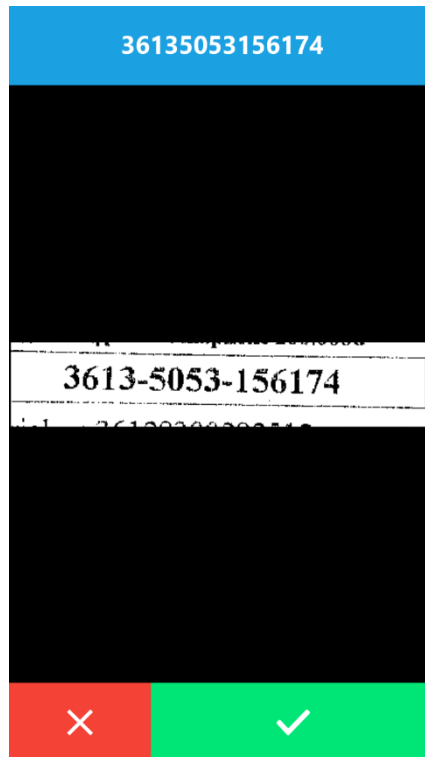
#### 3.1. Giao diện ứng dụng



*Giao diện chính của ứng dụng, bao gồm 5 chức năng chính (từ trái sang phải): Đặt lại camera, Quét, Chụp hình, Bật/Tắt flash, Chọn ảnh từ thư viện*



*Giao diện ứng dụng sau khi chụp ảnh hoặc chọn ảnh xong.*



*Giao diện sau khi nhận diện với các chức năng (từ trái sang phải): Hủy và quay lại, Tiếp tục, tiến hành gọi để nạp thẻ.*

### 3.2. Các mẫu dữ liệu và kết quả thu được

Nhóm thực hiện tiến hành kiểm tra, nghiệm thu kết quả của ứng dụng với những tham số là điều kiện như sau:

- Ánh sáng của môi trường bên ngoài: được đánh giá kết quả trong 03 thời điểm: 13h, 18h và 21h. Đối với thời điểm 13h, không sử dụng thêm bất kỳ nguồn sáng bổ sung nào khác; Đối với thời điểm 18h, có sử dụng thêm đèn tuýp, thời điểm 21h chỉ sử dụng đèn bàn.
- Khoảng cách từ camera đến thẻ: đánh giá trên 03 khoảng cách 5cm, 10cm, 15cm.
- Góc quay của thẻ (theo trục song song với trục dọc của điện thoại): thay đổi lần lượt với 03 góc  $0^\circ$ ,  $15^\circ$ ,  $30^\circ$ .

Từ những kết quả thực nghiệm thu được trong quá trình đánh giá trên các mẫu dữ liệu<sup>2</sup>, có thể rút ra kết luận khoảng cách phù hợp nhất để sử dụng ứng dụng là từ 5cm đến 10cm trong điều kiện ban ngày.

---

<sup>2</sup> Kết quả chi tiết của các thử nghiệm thuộc phần phụ lục

## 4. Những điểm ưu và hạn chế của ứng dụng, hướng phát triển

### 4.1. Đánh giá

#### a. Ưu điểm

- Tiện lợi, cho phép chọn ảnh từ thư viện hoặc chụp ảnh mới.
- Giao diện đơn giản, dễ sử dụng.

#### b. Hạn chế




- Để ứng dụng có thể nhận dạng được một cách chính xác nhất, người sử dụng cần phải cào hết lớp giấy bạc đối với loại thẻ có giấy bạc.
- Hoạt động kém trên ảnh chụp thẻ trong môi trường thiếu sáng.

### 4.2. Hướng phát triển

- Áp dụng Machine Learning vào việc huấn luyện tạo bộ dữ liệu với cả các ký tự bị khuyết nét, ký tự viết tay.
- Tận dụng sức mạnh phản cứng của các thiết bị hiện đại, với CPU và GPU mạnh hơn, có thể cho phép phần mềm tự động crop, tự động zoom theo thời gian thực vào vùng chứa mã số thẻ, có thể sử dụng thêm ML cho phần này do thông thường trên các loại thẻ, phần chứa mã số thẻ thường có thêm vùng bao quanh.

## Phụ lục

Các thẻ được đưa vào đánh giá

Số	1	2	3
Ảnh			

*Lưu ý: Trong cột “Kết quả” ký hiệu G ứng với việc đọc thành công, B ứng với không thành công*

Thời gian	Thẻ	Khoảng cách (cm)	Góc quay (độ)	Kết quả
13h	1	5	0	G
	1	10	0	G
	1	15	0	B
	2	5	0	G
	2	10	0	G
	2	15	0	G
	3	5	0	G
	3	10	0	G
	3	15	0	B
	1	5	15	G
	1	10	15	B
	1	15	15	B
	2	5	15	G
	2	10	15	G
	2	15	15	G
	3	5	15	G
	3	10	15	B
	3	15	15	B
	1	5	30	G
	1	10	30	B
	1	15	30	B
	2	5	30	G
	2	10	30	G
	2	15	30	G
	3	5	30	G
	3	10	30	B
	3	15	30	B

Thời gian	Thẻ	Khoảng cách (cm)	Góc quay (độ)	Kết quả
18h	1	5	0	G
	1	10	0	B
	1	15	0	B
	2	5	0	G
	2	10	0	G
	2	15	0	G
	3	5	0	G
	3	10	0	B
	3	15	0	B
	1	5	15	G
	1	10	15	B

	1	15	15	B
	2	5	15	G
	2	10	15	G
	2	15	15	B
	3	5	15	G
	3	10	15	B
	3	15	15	B
	1	5	30	G
	1	10	30	B
	1	15	30	B
	2	5	30	G
	2	10	30	G
	2	15	30	G
	3	5	30	G
	3	10	30	B
	3	15	30	B

Thời gian	Thẻ	Khoảng cách (cm)	Góc quay (độ)	Kết quả
21h	1	5	0	G
	1	10	0	B
	1	15	0	B
	2	5	0	B
	2	10	0	B
	2	15	0	B
	3	5	0	B
	3	10	0	B
	3	15	0	B
	1	5	15	G
	1	10	15	B
	1	15	15	B
	2	5	15	G
	2	10	15	G
	2	15	15	G
	3	5	15	G
	3	10	15	B
	3	15	15	B
	1	5	30	B
	1	10	30	B
	1	15	30	B
	2	5	30	G
	2	10	30	G



	2	15	30	G
	3	5	30	B
	3	10	30	B
	3	15	30	B