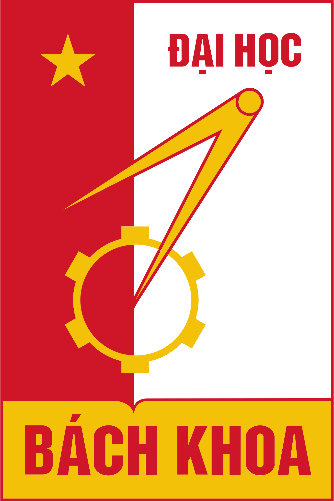
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

---o0o---



**XỬ LÝ ẢNH**

**Đề tài:**

*Nghiên cứu hệ thống nhận diện ký tự và ứng dụng chuyển đổi, dịch văn bản*

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | **TS. Mạc Thị Thoa**  **TS. Trương Công Tuấn** |
| **Sinh viên thực hiện:** | **Trần Long Quang Anh – 20184041**  **Nguyễn Trung Hiếu – 20184446**  **Lê Thu Thuỷ – 20185411** |

*Hà Nội, tháng 2 năm 2022*

**MỤC LỤC**

[**TÓM TẮT CÔNG TRÌNH** 3](#_Toc93355313)

[**CHƯƠNG 1. ĐẶT VẤN ĐỀ** 4](#_Toc93355314)

[**CHƯƠNG 2. THUẬT TOÁN OCR VÀ GIẢI PHÁP** 5](#_Toc93355315)

[**2.1. Giới thiệu kỹ thuật nhận dạng quang học OCR** 5](#_Toc93355316)

[**2.2. Cách thức hoạt động của thuật toán OCR** 5](#_Toc93355317)

[**2.3. Giải pháp thực hiện đề tài** 8](#_Toc93355318)

[**CHƯƠNG 3. KẾT QUẢ** 9](#_Toc93355319)

[**CHƯƠNG 4. KẾT LUẬN** 10](#_Toc93355320)

[**TÀI LIỆU THAM KHẢO** 11](#_Toc93355321)

# **TÓM TẮT CÔNG TRÌNH**

Xử lý ảnh là quá trình biển đổi ảnh ban đầu sang một ảnh mới với các đặc tính phù hợp mong muốn của người sử dụng. Quá trình xử lý ảnh bao gồm nhiều bước như: thu nhận ảnh, tiền xử lý, phân vùng ảnh, nhận dạng và biểu diễn ảnh. Ngoài ra việc áp dụng trí tuệ nhân tạo hay cụ thể hơn là học sâu (Deep Learning) sẽ mở rộng thêm khả năng ứng dụng của xử lý ảnh. Xử lý ảnh bằng bằng học sâu có khả năng xử lý các mẫu dữ liệu phi tuyến tính cao và cho phép khám phá cực khó khi phát hiện bởi các phương pháp thủ công.

Xử lý ảnh là một lĩnh vực mang lại rất nhiều lợi ích trong cuộc cách mạng công nghệ 4.0 hiện nay, với các sản phẩm tiêu biểu có thể kể tới như: hệ thống nhận dạng khuôn mặt, hệ thống phân loại sản phẩm, xe tự hành,… Dựa trên các ý tưởng ban đầu, nhóm chúng em đã xây dựng một chương trình xử lý ảnh ứng dụng học sâu để nhận dạng chữ viết thông qua các ảnh đầu vào được chụp bởi người sử dụng.

Nhóm thực hiện phân chia công việc như sau:

|  |  |
| --- | --- |
| Trần Long Quang Anh: | * Tìm hiểu về OCR và Tesseract, tìm hiểu các bước xử lý ảnh, tiền xử lý để cho ra kết quả tốt. * Nghiên cứu thiết kế giao diện, phụ trách chính viết giao diện sản phẩm. |
| Nguyễn Trung Hiếu: | * Tìm hiểu về OCR và Tesseract, tìm hiểu các bước xử lý ảnh, tiền xử lý để cho ra kết quả tốt. * Nghiên cứu thiết kế giao diện, chuẩn bị dữ liệu. * Viết báo cáo và hoàn chỉnh báo cáo |
| Lê Thu Thuỷ: | * Tìm hiểu về OCR và Tesseract, tìm hiểu các bước xử lý ảnh, tiền xử lý để cho ra kết quả tốt. * Tìm hiểu cách sử dụng Tesseract, phụ trách chạy thử kết quả các lần tiền xử lý. |

Các kết quả đạt được là chương trình đã chạy thành công theo thuật toán được đưa ra, nhận diện thành công các chữ viết xuất hiện trong ảnh và có kèm theo việc dịch sang các ngôn ngữ khác tuỳ theo mong muốn của người sử dụng.

# **CHƯƠNG 1. ĐẶT VẤN ĐỀ**

Ngày nay, nhu cầu về việc lấy văn bản từ hình ảnh đang ngày càng phát triển, bên cạnh sự gia tăng về nhu cầu là sự phát triển của công nghệ nhận dạng ký tự quang học (Optical Character Recognition) hay còn được gọi tắt là OCR. Đây là một công nghệ đã giúp chuyển đổi hình ảnh của chữ viết tay hoặc đánh máy thành các ký tự đã được mã hoá trong máy tính. Giả sử, chúng ta cần chỉnh sửa một số tài liệu giấy như: các bài viết trên tạp chí, tờ rơi, hoặc một tập tin PDF hình ảnh. Rõ ràng, chúng ta không thể sử dụng một máy quét để chuyển các tài liệu này thành tập tin văn bản để có thể chỉnh sửa (ví dụ như trình soạn thảo Microsoft Word).

Tất cả những gì máy quét có thể làm là tạo ra một hình ảnh hoặc một bản chụp của các tài liệu. Để giải nén và sử dụng lại giữ liệu từ tài liệu được quét, hình ảnh trong máy ảnh hoặc hình ảnh của các tập tin PDF, chúng ta cần một phần mềm OCR có chức năng được giới thiệu ở trên. Phần mềm OCR sẽ xuất ra các ký tự trên hình ảnh, ghép chúng thành từ và sau đó ghép chúng thành câu. Nhờ vậy, chúng ta có thể truy cập và chỉnh sửa nội dung của tài liệu gốc. Tương tự, với các tài liệu cổ đang bị hư hại theo thời gian và việc viết tay hay đánh máy lại những tài liệu này sẽ vô cùng tốn chi phí, thời gian cũng như không đảm bảo được độ chính xác và sự an toàn cho tài liệu nền. Việc này rất cần một công nghệ lấy từ ngữ từ tài liệu hình ảnh được chụp từ tài liệu nền.

Vì vậy, nhóm chúng em đã thống nhất xây dựng hệ thống xử lý ảnh có ứng dụng học sâu nhằm nhận diện chữ viết có trong hình ảnh do người dùng cung cấp, từ đó có thể cung cấp cho người dùng văn bản có thể chỉnh sửa và thậm chí có thể dịch sang các ngôn ngữ khác nếu người dùng có nhu cầu. Đồng thời trong tương lai, đề tài có thể được phát triển để tiến hành ứng dụng sâu rộng vào thực tế cuộc sống, phục vụ cho đời sống con người.

# **CHƯƠNG 2. THUẬT TOÁN OCR VÀ GIẢI PHÁP**

## **2.1. Giới thiệu kỹ thuật nhận dạng quang học OCR**

Nhận dạng ký tự quang học (Optical Character Recognition) là quá trình trích xuất và chuyển đổi hình ảnh văn bản dạng viết tay hoặc đánh máy, các video cũng như các tài liệu dưới dạng PDF sang định dạng ký tự đã được mã hoá bởi máy tính và có thể chỉnh sửa được. Đây là một lĩnh vực nghiên cứu về trí tuệ nhân tạo, cụ thể hơn là gắn liền với thị giác máy tính và nhận dạng mẫu. Với OCR, người dùng có thể mã hoá văn bản được in từ một hình ảnh, cho phép văn bản có thể được truy cập, sửa đổi, lưu trữ cũng như đệ trình vào không gian mạng. Từ đây, thông tin thu được từ OCR có thể được áp dụng cho hàng loạt các mục đích khác nhau mang tính chất từ cá nhân tới công cộng.

Thông thường, OCR được chia thành hai loại – OCR truyền thống và OCR viết tay. Về bản chất, cả hai đều hướng tới cùng một kết quả nhưng khác nhau về thông tin được chúng mã hoá. OCR truyền thống xem xét việc trích xuất văn bản từ các kiểu font chữ quen thuộc mà OCR đã được đào tạo trước đó. Mặt khác, chữ viết tay đối với mỗi cá nhân có sự khác biệt rõ rệt, khiến cho ngay cả AI cũng khó mã hoá hơn rất nhiều. Các chữ cái và chữ số có thể khác nhau đáng kể về kiểu dáng so với văn bản được đánh máy, ảnh hướng đến khả năng trích xuất của OCR. Chúng ta thậm chí đôi khi còn khó khăn trong việc đọc chữ viết tay của người khác chứ chưa nói đến việc AI đọc và chú thích từng từ với độ chính xác tối đa. Mặc dù còn tồn tại khó khăn và hạn chế, công nghệ OCR viết tay vẫn tiếp tục tồn tại nhưng đòi hỏi sự đào tạo bền bỉ hơn để nhận dạng mẫu tối ưu hơn.

## **2.2. Cách thức hoạt động của thuật toán OCR**

Bất kỳ một hệ thống OCR nào cũng tuân theo các bước: Pre – Processing (Tiền xử lý), Character Recognition (Nhận dạng ký tự), Post – Processing (Xử lý hậu kỳ).

**2.2.1. Tiền xử lý**

Để bắt đầu quá trình xử lý, phần cứng là một máy quét quang học thực hiện chuyển đổi hình ảnh vật lý của tài liệu thành hình ảnh kỹ thuật số. Ví dụ: nếu có một tài liệu nào đó trên một tờ giấy, máy quét quang học thực hiện sao chép tài liệu đó dưới dạng kỹ thuật số. Trong quá trình này, điều tối quan trọng với quá trình OCR là xác định các khu vực cần được quan tâm trong bức ảnh. Cụ thể, các khu vực được quan tâm là những khu vực hình ảnh có chứa văn bản và coi các khu vực “trắng” trong hình ảnh là rỗng. Quá trình trên được gọi là chuyển đổi hình ảnh thành nền (khu vực trắng, trống ký tự) và ký tự (vùng tối).

Mục tiêu của quá trình tiền xử lý nhằm cho dữ liệu không bị nhiễu để bức ảnh có độ rõ nét hơn. Điều này có liên quan tới việc loại bỏ một số điểm bất lợi như lốm đốm và có thể làm nghiêng để căn chỉnh văn bản hoặc giãn cách, tách biệt các ký tự, xoá các dòng chồng chéo và xác định bố cục ký tự để xử lý chính xác. Những kỹ thuật này giúp cải thiện kết quả tổng thể của hình ảnh với chất lượng đầu ra tốt hơn.



Hình : Mô tả quá trình tiền xử lý

**2.2.2. Nhận dạng ký tự**

Khi nền và ký tự đã được tách biệt, OCR có thể bắt đầu quá trình xác định nội dung cụ thể của từng ký tự. Các vùng tối (ký tự) xác định các số và chữ cái, phần còn lại là vùng nền của ảnh. Việc phân tích các ký tự này không được thực hiện hàng loạt mà trải qua từng phân đoạn nhỏ tại từng thời điểm. Tại từng thời điểm, hình ảnh được tách ra thành các phần nhỏ hơn để bắt đầu giải mã các ký tự trong đó. Cụ thể như một trong hai phương pháp sau:

* Nhận dạng mẫu: OCR được training với với nhiều mẫu văn bản khác nhau để có thể tự động tìm ra các kết quả phù hợp bằng cách so sánh văn bản từ tài liệu được quét với các ký tự mà nó đã học. Ký tự sau khi được phân lập thì được so sánh với một font chữ mẫu có sẵn trước đó. Quá trình được thực hiện qua từng pixel.
* Trích xuất đặc trưng: Quá trình này tập trung vào việc chia nhỏ một ký tự thành các đối tượng đặc trưng khác nhau, cụ thể là đường thẳng, đường cong và giao điểm của các đường với nhau. Tiếp đó, OCR sử dụng một tập hợp các quy tắc phát hiện đặc điểm cụ thể của chữ cái hoặc chữ số. Ví dụ, khi ta nhìn vào chữ cái “A”, chúng ta đơn giản nhìn nhận theo thói qua rằng đó là chữ “A”. Tuy nhiên, đối với AI, nó là sự kết hợp của ba đường thẳng, hai trong số ba đường thẳng đó nằm xiên và một đường thẳng nằm ngang. Bởi vì quá trình tuân theo các quy tắc nhất định nên quá trình có khả năng nhận dạng tốt hơn so với Nhận dạng mẫu đặc biệt khi xem xét với các font chữ mới, thậm chí là cả chữ viết tay.

**2.2.3. Xử lý hậu kỳ**

Sau khi tất cả các ký tự trong tài liệu được xác định, chúng sẽ được chuyển đổi thành mã ASCII có thể được lưu trữ và tiếp tục sử dụng. Tuy nhiên, bởi quá trình Nhận dạng ký tự không thể nhận dạng chính xác tuyệt đối nên đôi khi, quá trình nhận dạng phát hiện ra ký tự có sai sót. Ví dụ như ký tự “O” và ký tự “0” gần như không thể phân biệt được, đặc biệt là khi có sự liên quan tới nét chữ viết tay của con người. Đó là lí do vì sao hầu hết các hệ thống OCR hiện nay đều phải thực hiện quá trình Xử lý hậu kỳ.

Giai đoạn này kiểm tra tính chính xác của văn bản đã được trích xuất bằng cách so sánh nó với một từ vựng đã được thiết lập trước đó. Phần mềm thực hiện phân tích các “hàng xóm” gần với từ đó nhất (các “hàng xóm” là các từ thông dụng được sử dụng). Ví dụ khi trích xuất được một cụm từ “clout – based”, phần mềm sẽ nhận diện lại thành “cloud – based” bởi vì “cloud – based” được sử dụng phổ biến hơn.

## **2.3. Giải pháp thực hiện đề tài**

### 2.3.1. Quá trình đào tạo dữ liệu:

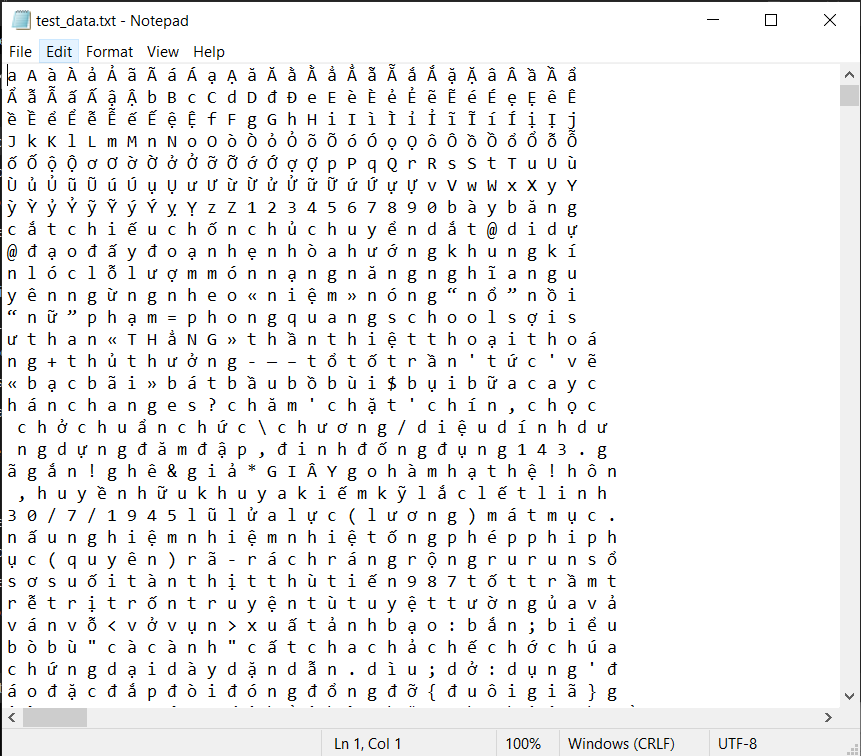
1. Chuẩn bị trước khi train:

* Công cụ để train Tesseract – [**jTessBoxEditor**](https://sourceforge.net/projects/vietocr/files/jTessBoxEditor/).
* File font chữ cần đào tạo có đuôi là ttf. Ví dụ: TimeNewRoman.ttf
* File văn bản khoảng 600KB đến 1MB để học được nhiều kí tự khác nhau.
* Cài Java Runtime Environment (JRE) là một lớp phần mềm cung cấp các dịch vụ cần thiết để thực thi những ứng dụng Java.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình : File font chữ TimeNewRoman.ttf



Hình : File văn bản

1. Tiến hành train

* Vào Tab **TIFF/Box Generator**, chọn input là file đã chuẩn bị:

A screenshot of a computer

Description automatically generated

Hình : Chọn 1 file đã chuẩn bị

* Kết quả hiện thị như sau:

A picture containing text, screenshot, people

Description automatically generated

Hình : Kết quả sau khi chọn file

* Chọn thư mục lưu font, chọn font chữ, kích thước, tracking, leading sau đó **Generate**.

Graphical user interface, text, application, Word

Description automatically generated

Hình : Thông báo generate thành công

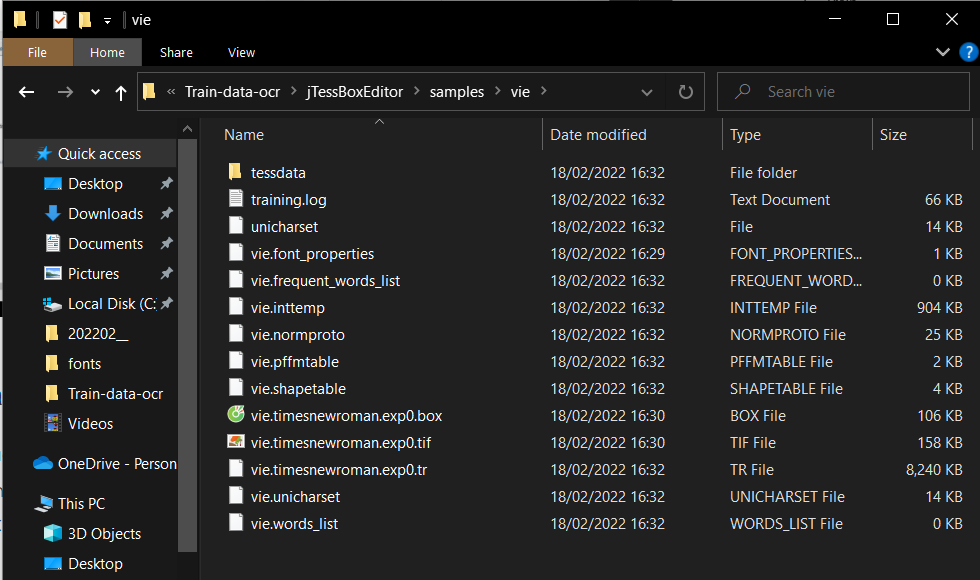
1. Train

* Sang tab **Trainer**, chọn **Training Data** trỏ vào thư mục để font vừa tạo, chọn **Train with Existing Box** và nhấn **Run**.

Graphical user interface, text, application, email

Description automatically generated

* Hoàn thành ta thu được 1 bộ font để sử dụng.

A screenshot of a computer

Description automatically generated with medium confidence

Hình : Bộ font sau khi đã được tạo thành công

1. Kiểm tra lại các kí tự đã học được

* Vào lại công cụ **jTessBox** kiểm tra lại sẽ thấy những kí tự đã học được.

Graphical user interface, text, application

Description automatically generated

Như vậy, với tương lai thử phát hiện chữ viết tay ta sẽ tự tạo 1 phông chữ viết tay sau đó train data.

### 2.3.2. Các phương pháp tiền xử lý:

|  |  |
| --- | --- |
| **Remove noise: cv2.mediaBlur()**  Lấy trung vị của tất cả các pixel dưới khu vực hạt nhân và phần tử trung tâm được thay thế bằng giá trị trung bình này. Điều này có hiệu quả cao chống lại tiếng ồn muối tiêu trong một hình ảnh.  # noise removal  def remove\_noise(image):      return cv2.medianBlur(image, 5) | Ảnh có chứa văn bản  Mô tả được tạo tự động |
| **Erosion: cv2.erode()**  Phương pháp được sử dụng để thực hiện xói mòn trên hình ảnh. Ý tưởng cơ bản về xói mòn chỉ giống như xói mòn đất, nó làm xói mòn ranh giới của vật thể tiền cảnh  # erosion  def erode(image):      kernel=np.ones((5, 5), np.uint8)      return cv2.erode(image, kernel, iterations=1) | Ảnh có chứa văn bản  Mô tả được tạo tự động |
| **Dilation: cv2.dilate()**  Bởi vì, xói mòn loại bỏ tiếng ồn trắng, nhưng nó cũng thu nhỏ đối tượng. Vì vậy, cần làm giãn nó vì tiếng ồn đã được loại bỏ  # dilation  def dilate(image):      kernel=np.ones((5, 5), np.uint8)      return cv2.dilate(image, kernel, iterations=1) | Ảnh có chứa ngoài trời, trắng  Mô tả được tạo tự động |

* Từ kết quả cho thấy với ảnh chữ nhỏ việc sử dụng Remove noise, Erosion, Dilation cho kết quả không tốt, không phù hợp ở cụ thể bài toán đưa ra.

### 2.3.3. Phương pháp thực hiện:

Chi tiết các bước thực hiện được trình bày ở phần dưới đây, giả sử ảnh đầu vào muốn trích xuất thông tin có dạng:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Resize lại kích thước ảnh đầu vào:

def image\_resize(image, width = None, height = None, inter = cv2.INTER\_AREA):  
 # initialize the dimensions of the image to be resized and  
 # grab the image size  
 dim = None  
 (h, w) = image.shape[:2]  
  
 # if both the width and height are None, then return the  
 # original image  
 if width is None and height is None:  
 return image  
  
 # check to see if the width is None  
 if width is None:  
 # calculate the ratio of the height and construct the  
 # dimensions  
 r = height / float(h)  
 dim = (int(w \* r), height)  
  
 # otherwise, the height is None  
 else:  
 # calculate the ratio of the width and construct the  
 # dimensions  
 r = width / float(w)  
 dim = (width, int(h \* r))  
  
 # resize the image  
 resized = cv2.resize(image, dim, interpolation = inter)  
  
 # return the resized image  
 return resized

Kết quả của ảnh hậu resize:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Tiền xử lý điều chỉnh độ sáng tương phản:

def adjust\_contrast\_brightness(img, contrast:float=1.0, brightness:int=0):  
 *"""  
 Adjusts contrast and brightness of an uint8 image.  
 contrast: (0.0, inf) with 1.0 leaving the contrast as is  
 brightness: [-255, 255] with 0 leaving the brightness as is  
 """* brightness += int(round(255\*(1-contrast)/2))  
 return cv2.addWeighted(img, contrast, img, 0, brightness)

Kết quả của ảnh sau tiền xử lý:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Chuyển ảnh màu thành ảnh xám

# get grayscale image - anh xám  
def get\_grayscale(image):  
 return cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

Ảnh xám thu được:

Ảnh có chứa văn bản

Mô tả được tạo tự động

AdaptiveThreshold:

# do adaptive threshold on gray image  
thresh = cv2.adaptiveThreshold(contrast1, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 21, 15)  
# make background of input white where thresh is white  
result = image.copy()  
result[thresh==255] = (255,255,255)

def importing(image):  
 image = cv2.imread(image)  
 image = image\_resize(image, height = 600)  
 contrast1 = adjust\_contrast\_brightness(image, contrast=1.0, brightness=0)  
 gray = get\_grayscale(contrast1) # tao anh gray  
# do adaptive threshold on gray image  
 thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 21, 15)  
# make background of input white where thresh is white  
 result = image.copy()  
 result[thresh == 255] = (255, 255, 255)  
 return result

Ảnh có chứa văn bản

Mô tả được tạo tự động

Chuyển sang dạng text:

text = pytesseract.image\_to\_string(Image.open(filename), lang='vie')

Sửa lỗi xuống dòng:

def clean\_text(text):  
 ret = text.replace("\n\n","\n")  
 ret = ret.replace("\n\n", "\n")  
 return ret

Hàm xuất ra text:

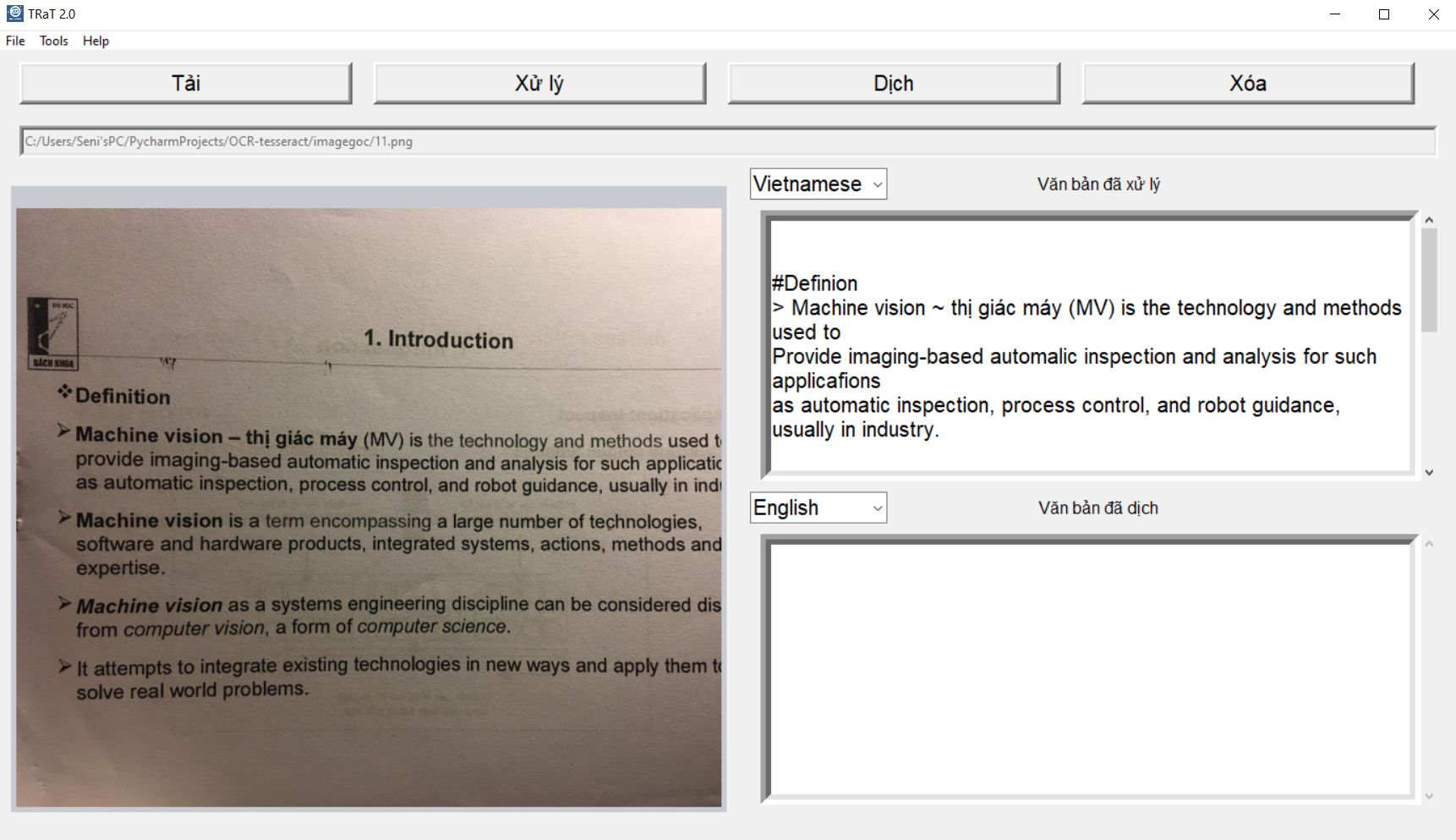
def exporting(image):  
 # Ghi tạm ảnh xuống ổ cứng để sau đó apply OCR  
 filename = "{}.png".format(os.getpid())  
 cv2.imwrite(filename, image)  
  
 # Load ảnh và apply nhận dạng bằng Tesseract OCR  
 text = pytesseract.image\_to\_string(Image.open(filename), lang='vie')  
  
 # Xóa ảnh tạm sau khi nhận dạng  
 os.remove(filename)  
 return text  
  
def clean\_text(text):  
 ret = text.replace("\n\n","\n")  
 ret = ret.replace("\n\n", "\n")  
 return ret  
  
def Ham\_cua\_Thuy(link):  
 image2 = importing(link)  
 text2 = exporting(image2)  
 text2=clean\_text(text2)  
 return text2

Dịch văn bản:

def translate():  
 t = Translator()  
 inp = inp\_text.get(1.0, END)  
 inp\_lang = inp\_combobox.get()  
 out\_lang = out\_combobox.get()  
  
 try:  
 out = t.translate(inp, src=inp\_lang, dest=out\_lang)  
 out = out.text  
 out\_text.config(state=NORMAL)  
 out\_text.delete(1.0, END)  
 out\_text.insert(END, out)  
 out\_text.config(state=DISABLED)  
   
 except ValueError:  
 messagebox.showerror("Error", '''Kiểm tra lại ngôn ngữ bạn nhé!!!\  
 Please check your language!!!\n-\_-''')  
 except TypeError:  
 messagebox.showerror("Error", '''Kiểm tra lại ngôn ngữ bạn nhé!!!\  
 Please check your language!!!\n-\_-''')

# **CHƯƠNG 3. KẾT QUẢ**

Để có thể dễ hơn trong việc tương tác với người dùng, nhóm chúng em quyết định xây dựng một giao diện người dùng với một số chức năng cơ bản như: Dẫn link tới file ảnh cần trích xuất, trích xuất ảnh, lưu ảnh và có thêm chức năng phiên dịch văn bản sang ngôn ngữ khác tuỳ vào mong muốn của người sử dụng. Dưới đây là một số hình ảnh cho kết quả của nhóm.



Độ chính xác được tính bằng tay giữa kí tự sai và kí tự đúng lên đến 92%, được cho là chính xác cao.

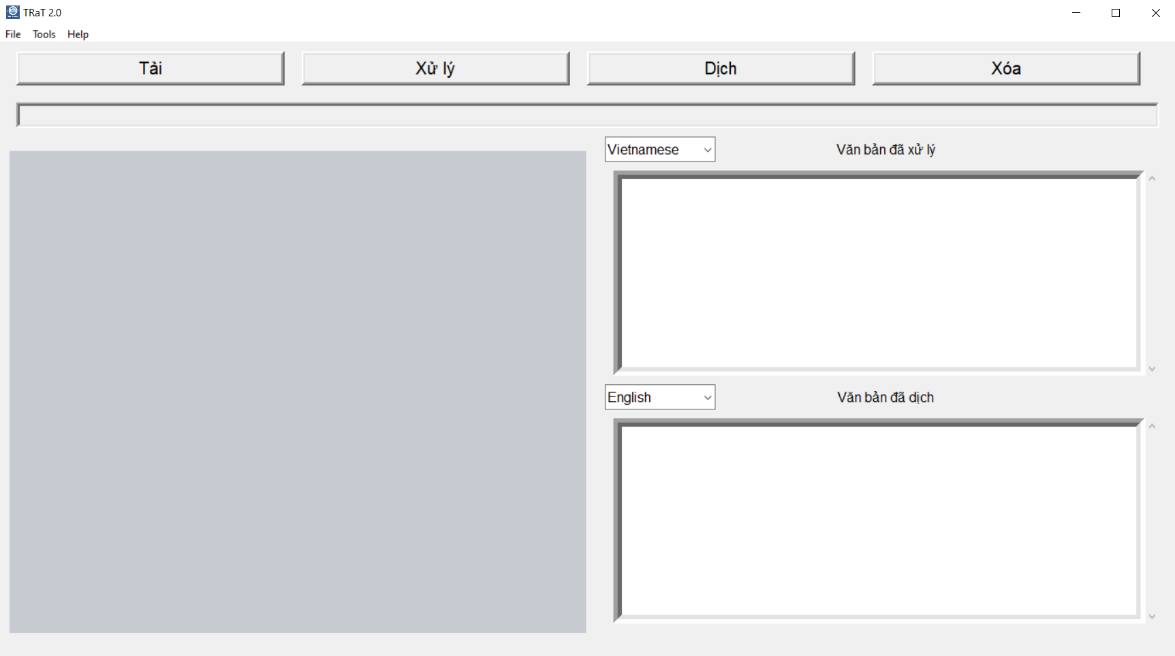
Tuy nhiên, với ảnh đầu vào không tốt như ảnh nghiêng quá nhiều, chất lượng ảnh không tốt, ảnh mờ sẽ không thu được kết quả tốt, nhận diện ký tự bị sai rất nhiều

Ảnh có chứa văn bản

Mô tả được tạo tự động

Giao diện mở đầu bao gồm các PushButton: Tải, Xử lý, Dịch, Xóa

* Tải có chức năng chọn ảnh mà mình mong muốn chuyển sang text
* Xử lý: để bắt đầu chuyển đổi từ ảnh sang các kí tự
* Dịch: để dịch từ văn bản vừa thu được sang các ngôn ngữ khác có thể chọn các loại ngôn ngữ
* Xóa: để xóa hiện thị các kết quả vừa mới thu được.



Hình : Hình ảnh giao diện mở đầu sản phẩm

Ngoài ra còn có 3 phần khung để hiện thị, khung hiển thị ảnh muốn xử lý, khung hiện thị văn bản thu được sau khi nhận diện và khung hiện thị văn bản sau khi dịch

Chương trình bao gồm các tab chức năng như là File, Tools, Help.

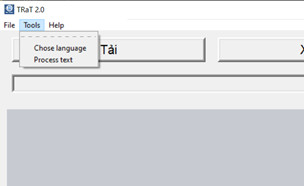
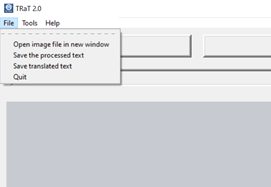
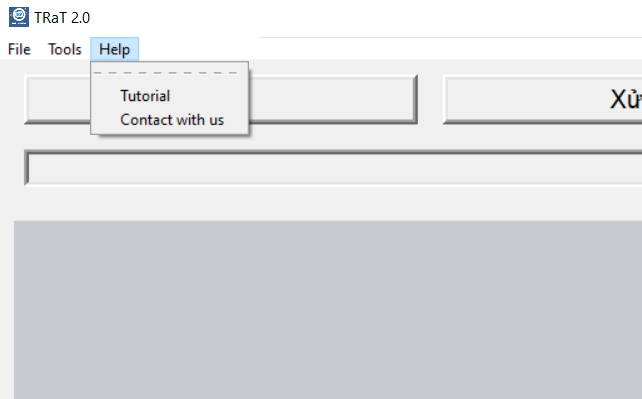
Tab chức năng File có các tính năng sau:

* Open image file in new window: mở file ảnh ở cửa sổ mới.
* Save the processed text: Lưu văn bản đã xử lý.
* Save the translated text: Lưu văn bản sau khi dịch.
* Quit: Thoát chương trình.

Tab chức năng Tools có tính năng:

* Choose language: Chọn ngôn ngữ hiện thị chương trình
* Process text: Xử lý văn bản, sửa lỗi văn bản như lỗi viết hoa,…

Tab chức năng Help bao gồm Tutorial chứa hướng dẫn sử dụng chương trình và Contact with us chứa thông tin liên lạc nhà phát triển sản phẩm.



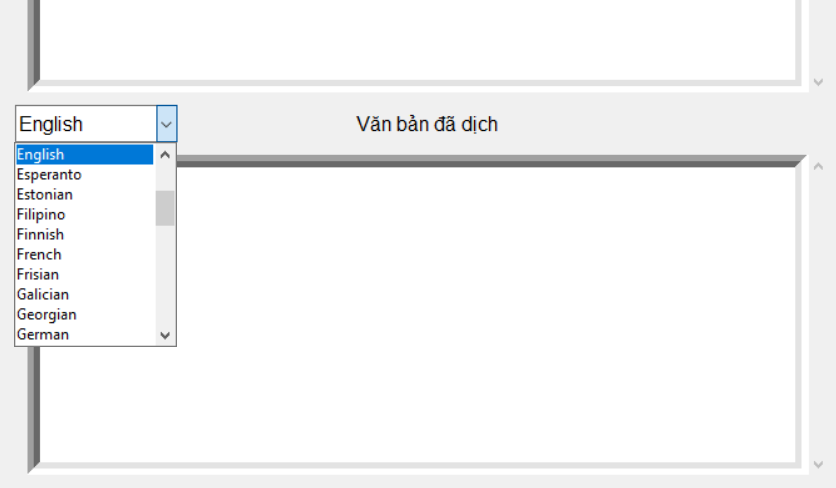
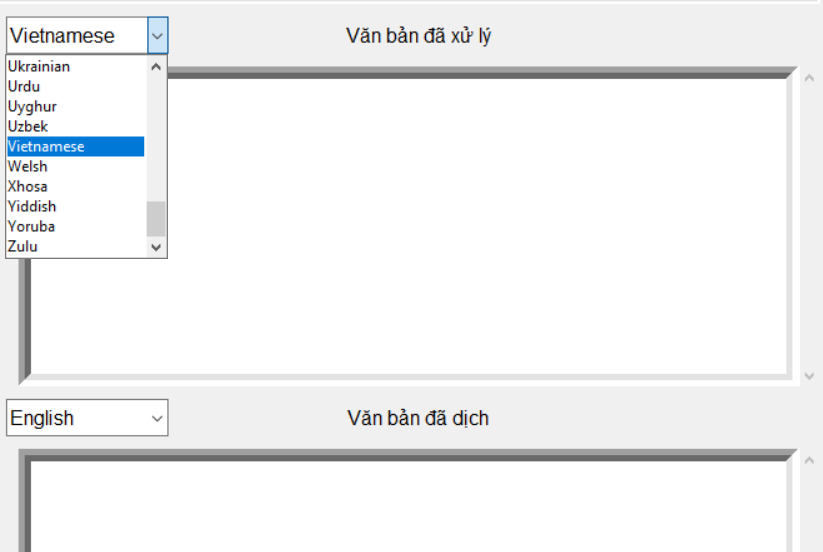
(a)

(b)

(c)

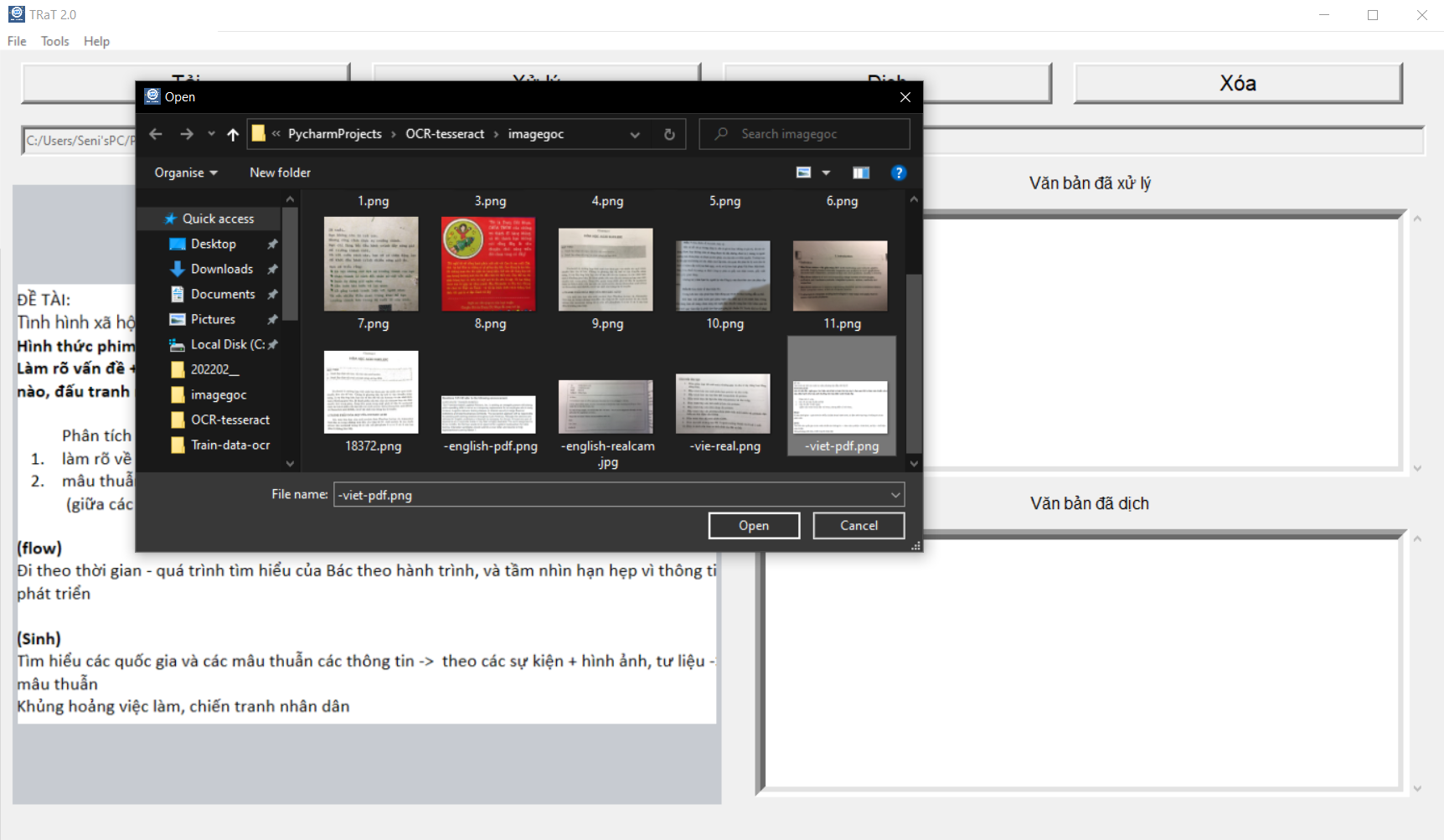
Hình : (a) Tab File (b) Tab Tools (c) Tab Help

Tính năng dịch qua các ngôn ngữ khác



Hình : Tính năng dịch qua ngôn ngữ khác

Chọn ảnh để xử lý – Kết quả thu được:



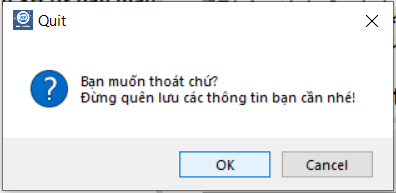
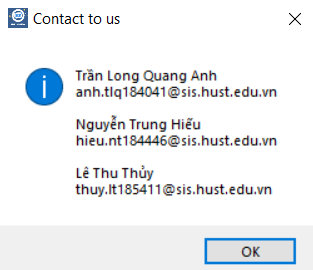
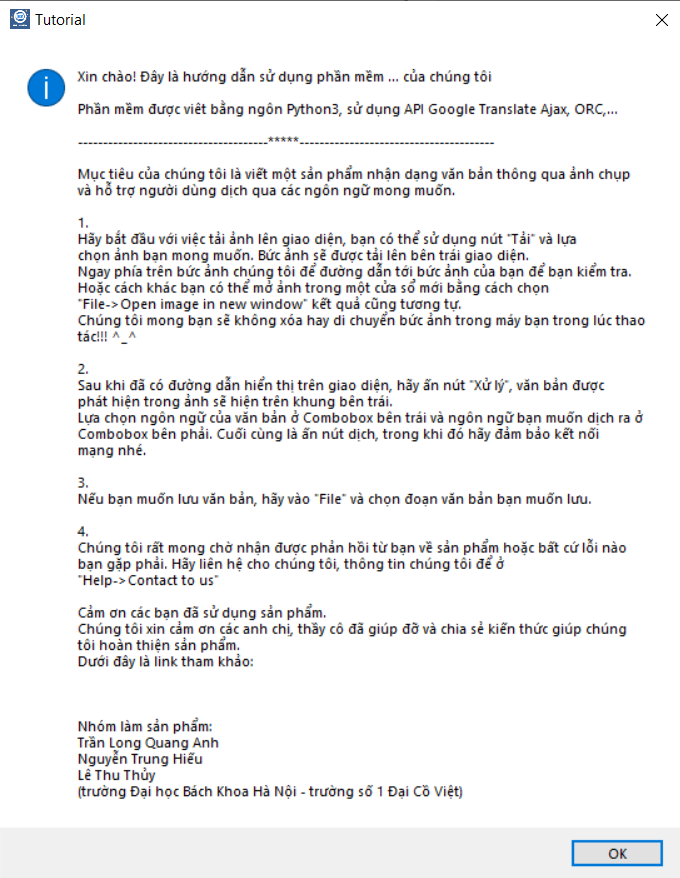
Hình : Chọn ảnh để xử lý

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình : Kết quả sau xử lý

Tutorial và Contact with us, Lời nhắc trước khi tắt chương trình



Hình : (a) Tutorial, (b) Contact to us, (c) Quit

# **CHƯƠNG 4. KẾT LUẬN**

Thông qua bài tập lớn của môn học, chúng em có cơ hội tổng hợp, vận dụng những kiến thức đã học cũng như tìm hiểu thêm những kiến thức mới, bổ ích về xử lý ảnh và học sâu Deep Learning. Bên cạnh đó, chúng em cũng hiểu thêm về quá trình thiết kế, xây dựng một chương trình nhận chữ viết nhằm ứng dụng cho bài toán chuyển đổi văn bản dưới dạng hình ảnh sang dạng văn bản số hoá có thể được lưu trữ và sử dụng về sau, hay mục tiêu xa hơn là ứng dụng rộng rãi trong đời sống con người.

Trong tương lai, nhóm sẽ hướng tới phát triển hệ thống phần cứng, tối ưu chương trình hơn nữa để sử dụng được trong các hệ thống nhúng. Đồng thời, phát triển và bổ sung một số tính năng bảo mật văn bản để có thể ứng dụng trong các hệ thống lưu trữ tài liệu quan trọng như công văn, hợp đồng, nghiên cứu phát triển để nhận dạng chữ viết tay,…

Bên cạnh đó Tesseract vẫn còn nhiều hạn chế. Tesseract hoạt động tốt nhất khi có sự phân đoạn rõ ràng của văn bản và nền, trong thực tế điều này rất khó khăn. Một số hạn chế cụ thể của Tesseract như OCR không chính xác như một số giải pháp có sẵn khác, không hoạt động tốt với các hình ảnh bị ảnh hưởng bởi hiện vật, bản quét chất lượng kém sẽ tạo ra OCR chất lượng kém …

# **TÀI LIỆU THAM KHẢO**

[1] What is Optical Character Recognition (OCR): Overview and use cases.

[2] Nguyễn Văn Vương, “Số hoá và rút trích tự động thông tin công văn”, 2019.

[3] <https://nanonets.com/blog/ocr-with-tesseract/>

[4] <https://vncoder.vn/bai-viet/ocr-dua-tren-nen-tang-deep-learning-de-nhan-dang-van-ban-trong-hinh-anh-canh-tu-nhien>

[5] Richard Szeliski. Computer Vision: Algorithms and Applications. Springer, New York, 2nd edition, 2022

[6] Adrian Rosebrock. Rotate images (correctly) with OpenCV and Python.  
https://www.pyimagesearch.com/2017/01/02/rotate-imagescorrectly-with-opencv-and-python/. 2017

[7] Adrian Rosebrock, PhD, “*OCR with OpenCV, Tesseract, and Python”,* 2020