TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

ĐỒ ÁN
PROJECT 1

NGÀNH CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI

1. **(*WINDOWS*) NHẬN DIỆN SỐ VIẾT TAY (CHỮ NẾU CÓ THỂ) SỬ DỤNG *TESSERACT***
2. **(*ANDROID*) NHẬN DIỆN KÍ TỰ IN TRONG *VIDEO REALTIME* VÀ MÃ HÓA THÀNH *QRCODE* SỬ DỤNG *MOBILE VISION API***

Sinh viên thực hiện: **Nguyễn Hữu Tráng**
Lớp CNTT1.02 -K61
Giáo viên hướng dẫn: ThS. **Nguyễn Đức Tiến**

HÀ NỘI 11-2018

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN MÔN HỌC

1. Thông tin về sinh viên

Họ và tên sinh viên: Nguyễn Hữu Tráng

Điện thoại liên lạc: 0352408617 Email: nguyenhuutrang.it@gmail.com

Lớp: CNTT1.02-K61 Hệ đào tạo: đại học chính quy

Thời gian làm đồ án: Từ ngày 14/09/2018 đến ngày 16/11/2018

2. Mục đích nội dung của đồ án

- Tìm hiểu về bài toán text recognition:
    - *Trên Windown:* Tìm hiểu và sử dụng tesseract để nhận diện số viết tay
    - *Trên Android:* Tìm hiểu và sử dụng Mobile Vision API để xây dựng một ứng dụng cho phép nhận diện chữ in (realtime) trong video thu được từ camera của thiết bị, hiển thị chữ nhận diện được ngay phía dưới vị trí của nó trong ảnh. Sau đó mã hóa chuỗi này thành QR-code
- Làm quen với github: Sử dụng được git ở mức độ cơ bản. Biết cách sử dụng branch để phân ra nhiều nhánh trong quá trình làm project

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu: Tesseract và Vision API cho bài toán text recognition ; sử dụng github để quản lý mã nguồn và các phiên bản phát triển
- Phân tích/ Thiết kế/Phát triển/Triển khai/Nâng cấp:
    - Với bài toán nhận diện số viết tay:
        - Ban đầu: Chỉ có 1 khung điểm hình chữ nhật, sử dụng tesseract để nhận diện
        - Tiếp theo chia ô cũ làm 2. Một nửa để chứa MSSV, nửa còn lại là chứa điểm
        - Tiến hành traning bổ sung để nâng cao khả năng nhận diện của tesseract (chưa làm được)
    - Với bài toán sử dụng Vision API:
        - Ban đầu: tìm hiểu về Vision API của google, liên quan đến bài toán text recognition
        - Tiếp theo làm 1 ứng dụng đơn giản để nhận diện trên dòng camera
        - Tách branch, 1 branch để nâng cấp chức năng text recognition, 1 branch để thêm chức năng hamburger menu
- Kiểm thử:
    - Với bài toán nhận diện số viết tay: test trên các bài test tự tạo, các file test cũng được upload cùng mã nguồn trên github
    - Với bài toán sử dụng Vision API: Test với các mặt chữ trên sách, máy chiếu, biển quảng cáo …

- Tổng kết và đánh giá.

4. Lời cam đoan của sinh viên:

Tôi – *Nguyễn Hữu Tráng* - cam kết đồ án là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của giáo viên. Các kết quả nêu trong đồ án là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

	Hà Nội, ngày 21 tháng 11 năm 2018 Tác giả đồ án

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của đồ án

	Hà Nội, ngày ... tháng ... năm ..... Giáo viên hướng dẫn

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT VÀ THUẬT NGỮ

Số thứ tự	Từ viết tắt	Ý nghĩa
1	ĐA	Đồ án
2	OS	Operating System – Hệ điều hành
3	SDK	Software Development Kit – Bộ công cụ phát triển phần mềm

PHẦN I: ĐẶT VẤN ĐỀ VÀ ĐỊNH HƯỚNG GIẢI PHÁP

1.1. Tổng quan

1.1.1. Các vấn đề, khó khăn hiện tại

- Đối với bài toán nhận diện số viết tay:
  - Xử lý ảnh đầu vào: tính góc lệch và xoay, cắt đúng vùng ảnh có chứa số cần nhận diện, khử nhiễu ảnh, tách từng chữ số một để nhận diện …
  - Về thuật toán nhận diện: chưa có đủ khả năng để tự xây dựng một model học máy để nhận diện số viết tay
- Đối với bài toán sử dụng Vision API:
  - Định hướng ban đầu là thực hiện lời gọi tới Google Translate để dịch và hiển thị dòng chữ bằng tiếng Việt ngay bên dưới nhưng không thực hiện được do
    - Không có API Key để sử dụng dịch vụ translate trên Cloude API
    - Với lựa chọn gửi request lên các server có chức năng translate và phân tích html trả về không thực hiện được. Lý do là mặc dù lấy được dữ liệu với hầu hết các trang web, tuy nhiên không hiểu sao không nhận được kết quả trả về khi gửi request tới những trang như:
      - https://translate.google.com/
      - http://translate.googleapis.com/translate_a/single?sl=en&tl=vi&dt=t&q=hello

1.1.2. Mục tiêu cần đạt được

- Với bài toán nhận diện số viết tay: Tỉ lệ nhận diện đúng lớn hơn 80%
- Với bài toán sử dụng Vision API:
  - Với chữ in tiếng anh: tỉ lệ nhận diện đúng phải >95%
  - Với chữ in tiếng việt không dấu: tỉ lệ nhận diện đúng > 90%
  - Với chữ in tiếng việt có dấu: tỉ lệ nhận diện đúng > 75%

1.1.3. Lựa chọn và định hướng thiết kế

- Với bài toán nhận diện số viết tay:
  - Sử dụng OpenCV để xử lý ảnh
  - Sử dụng tesseract làm core thuật toán nhận diện
  - Chương trình viết bằng python, sử dụng lập trình hàm, chạy bằng cmd với các tham số được truyền vào
- Với bài toán sử dụng Vision API:
  - Xây dựng ứng dụng bằng Android Studio

- Sử dụng Mobile Vision API làm core thuật toán nhận diện chữ in
- Thay thế bài toán dịch thuật gặp khó khan bằng bài toán "nhân dạng chữ xong → hiển thị ảnh QRCode tương ứng với đoạn chữ đó"

1.2. Các framework/thư viện lập trình

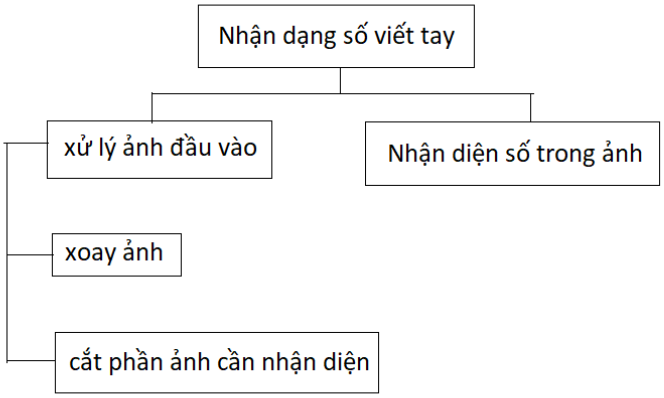
➔ Chi tiết về các thư viện đều có trong file readme trên github

- Đối với bài toán nhận diện chữ viết tay:
  - OpenCV : để xử lý ảnh
  - pytesseract: làm core thuật toán nhận diện
  - Github:https://github.com/nguyenhuutrang/Project-1-Handwritten-digit-recognition
- Đối với bài toán sử dụng Vision API:
  - Sử dụng Mobile Vision API để nhận diện chữ in, giải mã QRcode
  - Sử dụng Zxing để mã hóa chuỗi nhận diện được về dạng QRcodee
  - Github: https://github.com/nguyenhuutrang/ApplyVisionAPI-Project1

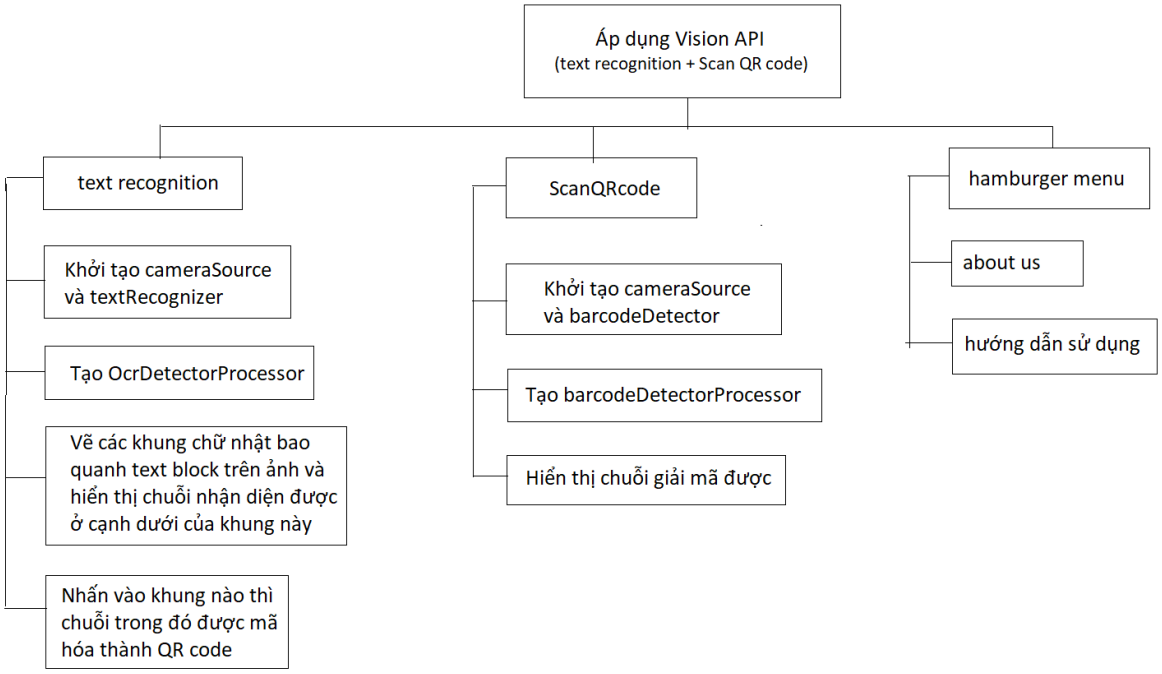
PHẦN II: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Các chức năng

- Đối với bài toán nhận diện số viết tay:



- Đối với bài toán sử dụng Vision API:



2.2. Thuật toán cốt lõi

- Đối với bài toán nhận diện số viết tay:

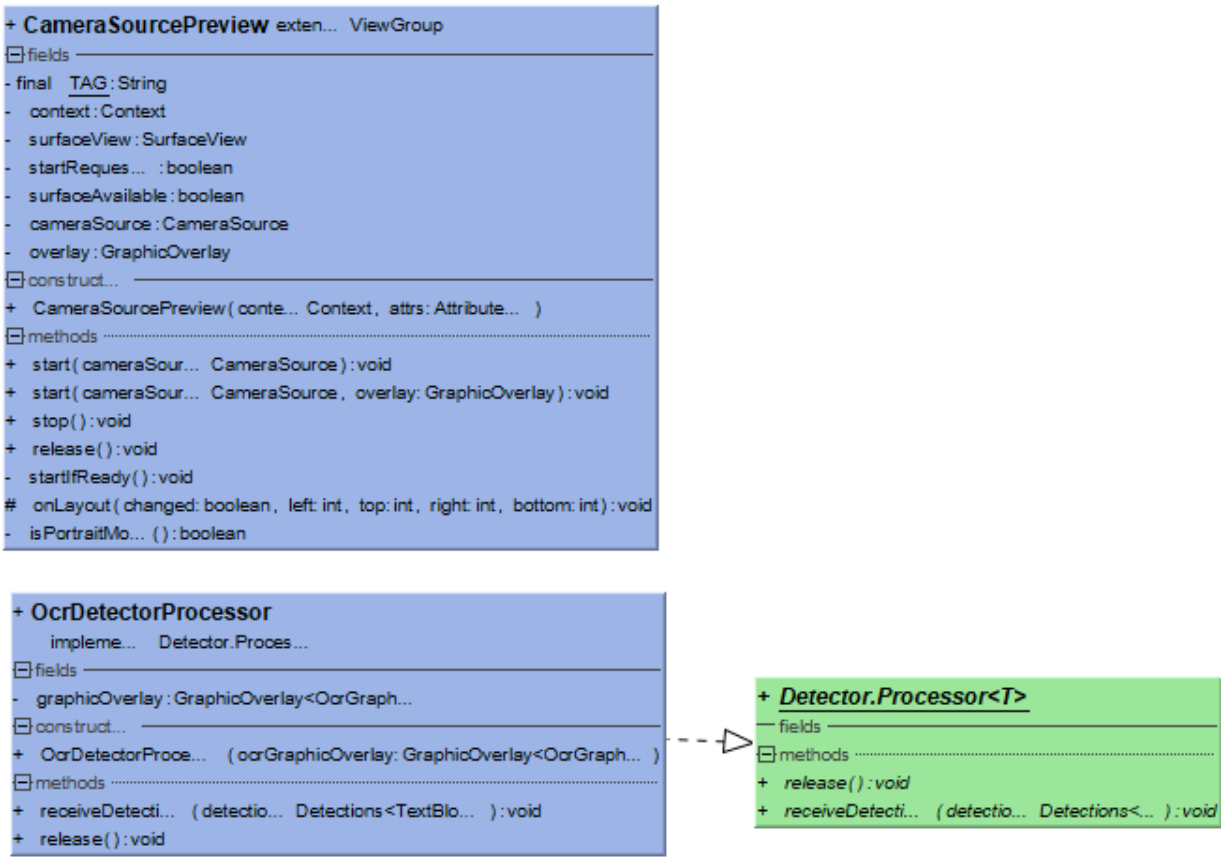
  →Sử dụng OpenCV xử lý ảnh

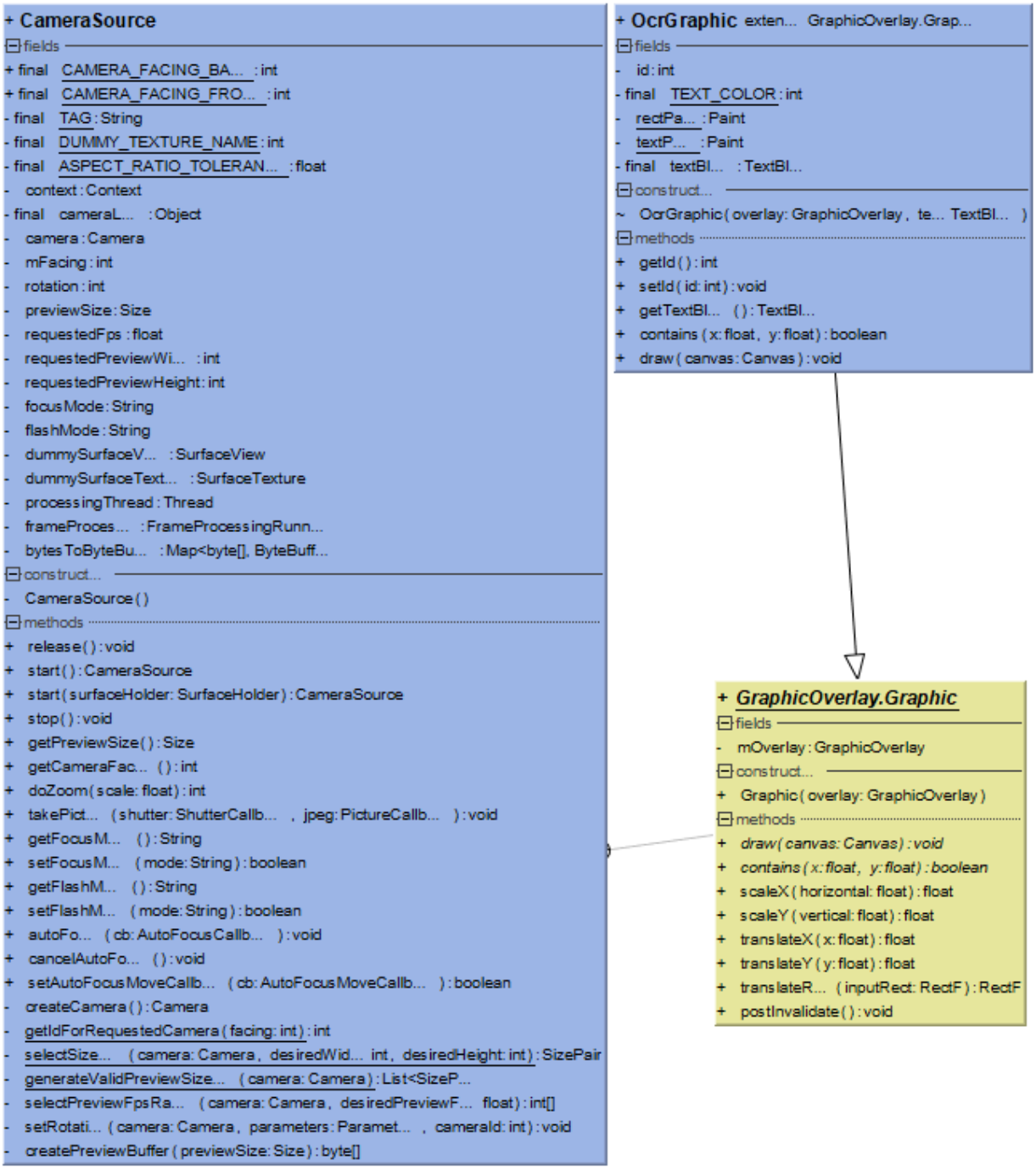
  →Sử dụng tesseract làm core thuật toán nhận diện

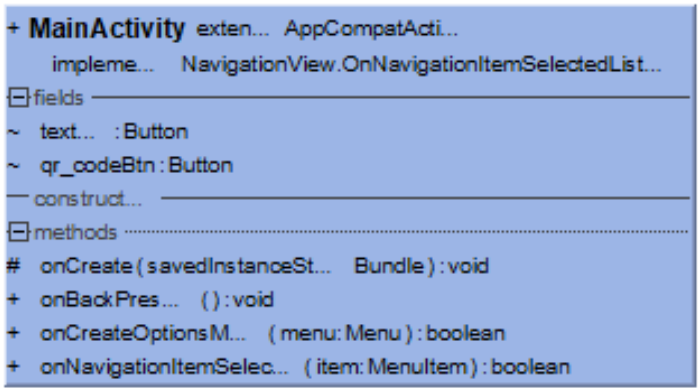
- Đối với bài toán sử dụng Vision API:
  →Sử dụng Mobile Vision API để phát hiện, nhận dạng text block, giải mã QRcode
  →Sử dung Zxing để mã hóa chuỗi nhận diện được thành QRcode

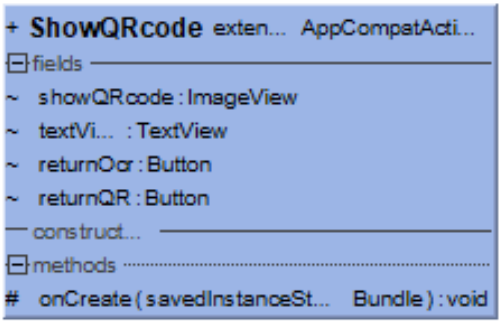
2.3. Các lớp đối tượng

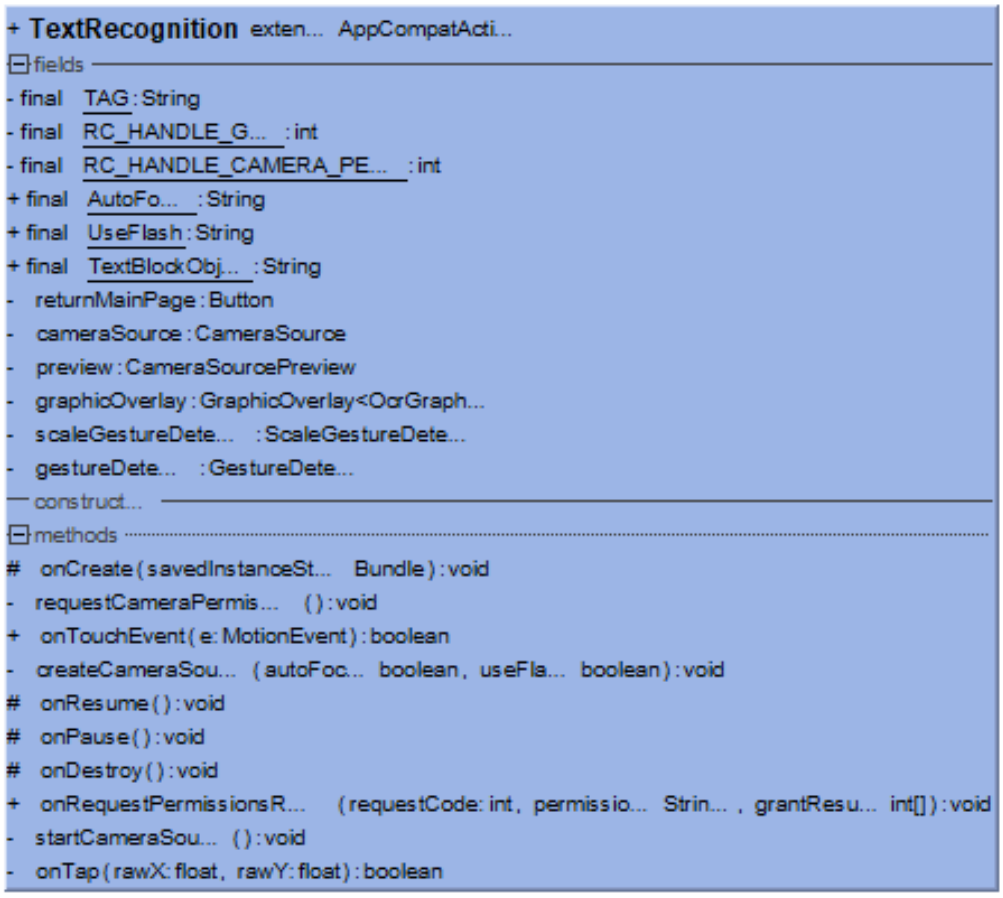
- Đối với bài toán sử dụng Vision API:

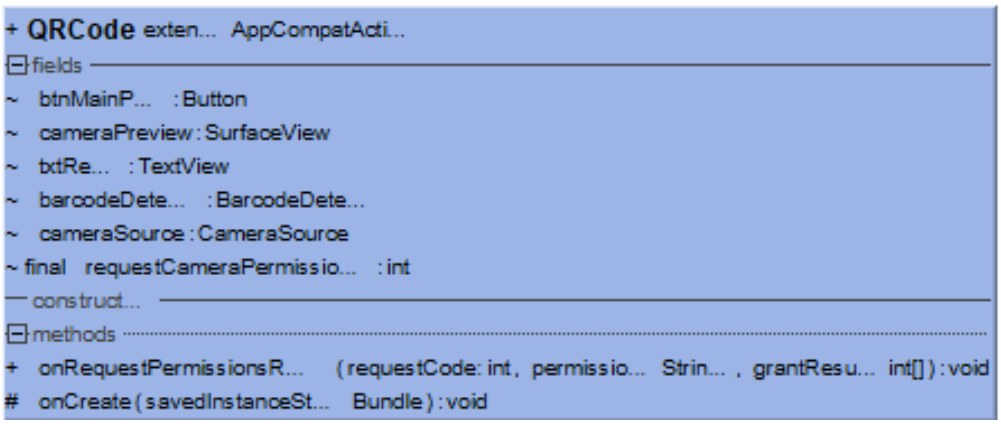



+ CameraSource
fields
+ final CAMERA_FACING_BA... : int
+ final CAMERA_FACING_FRO... : int
- final TAG : String
- final DUMMY_TEXTURE_NAME : int
- final ASPECT_RATIO_TOLERAN... : float
- context : Context
- final cameraL... : Object
- camera : Camera
- mFacing : int
- rotation : int
- previewSize : Size
- requestedFps : float
- requestedPreviewWi... : int
- requestedPreviewHeight : int
- focusMode : String
- flashMode : String
- dummySurfaceV... : SurfaceView
- dummySurfaceText... : SurfaceTexture
- processingThread : Thread
- frameProces... : FrameProcessingRunn...
- bytesToByteBu... : Map<byte[], ByteBuff...
construct...
- CameraSource ( )
methods
+ release ( ) : void
+ start ( ) : CameraSource
+ start (surfaceHolder: SurfaceHolder) : CameraSource
+ stop ( ) : void
+ getPreviewSize ( ) : Size
+ getCameraFac... ( ) : int
+ doZoom (scale: float) : int
+ takePict... (shutter: ShutterCallb... , jpeg: PictureCallb...) : void
+ getFocusM... ( ) : String
+ setFocusM... (mode: String) : boolean
+ getFlashM... ( ) : String
+ setFlashM... (mode: String) : boolean
+ autoFo... (cb: AutoFocusCallb...) : void
+ cancelAutoFo... ( ) : void
+ setAutoFocusMoveCallb... (cb: AutoFocusMoveCallb...) : boolean
- createCamera ( ) : Camera
- getIdForRequestedCamera (facing: int) : int
- selectSize... (camera: Camera , desiredWid... int , desiredHeight: int) : SizePair
- generateValidPreviewSize... (camera: Camera) : List<SizeP...
- selectPreviewFpsRa... (camera: Camera , desiredPreviewF... float) : int[]
- setRotati... (camera: Camera , parameters: Paramet... , cameraId: int) : void
- createPreviewBuffer (previewSize: Size) : byte[]
+ OcrGraphic exten... GraphicOverlay.Grap...
fields
- id : int
- final TEXT_COLOR : int
- rectPa... : Paint
- textP... : Paint
- final textBl... : TextBl...
construct...
~ OcrGraphic (overlay: GraphicOverlay , te... TextBl...)
methods
+ getId ( ) : int
+ setId (id: int) : void
+ getTextBl... ( ) : TextBl...
+ contains (x: float , y: float) : boolean
+ draw (canvas: Canvas) : void
+ GraphicOverlay.Graphic
fields
- mOverlay : GraphicOverlay
construct...
+ Graphic (overlay: GraphicOverlay)
methods
+ draw (canvas: Canvas) : void
+ contains (x: float , y: float) : boolean
+ scaleX (horizontal: float) : float
+ scaleY (vertical: float) : float
+ translateX (x: float) : float
+ translateY (y: float) : float
+ translateR... (inputRect: RectF) : RectF
+ postInvalidate ( ) : void


+ MainActivity exten... AppCompatActi...
impleme... NavigationView.OnNavigationItemSelectedList...
fields
~ text... : Button
~ qr_codeBtn : Button
construct...
methods
onCreate (savedInstanceSt... Bundle) : void
+ onBackPres... ( ) : void
+ onCreateOptionsM... (menu: Menu) : boolean
+ onNavigationItemSelec... (item: MenuItem) : boolean


+ ShowQRcode exten... AppCompatActi...
fields
~ showQRcode : ImageView
~ textVi... : TextView
~ returnOcr : Button
~ returnQR : Button
construct...
methods
onCreate (savedInstanceSt... Bundle) : void


+ TextRecognition exten... AppCompatActi...
fields
- final TAG : String
- final RC_HANDLE_G... : int
- final RC_HANDLE_CAMERA_PE... : int
+ final AutoFo... : String
+ final UseFlash : String
+ final TextBlockObj... : String
- returnMainPage : Button
- cameraSource : CameraSource
- preview : CameraSourcePreview
- graphicOverlay : GraphicOverlay<OcrGraph...
- scaleGestureDete... : ScaleGestureDete...
- gestureDete... : GestureDete...
construct...
methods
onCreate (savedInstanceSt... Bundle) : void
- requestCameraPermis... ( ) : void
+ onTouchEvent (e: MotionEvent) : boolean
- createCameraSou... (autoFoc... boolean, useFla... boolean) : void
onResume ( ) : void
onPause ( ) : void
onDestroy ( ) : void
+ onRequestPermissionsR... (requestCode: int, permissio... Strin... , grantResu... int[]) : void
- startCameraSou... ( ) : void
- onTap (rawX: float, rawY: float) : boolean


+ QRCode exten... AppCompatActi...
fields
~ btnMainP... : Button
~ cameraPreview : SurfaceView
~ txtRe... : TextView
~ barcodeDete... : BarcodeDete...
~ cameraSource : CameraSource
~ final requestCameraPermissio... : int
construct...
methods
+ onRequestPermissionsR... (requestCode: int, permissio... Strin... , grantResu... int[]) : void
# onCreate (savedInstanceSt... Bundle) : void

PHẦN III: CÀI ĐẶT VÀ TRIỂN KHAI GIẢI PHÁP

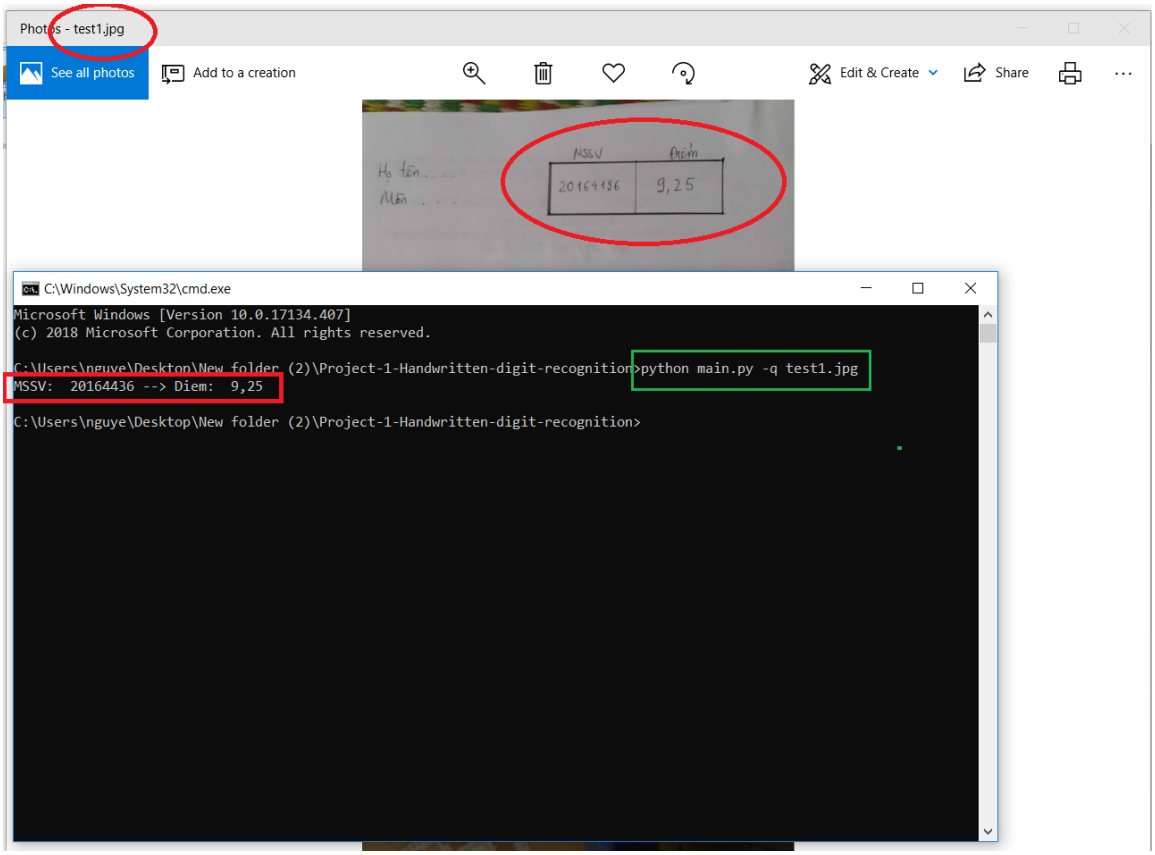
3.1. Bài toán nhận diện chữ viết tay

3.1.1. Tính năng 1

- **Cú pháp:** python main.py -q <tên ảnh đầu vào (nếu không cùng thư mục với project thì phải thêm đường dẫn)>
    - VD: Khi clone từ github về, trong project có sẵn các file test, giả sử ta muốn sử dụng file test1.jpg:

```
python main.py -q test1.jpg
```

- **Giao diện và kết quả ví dụ**

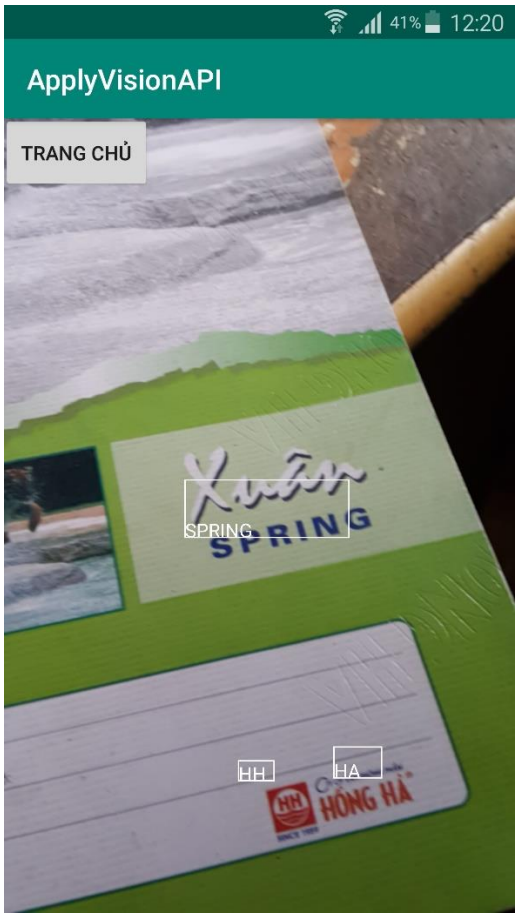


- **Các hàm quan trọng**
    - Hàm `find()` trong file ***find_and_crop.py*** trả về tọa độ 4 góc của hình chữ nhật lớn chứa cả điểm và MSSV trong ảnh đầu vào
    - Hàm `rotateAndCrop()` trong file ***find_and_crop.py*** nhận tham số đầu tiên là kết quả của hàm `find()` và trả về phần hình ảnh trong khung chữ nhật xác định bởi bộ tọa độ kết quả của hàm `find()`
    - `remove_Border()` trong file ***find_and_crop.py*** loại bỏ các cạnh của hình chữ nhật bao quanh các chữ số cần nhận diện

3.2. Bài toán sử dụng Vision API

3.2.1. Tính năng 1

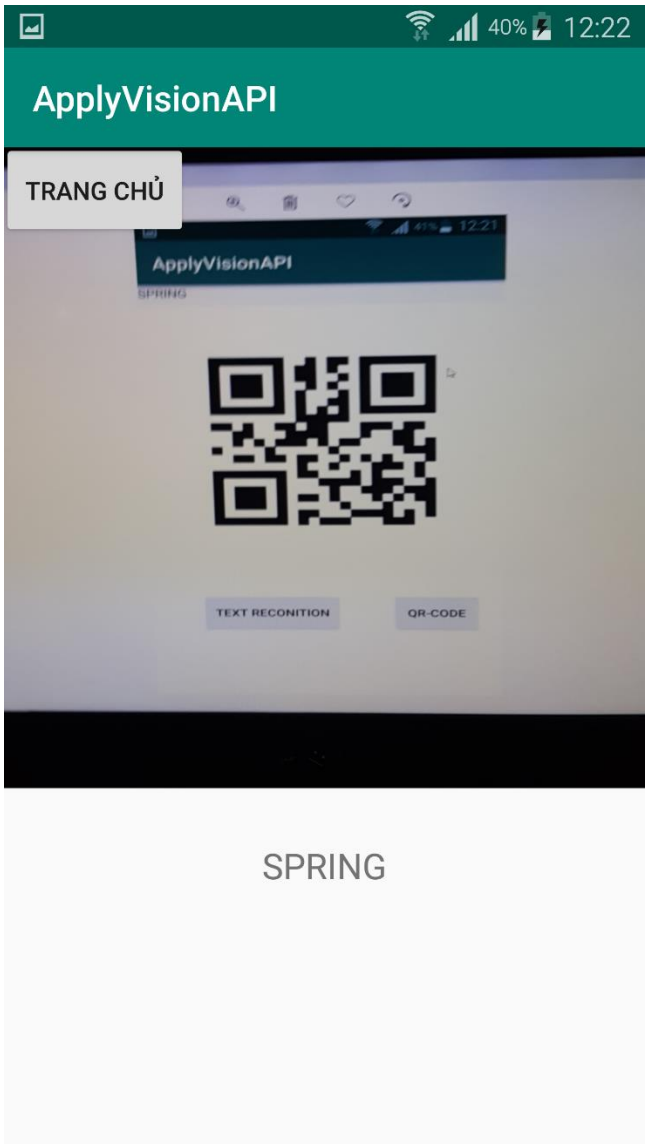
- **Text recognition: giao diện**



- **Các hàm quan trọng (Mã nguồn có trên github)**
  - `requestCameraPermission()` trong file ***TextRecognition.java*** : Có chức năng xin cấp quyền sử dụng camera của thiết bị
  - `createCameraSource()` trong file ***TextRecognition.java***: khởi tạo cameraSource và TextRecognizer
  - `onRequestPermissionsResult()` trong file ***TextRecognition.java*** : Kiểm tra xem quyền camera có được cấp hay không
  - `onTap()` trong file ***TextRecognition.java*** : Xử lý sự kiện nhấp vào màn hình, chức năng mã hóa text → qrcode ở đây
  - File **OcrDetectorProcessor.java** có tác dụng tạo một DetectorProcessor để xử lý những text block phát hiện được
  - Các file trong package ***ui.camera*** và file ***OcrGraphic.java*** dùng để xác định tọa độ cũng như vẽ khung chữ nhật xung quanh text block trên ảnh
  - File ***ShowQRcode.java*** có chức năng hiển thị hình ảnh QR code đã mã hóa từ chuỗi nhận diện được

3.2.2. Tính năng 2

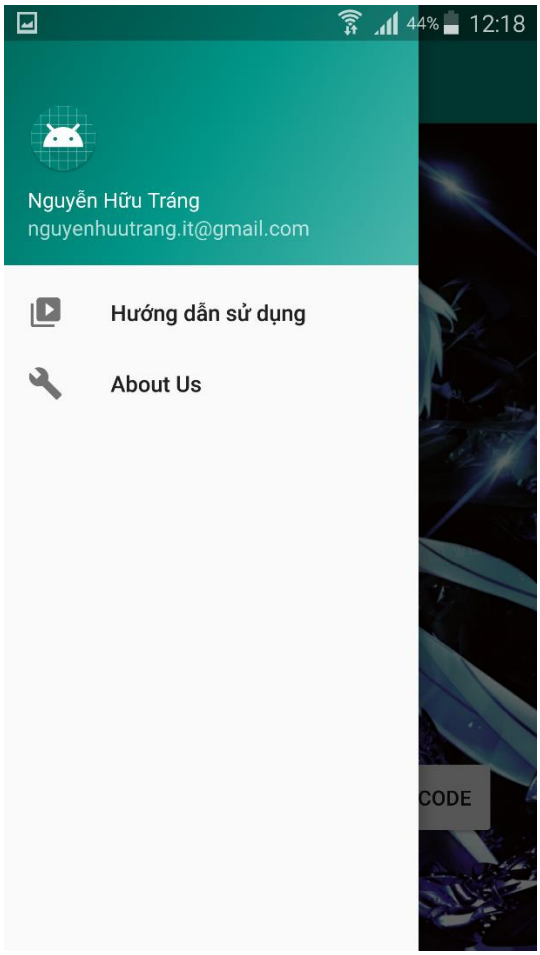
- **QR code(scan QR code): giao diện**



- **Các hàm quan trọng (Mã nguồn có trên github)**
  - `onRequestPermissionsResult()` trong file ***QRCode.java*** : để kiểm tra xem ứng dụng có được cấp quyền camera không
  - `onCreate()` trong file ***QRCode.java*** : hàm này được gọi khi activity active, bên trong hàm này có chứa khởi tạo BarcodeDetector, CameraSource, thiết lập BarcodeDetectorProcessor, yêu cầu quyền truy cập camera trong hàm `surfaceCreated()` , cũng như giải mã và trả kết quả ra màn hình thông qua hàm `receiveDetections()`

3.2.3. Tính năng 3

- **Hamburger menu: Giao diện**



- **Các hàm quan trọng (Mã nguồn có trên github)**
  - `onNavigationItemSelected()` trong file ***MainActivity.java*** : xử lý khi có một item trong hamburger menu được nháy. Đây chính là nơi xử lý cho các dialog.

PHẦN IV: KẾT LUẬN

- **Những công việc đã làm được**
    - Biết cách sử dụng tesseract, OpenCV cơ bản
    - Biết cách tích hợp các API vào ứng dụng android
    - Làm quen được với nhiều khái niệm, kiến thức trong bài toán text recognition
    - Có kiến thức cơ bản về cách xử lý ảnh
    - Biết cách sử dụng github để quản lý mã nguồn
- **Những công việc chưa làm được**
    - Chưa training bổ sung được cho tesseract để tang độ chính xác
    - Chưa giải quyết được bài toán đẩy text vào translate module
    - Chưa thể tự xây dựng một thuật toán học máy để nhận diện số viết tay.

TÀI LIỆU THAM KHẢO

**** Danh mục sách:***

**** Danh mục tạp chí:***

**** Danh mục hội thảo:***

**** Danh mục internet:***

1. Qui định về đồ án tốt nghiệp của Viện CNTT-TT https://soict.hust.edu.vn/index.php/2017/03/16/quy-dinh-ve-do-an-tot-nghiep/
2. Tesseract: https://github.com/tesseract-ocr/tesseract
3. pytesseract: https://pypi.org/project/pytesseract/
4. OpenCV: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
5. Mobile Vision API: https://developers.google.com/vision/
6. zxing generate qr code : https://www.android-examples.com/generate-qr-code-in-android-using-zxing-library-in-android-studio/