

**TRƯỜNG ĐẠI HỌC TRẦN ĐẠI NGHĨA  
KHOA CÔNG NGHỆ THÔNG TIN**

---



**ĐỒ ÁN MÔN HỌC**

**MÔN HỌC: NHẬP MÔN XỬ LÝ ẢNH**

**ĐỀ TÀI:**

**TÌM HIỂU THƯ VIỆN OPENCV, NGÔN NGỮ LẬP TRÌNH  
PYTHON VÀ VIẾT ỨNG DỤNG NHẬN DIỆN LÀN ĐƯỜNG  
CHO XE TỰ LÁI**

**TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2020**

**TRƯỜNG ĐẠI HỌC TRẦN ĐẠI NGHĨA  
KHOA CÔNG NGHỆ THÔNG TIN**

---



**ĐỒ ÁN MÔN HỌC**

**MÔN HỌC: NHẬP MÔN XỬ LÝ ẢNH**

**ĐỀ TÀI:**

**TÌM HIỂU THƯ VIỆN OPENCV, NGÔN NGỮ LẬP TRÌNH  
PYTHON VÀ VIẾT ỨNG DỤNG NHẬN DIỆN LÀN ĐƯỜNG  
CHO XE TỰ LÁI**

**Nhóm báo cáo:**

**Nguyễn Tiểu Phụng**

**Huỳnh Đức Anh Tuấn**

**Giảng viên hướng dẫn:**

**Th.s Ngô Thanh Tú**

**TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2020**

# MỤC LỤC

CHƯƠNG 1: TÌM HIỂU NGÔN NGỮ LẬP TRÌNH PYTHON .....	1
1.1. Giới thiệu ngôn ngữ lập trình Python .....	1
1.1.1. Lịch sử phát triển.....	1
1.1.2. Phiên bản .....	1
1.1.3. Một số điểm khác nhau giữa phiên bản 3x và 2x.....	2
1.1.4. Đặc điểm của Python .....	5
1.2. Hướng dẫn cài đặt bằng Anaconda/Miniconda.....	6
1.2.1. Giới thiệu Ananconda/Miniconda và cài đặt .....	6
1.2.2. Download Anaconda/Miniconda và hướng dẫn cài đặt.....	6
1.2.3. Hướng dẫn cài thêm thư viện bằng conda.....	13
CHƯƠNG 2: TÌM HIỂU XỬ LÝ ẢNH VÀ THƯ VIỆN OPENCV .....	16
2.1 . Tìm hiểu môn học xử lý ảnh .....	16
2.1.1. Lịch sử phát triển.....	16
2.1.2. Các lĩnh vực ứng dụng .....	18
2.1.3. Một số khái niệm cơ bản trong xử lý ảnh .....	22
2.1.4. Một số thư viện nổi tiếng .....	24
2.2 . Các phương pháp và kỹ thuật cụ thể.....	26
2.2.1. Trình bày đặc trưng Canny. ....	26
2.2.2. Ứng dụng phát hiện làn đường cho xe tự lái.....	27
2.3 . Tìm hiểu thư viện OpenCV.....	27
2.3.1. Lịch sử phát triển.....	27
2.3.2. Các chức năng và phiên bản sử dụng.....	28
2.3.3. Chức năng của thư viện OpenCV .....	29
2.3.4. Các nhóm hàm trong thư viện openCV.....	34
CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG.....	36
3.1 . Nêu bài toán .....	36
3.2 . Các bước thực hiện .....	36
3.2.1. Bước 1: Đọc và tiền xử lý hình ảnh. ....	36
3.2.2. Bước 2: Tạo xử lý hình ảnh.....	37
3.2.3. Bước 3: Xử lý nhận dạng làn đường .....	39

3.2.4.	Bước 4: Đưa hình ảnh lên màn hình .....	43
3.2.5.	Bước 4: Tạo bảng điều khiển tọa độ cho lớp phủ làn đường. ....	43
3.2.6.	Bước 6: Tạo giao diện đồ họa .....	44
3.3 .	Hướng dẫn sử dụng .....	44
3.4 .	Đánh giá về kết quả và đề xuất hướng phát triển .....	46
3.4.1.	Kết quả đạt được .....	46
3.4.2.	Hạn chế.....	47
3.4.3.	Hướng phát triển .....	47
TÀI LIỆU THAM KHẢO .....		48

## **LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành đến thầy cô giảng viên trong khoa Công nghệ thông tin trường Đại học Trần Đại Nghĩa. Và đặc biệt là thầy Thạc sĩ Ngô Thanh Tú – giảng viên học phần “Nhập môn xử lý ảnh” đã tận tình hướng dẫn, truyền đạt kiến thức và kỹ năng cần thiết để em có thể hoàn thành đồ án môn học này.

Tuy nhiên, trong quá trình tìm hiểu và nghiên cứu đề tài, do kiến thức chuyên ngành và thời gian còn hạn chế em vẫn còn nhiều thiếu sót trong quá trình tìm hiểu, thực hiện, đánh giá và trình bày về đề tài. Rất mong được sự quan tâm, góp ý của các thầy cô và giảng viên bộ môn để đồ án môn học của em được hoàn chỉnh hơn.

Xin chân thành cảm ơn!

# MỞ ĐẦU

## 1. Lý do chọn đề tài

- Ngày nay, công nghệ xử lý ảnh chiếm vị trí cực kỳ quan trọng trong guồng máy vận hành của nhiều lĩnh vực như kinh tế, tài chính, giải trí, du lịch, dịch vụ, giao thông vận tải. Nó giờ được coi là xu hướng công nghệ thế giới và nhiều người cho rằng đó là cuộc cách mạng công nghiệp lần thứ 4.
- Trong đó các nghiên cứu về khả năng tự lái cho một hệ thống trợ lý lái xe tiên tiến (ADAS) đã nhận được nhiều sự chú ý lớn. Một trong những mục tiêu chính của lĩnh vực nghiên cứu này là cung cấp chức năng thông minh và an toàn hơn cho người lái xe bằng cách sử dụng công nghệ thông tin và điện tử. Trong điều kiện đường xá đặc thù thì khả năng nhận biết và phát hiện các biển báo đường, làn đường và đèn giao thông là rất quan trọng và đóng vai trò quan trọng đối với các hệ thống ADAS.
- Nhận thấy tầm quan trọng và tính ứng dụng cao của việc nhận diện làn đường cho xe tự lái. Trong đồ án này em sẽ tìm hiểu về thư viện đồ họa OpenCV và đặc trưng nhận diện biên ảnh để xây dựng ứng dụng nhận diện làn đường cho xe tự lái.

## 2. Mục đích nghiên cứu

- Nghiên cứu tìm hiểu thư viện đồ họa OpenCV-python đặc trưng nhận diện biên ảnh. Khi đưa vào hình ảnh từ camera hoặc video thì có thể sử dụng đặc trưng nhận diện biên ảnh xác định làn đường dành cho xe tự lái.

## 3. Cấu trúc đồ án

- Chương 1: Tìm hiểu ngôn ngữ lập trình Python
- Chương 2: Tìm hiểu xử lý ảnh và thư viện OpenCV
- Chương 3: Xây dựng ứng dụng

# CHƯƠNG 1: TÌM HIỂU NGÔN NGỮ LẬP TRÌNH PYTHON

## 1.1. Giới thiệu ngôn ngữ lập trình Python

### 1.1.1. Lịch sử phát triển

- Ngôn ngữ Python được Guido van Rossum tạo ra và được phát hành lần đầu tiên vào tháng 2 năm 1991.
- Python khá giống Perl, Ruby, Scheme, Smalltalk và Tcl.
- Python được phát triển trong một dự án mã mở do một tổ chức phi lợi nhuận Python Software Foundation quản lý.
- Python được phát triển để chạy trên nền Unix. Nhưng theo thời gian, nó đã "bành trướng" sang mọi hệ điều hành từ MS-DOS đến MAC OS, OS/2, Windows, Linux và một số điều hành khác thuộc họ Unix.
- Python là ngôn ngữ bậc cao (high-level), có hình thức sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cho phép người sử dụng viết mã với số lần gõ phím tối thiểu.
- Python cũng là một trong những ngôn ngữ phổ biến nhất thế giới.
- Python không phải được đặt theo tên của con rắn thần Python trong thần thoại Hy Lạp đâu. Rossum là fan của một sê-ri chương trình hài cuối những năm 1970, và cái tên "Python" được lấy từ tên một phần trong sê-ri đó "Monty Python's Flying Circus".

### 1.1.2. Phiên bản

Bảng 1.1 các phiên bản Python đã phát hành	
Phiên bản	Ngày phát hành
Python 1.0 (bản phát hành chuẩn đầu tiên)	01/1994
Python 1.6 (Phiên bản 1.x cuối cùng)	05/09/2000
Python 2.0 (Giới thiệu list comprehension)	16/10/2000
Python 2.7 (Phiên bản 2.x cuối cùng)	03/07/2010
Python 3.0 (Loại bỏ cấu trúc và mô-đun trùng lặp)	03/12/2008

Bảng 1.1 các phiên bản Python đã phát hành	
Phiên bản	Ngày phát hành
Python 3.3	2012
Python 3.5 (hỗ trợ cho các byte và bytearray)	13/09/2015
Python 3.6(cải tiến đáng kể trong thư viện tiêu chuẩn)	23/12/2016
Python 3.7(Cải tiến mô hình dữ liệu Python)	27/06/2018
Python 3.8 (Được bổ sung nhiều tính năng mới)	14/10/2019
Python 3.9 (Loại bỏ hết các tính năng dùng tương thích ngược ở bản 2x)	27/04/2020

### 1.1.3. Một số điểm khác nhau giữa phiên bản 3x và 2x

#### 1.1.3.1. Sự khác biệt ở hàm PRINT

- Ở 2x print 'a','b' dễ gây hiểu lầm với kiểu dữ liệu Tuple khi ta truyền nhiều đối số vào parantheses.
- Ở phiên bản 3x hàm print dùng ().

<p>Python 2x</p> <pre>print 'Hello, World!' print ('Hello, World!')</pre> <p>&gt;&gt;&gt; Hello, Word!</p> <p>&gt;&gt;&gt; Hello, Word!</p>	<p>Python 3x</p> <pre>print ('Hello, World!') print 'Hello, World!'  &gt;&gt;&gt; Hello, World! File "&lt;ipython-input-3-139a7c5835bd&gt;", line 2     print 'Hello, World!'           ^ SyntaxError: invalid syntax</pre>
---	---



### 1.1.3.2. Toán tử DIV

- Ở Python 2x toán tử chia (/) có một ý nghĩa mơ hồ cho các đối số 'số': nó trả về sàn của kết quả toán học của phép chia nếu các đối số là kiểu ints hoặc long, nhưng nó trả về một xấp xỉ hợp lý của kết quả phân chia nếu các đối số là kiểu float hoặc phức. Vấn đề này lại được giải quyết ở bản 3x.

#### Python 2x

```
print '3 / 2 =', 3 / 2
print '3 // 2 =', 3 // 2
print '3 / 2.0 =', 3 / 2.0
print '3 // 2.0 =', 3 // 2.0
```

```
>>> 3 / 2 = 1
>>> 3 // 2 = 1
>>> 3 / 2.0 = 1.5
>>> 3 // 2.0 = 1.0
```

#### Python 3x

```
print ('3 / 2 =', 3 / 2)
print ('3 // 2 =', 3 // 2)
print ('3 / 2.0 =', 3 / 2.0)
print ('3 // 2.0 =', 3 // 2.0)
```

```
>>> 3 / 2 = 1.5
>>> 3 // 2 = 1
>>> 3 / 2.0 = 1.5
>>> 3 // 2.0 = 1.0
```

### 1.1.3.3. Kiểu STRING mặc định

- Python 2x có các kiểu str () thuộc kiểu ASCII, riêng biệt unicode (), nhưng không có kiểu byte.
- Python 3x có các chuỗi Unicode (utf-8) và 2 lớp byte: byte và bytearray.

#### Python 2x

```
print type(unicode('Chuỗi này giống kiểu  
str ở Python 3'))
```

```
>>> <type 'unicode'>
```

```
print type(b'Đây giống như một chuỗi str  
do không có kiểu byte ở 2x')
```

```
>>> <type 'str'>
```

#### Python 3x

```
print('strings are now utf-8  
\u03BCnico\u0394é!')
```

```
>>> strings are now utf-8 \u03BCnico\u0394é!
```

```
print(sys.version, ' has', type(b' bytes for  
storing data'))
```

```
print '2 chuỗi này' + b'giống nhau'
>>> 2 chuỗi này giống nhau
```

```
>>> 3.8.1 (tags/v3.8.1:1b293b6, Dec 18
2019, 22:39:24) [MSC v.1916 32 bit
(Intel)] has <class 'bytes'>
print('note that we cannot add a string' +
b'bytes for data')
>>> Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not
"bytes") to str
```

#### 1.1.3.4. Loại bỏ hàm XRANGE() ở bản 3x

- Việc sử dụng xrange() rất phổ biến trong Python 2.x để tạo một đối tượng có thể lặp lại.
- Ở python 3x đã loại bỏ hàm XRANGE() thay vào đó là sử dụng hàm RANGE(). Nhưng hàm range() ở bản 3x đã được tích hợp cơ chế 'lazy avaluation' cơ chế giải phóng bộ nhớ đã chiếm dụng của xrange() (bản 2x).

#### 1.1.3.5. Xử lý ngoại lệ

- Trong Python 3x yêu cầu xử dụng từ khóa as để xử lý ngoại lệ.

Python 2x	Python 3x
try:	try:
let_us_cause_a_NameError	let_us_cause_a_NameError
except NameError, err:	except NameError as err:
print err, '--> our error message'	print(err, '--> our error message')
>>>name	>>>name 'let_us_cause_a_NameError'
'let_us_cause_a_NameError' is not defined --> our error message	is not defined --> our error message

#### 1.1.3.6. Banker's Round

- Python 3 đã áp dụng cách làm tròn số thập phân chuẩn mới hiện nay khi kết quả là hòa (0,5) ở các chữ số có nghĩa cuối cùng. Bây giờ, trong Python 3, số thập phân được làm tròn đến số chẵn gần nhất.
- Mặc dù đó là một sự bất tiện cho tính di động của mã, nhưng nó được cho là cách làm tròn tốt hơn so với làm tròn cũ vì nó tránh được sự thiên vị đối với số lượng lớn.

Python 2x	Python 3x
<code>round(15.5)</code>	<code>round(15.5)</code>
<code>&gt;&gt;&gt; 16.0</code>	<code>&gt;&gt;&gt; 16</code>
<code>round(16.5)</code>	<code>round(16.5)</code>
<code>&gt;&gt;&gt; 17.0</code>	<code>&gt;&gt;&gt; 16</code>

#### 1.1.3.7. Ordering Comparisons

- Python 3.0 đã đơn giản hóa các quy tắc của toán tử so sánh:
- Các toán tử so sánh thứ tự (`<`, `<=`, `>=`, `>`) đưa ra một ngoại lệ `TypeError` khi các toán hạng không có thứ tự tự nhiên có ý nghĩa. Do đó, các biểu thức như `1 < "`, `0 > Không` có hoặc len `<= len` không còn hợp lệ và ví dụ: `Không < Không` làm tăng `TypeError` thay vì trả về `Sai`.

#### 1.1.4. Đặc điểm của Python

- Python là ngôn ngữ lập trình hướng đối tượng, bậc cao, mạnh mẽ. Ngoài ra, học Python là khá đơn giản và dễ dàng. Python cũng là một ngôn ngữ thông dịch, tức là ngôn ngữ không cần phải biên dịch ra file chạy mà đọc code đến đâu thì chạy đến đó. Khi chạy lệnh Python ta sẽ có một giao diện dòng lệnh giống của Unix, có thể chạy từng dòng code ngay trực tiếp tại đây.
- Python có rất nhiều ưu điểm để khiến cho các nhà lập trình web yêu thích và sử dụng nó cho ngôn ngữ lập trình đầu tiên của mình đó là:
  - + Đơn giản: Cú pháp đơn giản giúp cho người lập trình dễ dàng đọc và tìm hiểu.
  - + Tốc độ: Python có tốc độ xử lý nhanh hơn so với ngôn ngữ PHP.

- + Tương tác: Chế độ tương tác cho phép người lập trình thử nghiệm tương tác sửa lỗi của các đoạn mã.
- + Chất lượng: Thư viện có tiêu chuẩn cao, Python có khối cơ sở dữ liệu khá lớn nhằm cung cấp giao diện cho tất cả các CSDL thương mại lớn.
- + Thuận tiện: Python được biên dịch và chạy trên tất cả các nền tảng lớn hiện nay.
- + Mở rộng: Với tính năng này, Python cho phép người lập trình có thể thêm hoặc tùy chỉnh các công cụ nhằm tối đa hiệu quả có thể đạt được trong công việc.
- + Có trên tất cả các nền tảng hệ điều hành từ UNIX, MS – DOS, Mac OS, Windows và Linux và các OS khác thuộc họ Unix.
- + Tương thích mạnh mẽ với Unix, hardware, third-party software với số lượng thư viện khổng lồ (400 triệu người sử dụng)
- + Python với tốc độ xử lý cực nhanh, python có thể tạo ra những chương trình từ những script siêu nhỏ tới những phần mềm cực lớn như Blender 3D.

## **1.2. Hướng dẫn cài đặt bằng Anaconda/Miniconda**

### **1.2.1. Giới thiệu Anaconda/Miniconda và cài đặt**

- Anaconda là một Distribution miễn phí và mã nguồn mở của Python và R giúp đơn giản hóa việc cài đặt, quản lý và triển khai packages (numpy, scipy, tensorflow, ...).
- Anaconda phục vụ cho nhiều mục đích, đặc biệt trong Data Science (Khoa học dữ liệu), Machine learning (Máy học), Big Data (Dữ liệu lớn), Image Processing (Xử lý ảnh), ...
- Anaconda hiện nay đã có hơn 20 triệu người dùng và hơn 7500 packages khoa học dữ liệu dành cho Windows, Linux và MacOS.
- Trong khi đó Spyder là 1 trong những IDE (môi trường tích hợp dùng để phát triển phần mềm) tốt nhất cho data science và quan trọng hơn là nó được cài đặt khi bạn cài đặt Anaconda.

### **1.2.2. Download Anaconda/Miniconda và hướng dẫn cài đặt**

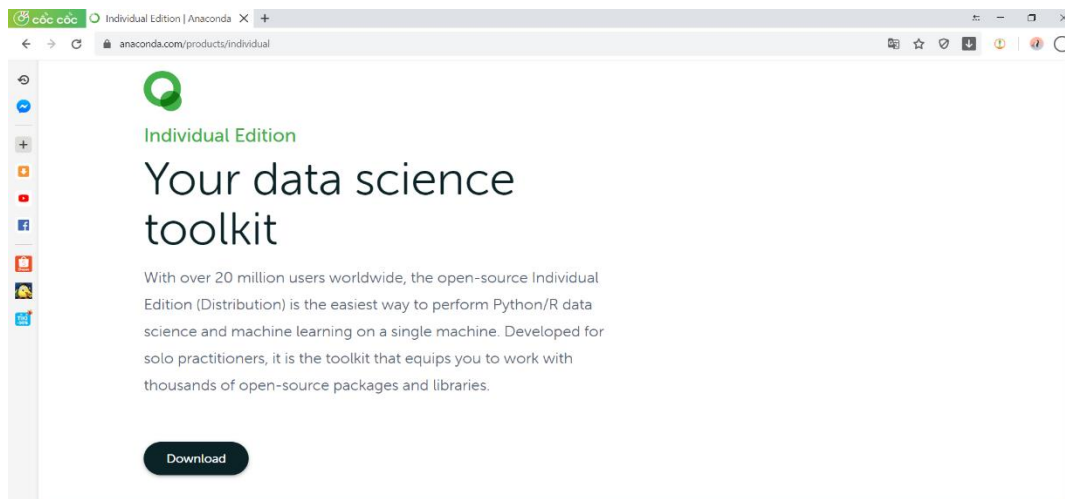
#### **1.2.2.1. Các bước cài đặt**

- Yêu cầu phần cứng và phần mềm:
  - + Hệ điều hành: Win 7, Win 8/8.1, Win 10, Red Hat Enterprise Linux/CentOS 6.7, 7.3, 7.4, and 7.5, and Ubuntu 12.04+.

- + Ram tối thiểu 4GB.
- + Ổ cứng trống tối thiểu 3GB để tải và cài đặt.

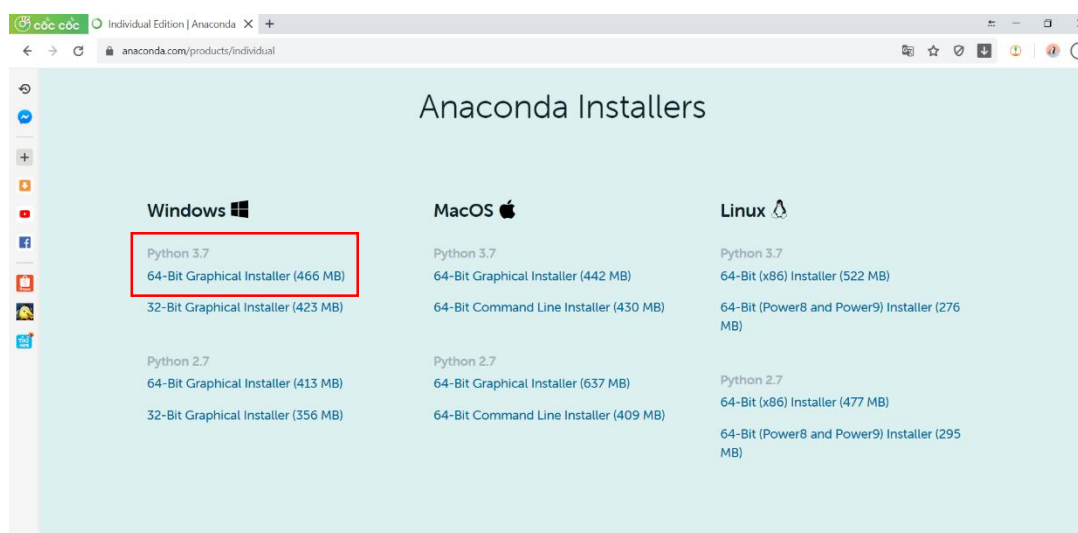
### Cài đặt:

- Bước 1: truy cập vào trang web <https://www.anaconda.com/>



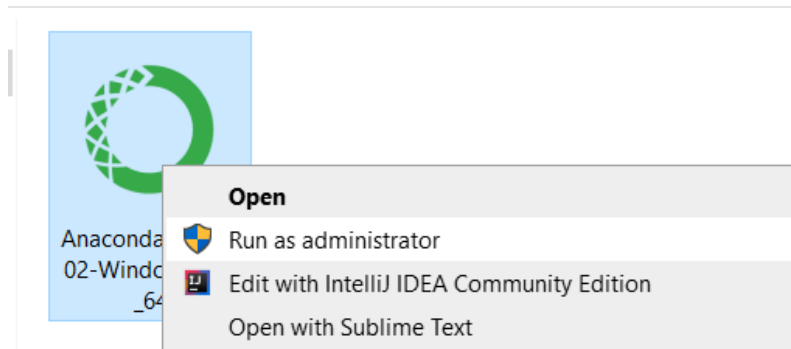
Hình 1.1 Trang chủ ananconda.com

- + Chọn xuống mục download: và chọn tải phiên bản thích hợp, ở đây em chọn hệ điều hành windows bản python 3.7 và 64-bit Graphical.



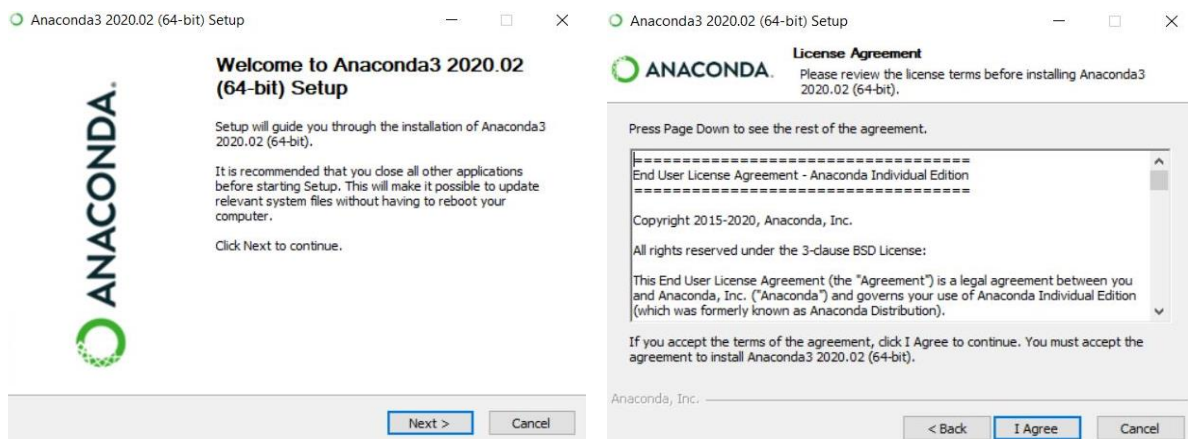
Hình 1.2 Download bản cài đặt

- Bước 2: Chạy file cài đặt với quyền admin



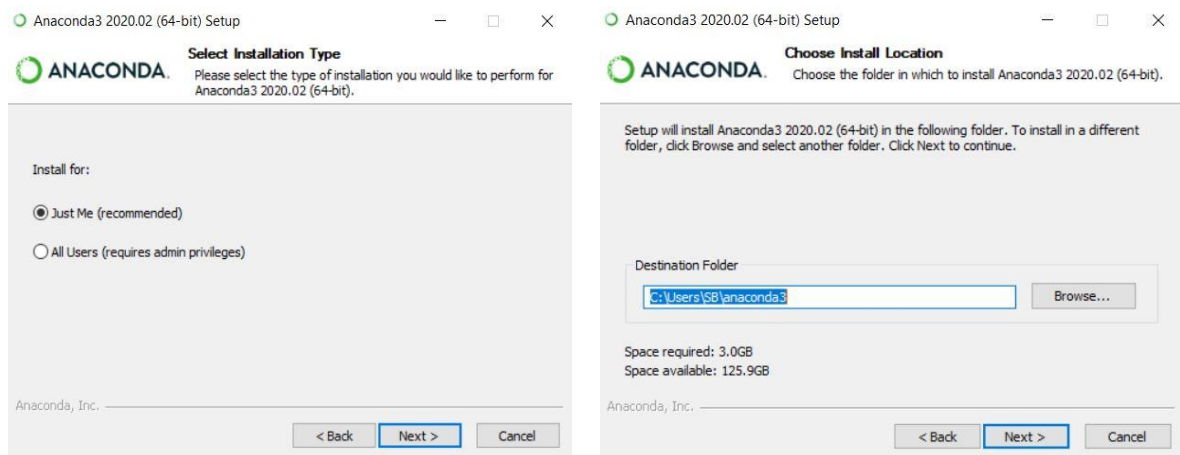
Hình 1.3 tiến hành cài đặt

- Bước 3: Chấp nhận các yêu cầu thiết lập và tiến hành cài đặt



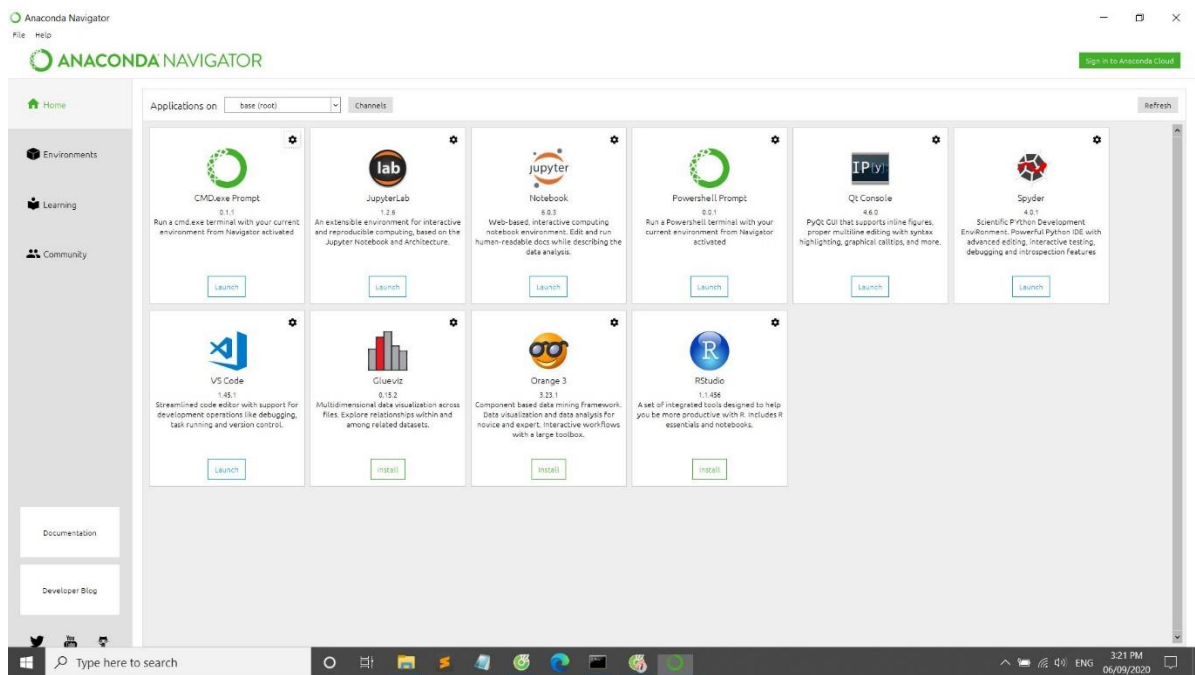
Hình 1.4 Giao diện cài đặt

- Bước 4: Chọn tài khoản và vị trí để cài đặt. Ở đây em chọn recommended và ổ C:/



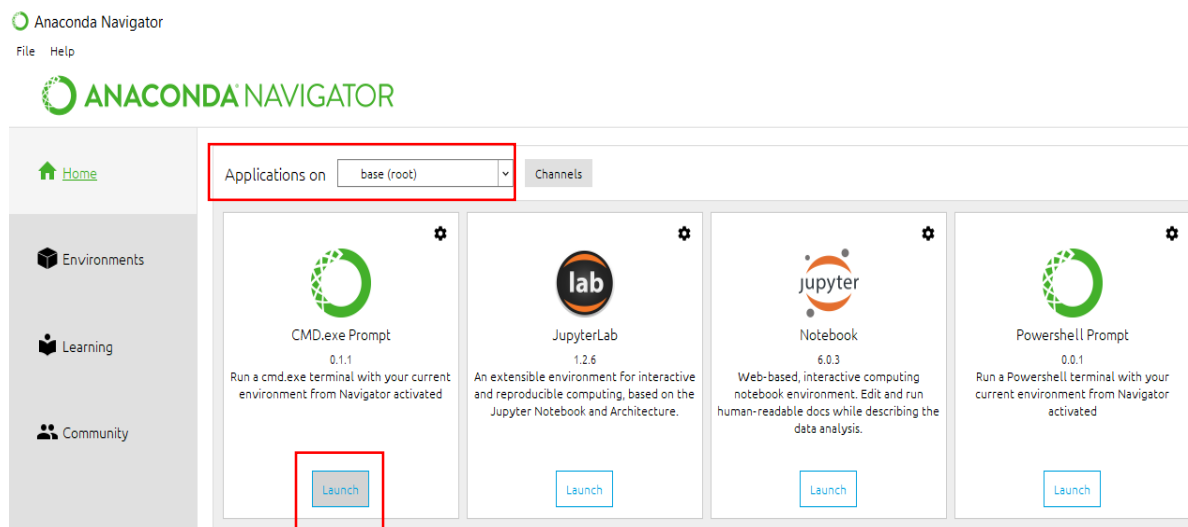
Hình 1.5 Chọn tài khoản và vị trí để cài đặt

- Bước 5: Giao diện Anaconda khi cài đặt xong



Hình 1.6 Giao diện của phần mềm Anaconda

– Bước 6: Kiểm tra lại phiên bản phần mềm Anaconda



Hình 1.7 Giao diện home của Anaconda

- + Chọn Launch ở Environments base(root) để mở CMD ở môi trường conda để kiểm tra lại phiên bản cài đặt bằng lệnh: **conda -V**

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

(base) C:\Users\SB>conda -V
conda 4.8.2

(base) C:\Users\SB>
```

Hình 1.8 kiểm tra phiên bản conda

```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

(base) C:\Users\SB>python
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

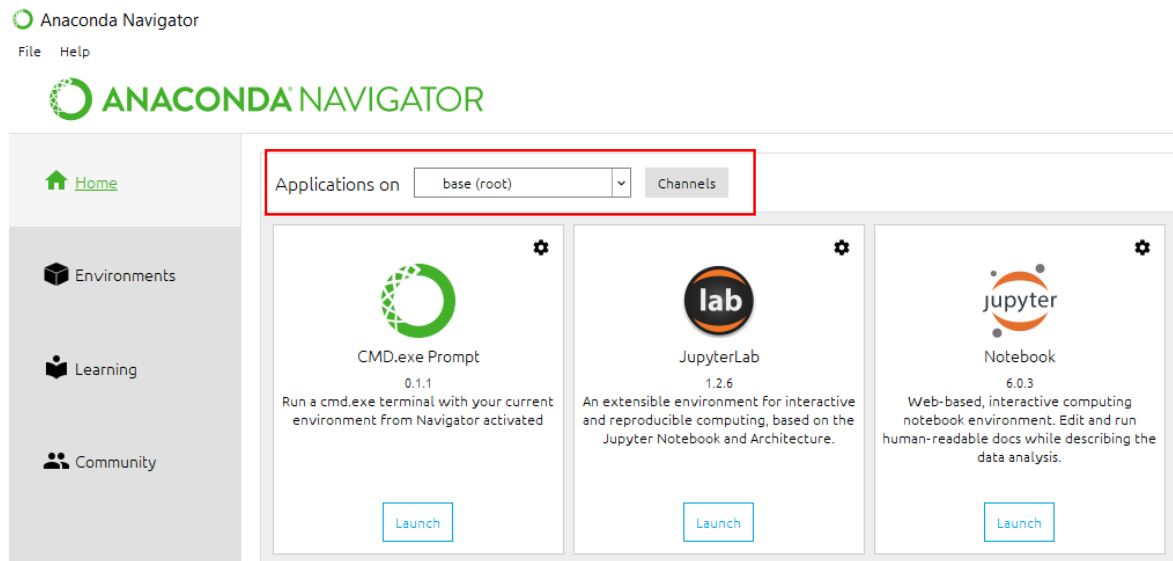
Hình 1.9 kiểm tra phiên bản python

#### 1.2.2.2. Quản lý môi trường

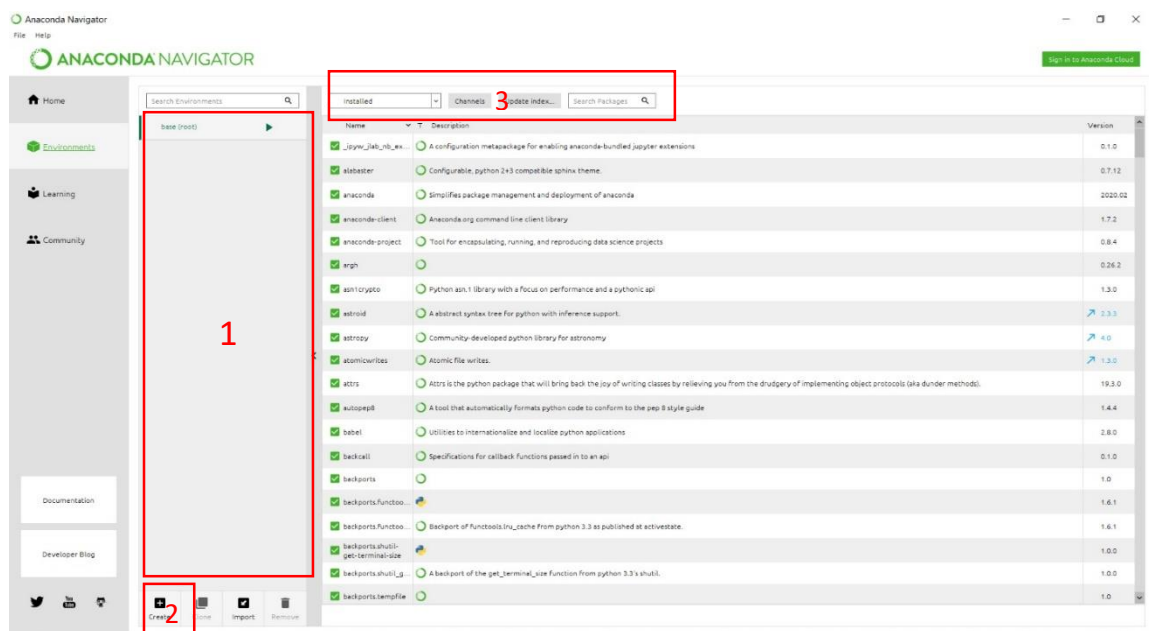
- Với Anaconda có nhiều packages khoa học phụ thuộc vào các phiên bản cụ thể của các packages khác. Các nhà khoa học dữ liệu thường sử dụng nhiều phiên bản của nhiều package và sử dụng nhiều môi trường để phân tách các phiên bản khác nhau này.
- Chương trình dòng lệnh (command-line program conda) vừa là trình quản lý các package vừa là trình quản lý môi trường (environment manager). Điều này giúp các nhà khoa học dữ liệu đảm bảo rằng mỗi phiên bản của mỗi package có tất cả các phụ thuộc mà nó yêu cầu và hoạt động chính xác.
- Anaconda Navigator cung cấp cho người dùng một giao diện đồ họa để quản lý các environment (môi trường) và package. Ta sẽ có environment mặc định là base (root) chứa các package cơ bản.



- Ở ngăn giao diện Home là nơi quản lý các Application (ứng dụng) tại một environment (trong vòng đỏ).



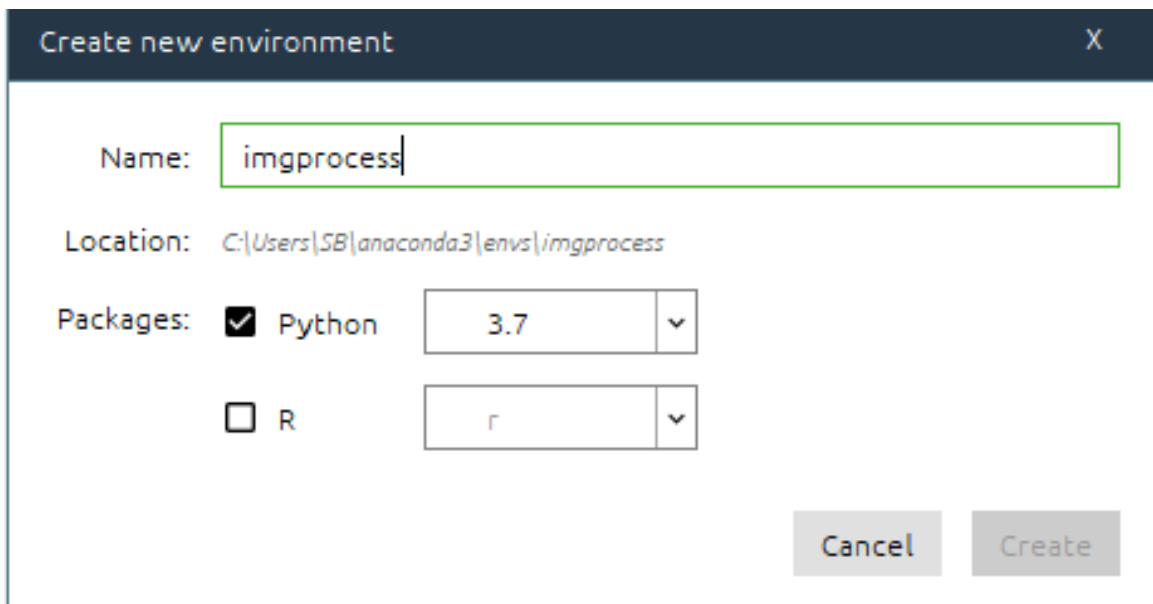
Hình 1.10 Giao diện environment



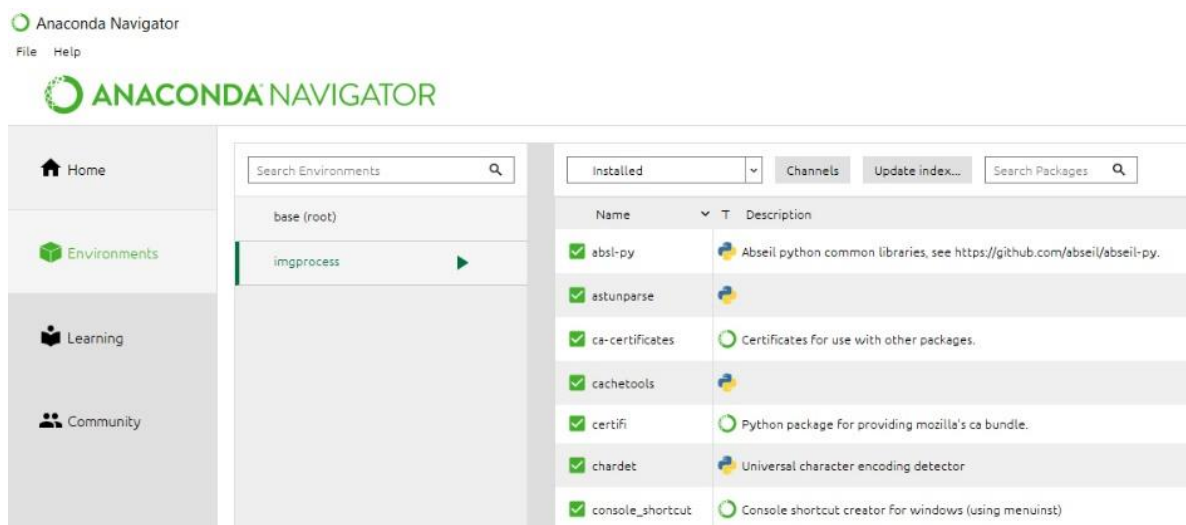
Hình 1.11 các ngăn của giao diện Environments

- Vùng số 1 là danh sách các environment ta đã tạo.
- Vùng số 2 là nút để tạo environment mới, sau nhấp chuột vào ta sẽ có giao diện như hình 1.12. Chúng ta chọn version của Python và đặt tên cho môi trường.

- Tương tự Clone là để sao chép một bản environment với các package giống một environment đã tạo. Import dùng để tạo environment bằng file có sẵn. Remove để xóa environment.
- Vùng thứ 3 dùng để tìm kiếm và cài đặt các package trong environment bạn đã chọn trong vùng thứ nhất.
- Ví dụ tạo mới một environment imgprocess.



Hình 1.12 tạo mới một environment imgprocess

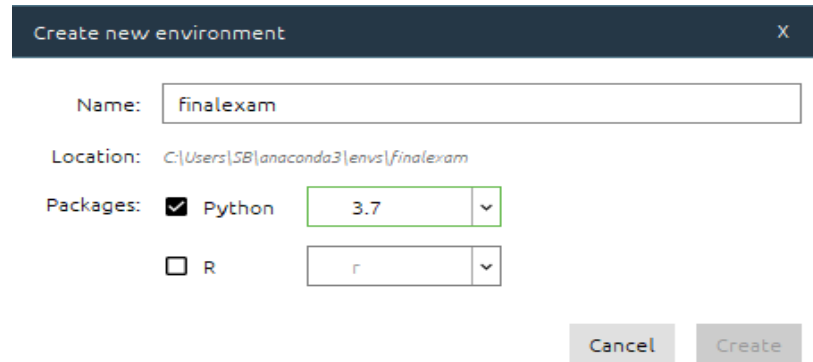


Hình 1.13 environment imgprocess đã được tạo thành công

### 1.2.3. Hướng dẫn cài thêm thư viện bằng conda

#### 1.2.3.1. Các thư viện sử dụng

- Tạo một environment finalexam để cài các thư viện hỗ trợ cho CHƯƠNG 3.



Hình 1.14 create environment finalexam

- Các thư viện cần cài đặt thêm như:

- + OpenCV
- + Numpy
- + PILLOW

#### 1.2.3.2. Cài đặt bằng dòng lệnh

- Mở CMD.exe Prompt
- Thư viện openCV: dùng lệnh ***conda install -c conda-forge opencv***

```
C:\Windows\system32\cmd.exe - conda install -c conda-forge opencv
Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

(finalexam) C:\Users\SB>conda install -c conda-forge opencv
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.8.2
  latest version: 4.8.3

  ca-certificates pkgs/main::ca-certificates-2020.1.1-0 --> conda-forge::ca-certificates-2020.4.5.2-hecda079_0
  certifi         pkgs/main::certifi-2020.4.5.1-py37_0 --> conda-forge::certifi-2020.4.5.2-py37hc8dfbb8_0

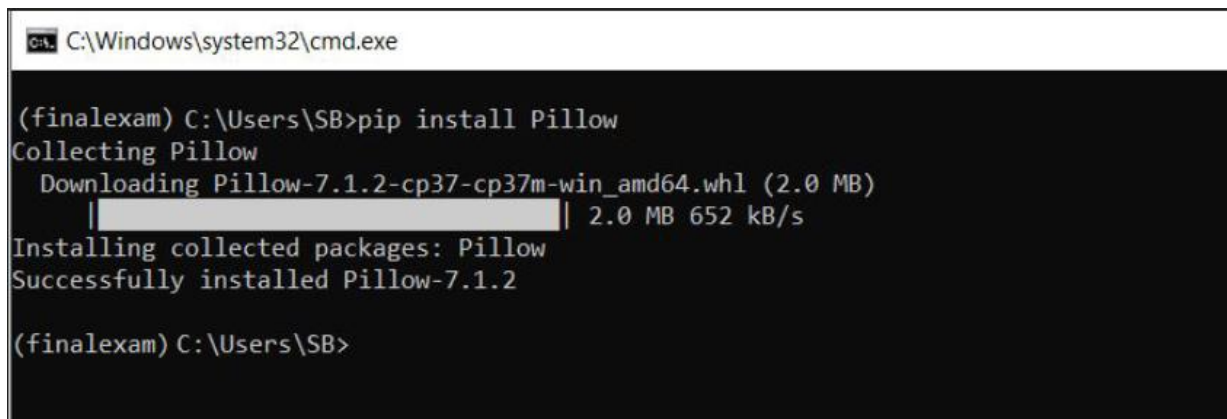
The following packages will be SUPERSEDED by a higher-priority channel:

  openssl                                pkgs/main --> conda-forge

Proceed ([y]/n)?
```

Hình 1.15 cài thư viện opencv trên env fianlexam bằng cmd

- Thư viện PILLOW: dùng lệnh *pip install Pillow*

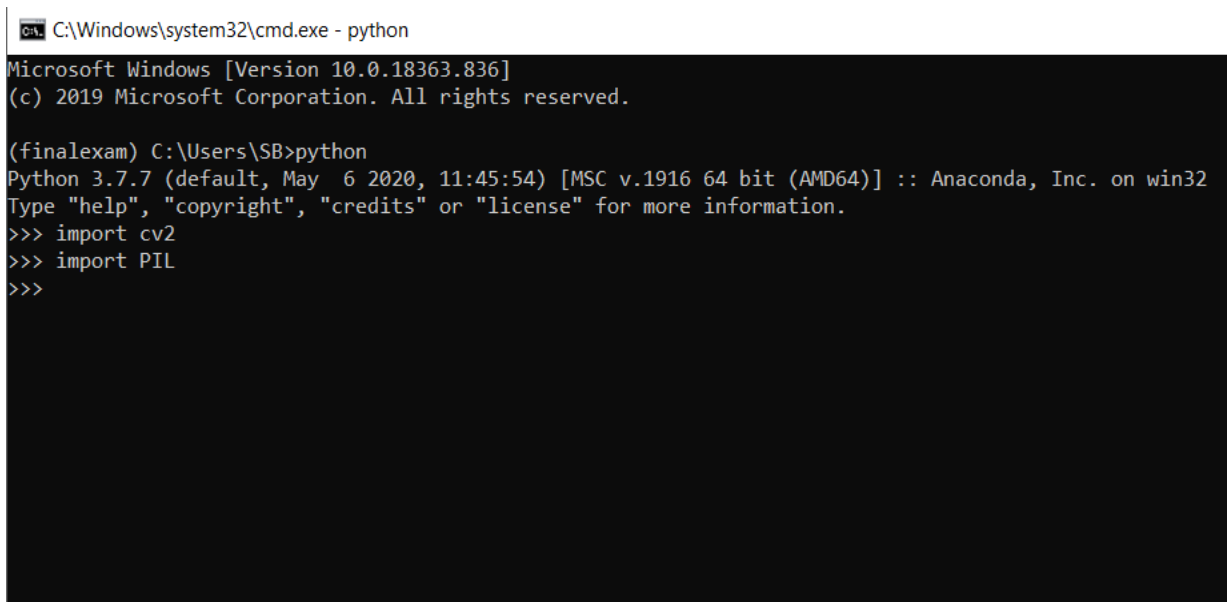


```
C:\Windows\system32\cmd.exe

(finalexam) C:\Users\SB>pip install Pillow
Collecting Pillow
  Downloading Pillow-7.1.2-cp37-cp37m-win_amd64.whl (2.0 MB)
    | 2.0 MB 652 kB/s
Installing collected packages: Pillow
Successfully installed Pillow-7.1.2

(finalexam) C:\Users\SB>
```

Hình 1.16 cài đặt thư viện Pillow bằng cmd



```
C:\Windows\system32\cmd.exe - python

Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

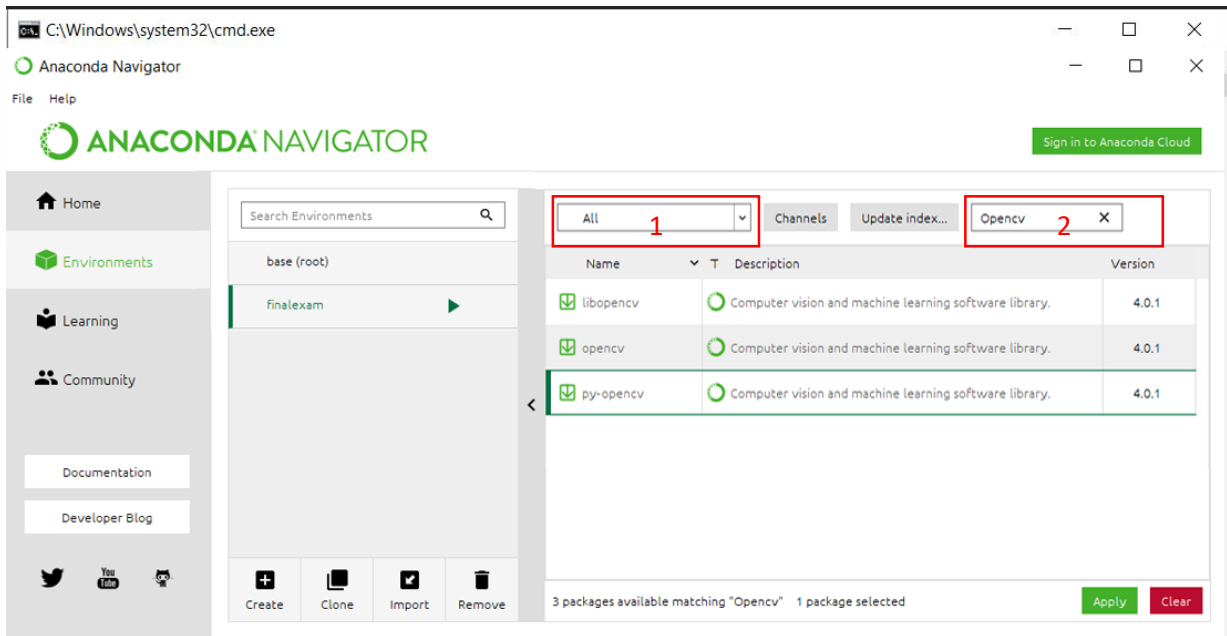
(finalexam) C:\Users\SB>python
Python 3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> import PIL
>>>
```

Hình 1.17 kiểm tra lại các thư viện đã cài đặt

### 1.2.3.3. Cài đặt bằng anaconda-navigator

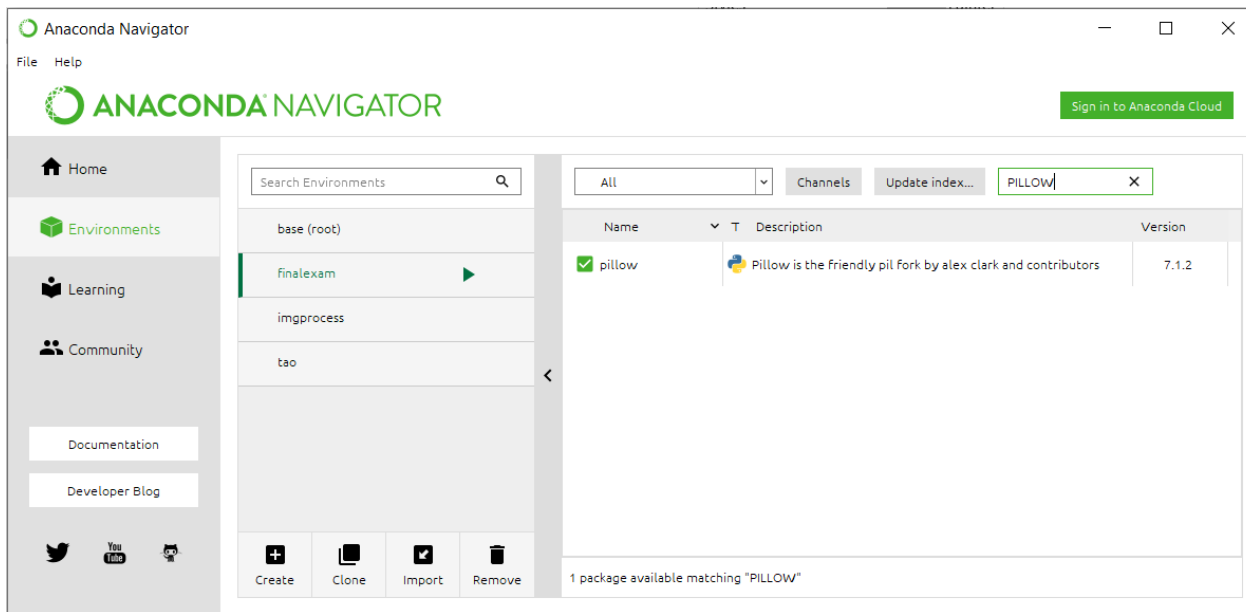
- Để cài thư viện với giao diện anaconda-navigator đầu tiên ta chọn môi trường cần cài thư viện (finalexam).
  - + Ở vùng số 1 chọn All, vùng số 2 gõ tên thư viện cần cài đặt vào.
  - + Chọn tick các package cần cài đặt và chọn Apply.

## Cài thư viện OpenCV bằng giao diện anaconda-navigator



Hình 1.18 cài đặt opencv bằng ananconda-navigator

## Cài thư viện Pillow bằng giao diện anaconda-navigator



Hình 1.19 cài đặt PIL bằng anaconda-navigator

## CHƯƠNG 2: TÌM HIỂU XỬ LÝ ẢNH VÀ THƯ VIỆN OPENCV

### 2.1 . Tìm hiểu môn học xử lý ảnh

#### 2.1.1. Lịch sử phát triển

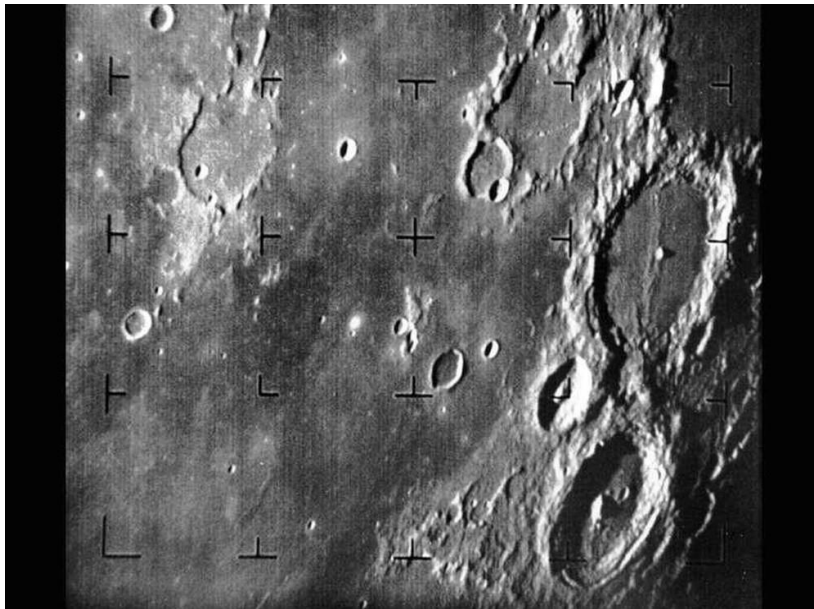
- Xử lý hình ảnh là một phương pháp để thực hiện một số thao tác trên một hình ảnh, để có được một hình ảnh nâng cao hoặc để trích xuất một số thông tin hữu ích từ nó. Đây là một loại xử lý tín hiệu trong đó đầu vào là hình ảnh và đầu ra có thể là hình ảnh hoặc đặc điểm / tính năng liên quan đến hình ảnh đó. Ngày nay, xử lý hình ảnh là một trong những công nghệ đang phát triển nhanh chóng. Nó tạo thành lĩnh vực nghiên cứu cốt lõi trong các ngành khoa học kỹ thuật và máy tính.
- Xử lý ảnh về cơ bản bao gồm ba bước sau:
  - + Nhập hình ảnh thông qua các công cụ thu nhận hình ảnh;
  - + Phân tích và xử lý hình ảnh;
  - + Đầu ra trong đó kết quả có thể được thay đổi hình ảnh hoặc báo cáo dựa trên phân tích hình ảnh.
- Lịch sử về xử lý ảnh:



Hình 2.1 Một bức tranh kỹ thuật số được sản xuất vào năm 1921 từ một cuộn băng được mã hóa bởi một máy in điện báo có khuôn mặt kiểu đặc biệt. (McFarlane.)

- + Bắt nguồn từ hai ứng dụng: nâng cao chất lượng thông tin hình ảnh và xử lý số liệu cho máy tính.
- + Ứng dụng đầu tiên là việc truyền thông tin ảnh báo giữa London và New York vào năm 1920 qua cable Bartlane.

- Mã hóa dữ liệu ảnh → khôi phục ảnh.
  - Thời gian truyền ảnh: Từ 1 tuần → 3 giờ.
- + Ảnh số được tạo ra vào năm 1921 từ băng mã hóa của một máy in điện tín.
  - + Ảnh số được tạo ra vào năm 1922 từ card đục lỗ sau 2 lần truyền qua Đại Tây Dương.
  - + Ảnh 15 cấp độ xám được truyền từ London đến New York năm 1929.(McFarlane)
  - + Hệ thống đầu tiên có khả năng mã hóa hình ảnh với mức xám là 5 và tăng lên 15 vào năm 1929.
  - + Trong khoản thời gian này, người ta chỉ nói đến ảnh số, chứ chưa đề cập gì đến xử lý ảnh số, vì một lý do đơn giản là máy tính chưa có.
  - + Năm 1964, hình ảnh của mặt trăng được đưa về Trái Đất thông qua máy chụp của tàu Ranger 7 của Jet Propulsion Laboratory (Pasadena, California) để cho máy tính xử lý (chỉnh méo).



Hình 2.2 Ảnh đầu tiên của mặt trăng được chụp bởi tàu vũ trụ Mỹ Ranger 7, vào 9 giờ 09 phút sáng ngày 31/7/1964 (nguồn: NASA)

- + Song song với các ứng dụng trong khám phá không gian, các kỹ thuật xử lý ảnh cũng đã được bắt đầu vào cuối những năm 1960 và đầu những năm 1970 trong y học, theo dõi tài nguyên Trái đất và thiên văn học.



- + Đến nay xử lý ảnh đã có một bước tiến dài trong nhiều ngành khoa học, từ những ứng dụng đơn giản đến phức tạp.

## 2.1.2. Các lĩnh vực ứng dụng

### 2.1.2.1. Phương pháp xử lý ảnh

- Có hai loại phương pháp được sử dụng để xử lý hình ảnh là xử lý hình ảnh tương tự và kỹ thuật số. Xử lý hình ảnh tương tự có thể được sử dụng cho các bản sao cứng như bản in và hình ảnh.
- Các nhà phân tích hình ảnh sử dụng các nguyên tắc cơ bản khác nhau khi sử dụng các kỹ thuật hình ảnh này. Kỹ thuật xử lý hình ảnh kỹ thuật số giúp xử lý các hình ảnh kỹ thuật số bằng cách sử dụng máy tính.

### 2.1.2.2. Một số lĩnh vực được ứng dụng:

- Xử lý ảnh vệ tinh, ảnh viễn thám bao gồm:
  - + Ứng dụng của ảnh vệ tinh và dữ liệu viễn thám được đưa vào thực tế hỗ trợ các hoạt động quản lý và giám sát trong mọi lĩnh vực như: An ninh – quốc phòng, tài nguyên môi trường, nông – lâm nghiệp, quản lý rừng...



Hình 2.3 Ảnh phân loại lớp phủ



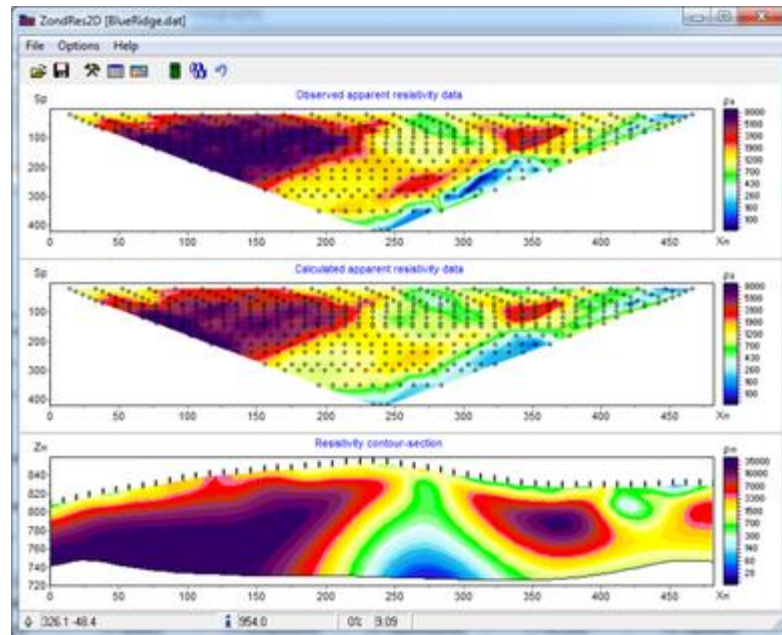
- Thiên văn nghiên cứu không gian và vũ trụ.
  - + Hình ảnh tinh vân Ring trong chòm sao Lyra, cách Trái Đất khoảng 2.000 năm ánh sáng, do kính viễn vọng không gian Hubble chụp và được công bố ngày 23/5/2013. Các chuyên gia đã đổ màu cho bức ảnh để minh họa thành phần hóa học của tinh vân.
  - + Vùng xanh đậm ở giữa tượng trưng cho heli, phần xanh nhạt ở vòng trong là hydro và oxy. Vòng ngoài màu cam đỏ cho thấy sự hiện diện của nitơ và lưu huỳnh.



Hình 2.4 tinh vân Ring

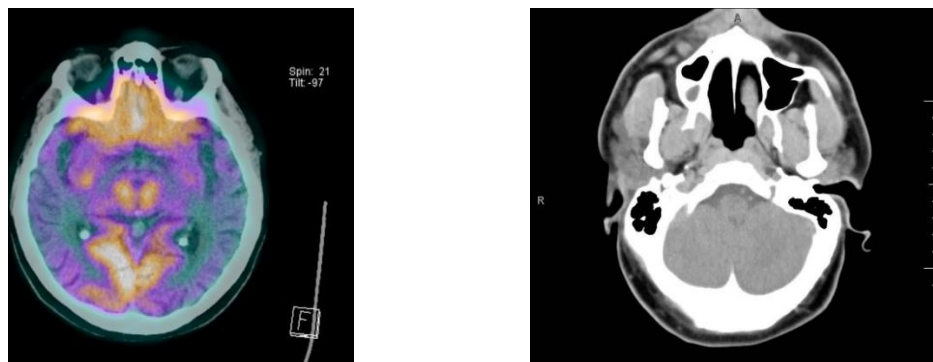
- Thăm dò địa chất.
  - + Trong lĩnh vực địa chất, hình ảnh nhận được từ vệ tinh có thể được phân tích để xác định cấu trúc bề mặt trái đất. Kỹ thuật làm nổi đường biên và khôi phục hình ảnh cho phép nâng cao chất lượng ảnh vệ tinh và tạo ra các bản đồ địa hình 3D với độ chính xác cao.
  - + Phương pháp được sử dụng cho lập bản đồ địa chất, khoáng sản, tìm nước ngầm, khảo sát địa chất công trình, địa chất môi trường và tai biến tự nhiên, khảo cổ học, tìm vật chưa nổ (UXO)... trên đất liền và trên biển gần bờ.

- + Mặt cắt ảnh điện, trong đó có mặt cắt ảnh điện hồ khoan, thường được thực hiện bằng phần mềm như Res2Dinv



Hình 2.5 Kết quả giải ngược Mặt cắt ảnh điện 2D bằng Res2Dinv

- Lĩnh vực y học.
  - + Trong y học các thuật toán xử lý ảnh cho phép biến đổi hình ảnh được tạo ra từ nguồn bức xạ X-ray hay nguồn bức xạ siêu âm thành hình ảnh quang học trên bề mặt phim x-quang hoặc trực tiếp trên bề mặt màn hình hiển thị.
  - + Hình ảnh các cơ quan chức năng của con người sau đó có thể được xử lý tiếp để nâng cao độ tương phản, lọc, tách các thành phần cần thiết (chụp cắt lớp) hoặc tạo ra hình ảnh trong không gian ba chiều (siêu âm 3 chiều).



Hình 2.6 Chụp cắt lớp phát xạ positron (PET) và CT não

- Robot và tự động hóa.
  - + Tùy vào từng ứng dụng cụ thể mà chúng ta sẽ có những hệ thống xử lý ảnh khác nhau. Một số ví dụ cho thấy xử lý ảnh được ứng dụng trong công nghiệp:
    - Trong công nghiệp đóng gói, người ta sử dụng hệ thống xử lý ảnh để kiểm tra xem các sản phẩm đã được dán nhãn chưa hoặc kiểm tra nhãn hiệu bao bì có đúng với thành phần chuẩn bị được đóng gói không.
    - Trong công nghiệp dược phẩm, áp dụng hệ thống xử lý ảnh để kiểm tra số lượng viên thuốc có trong vỉ thuốc.
    - Trong lĩnh vực điện, điện tử xử lý ảnh dùng để phát hiện sự thiếu sót các mối hàn sau khi hàn các chân linh kiện vào board mạch.
  - + Hiện nay, camera sử dụng trong công nghiệp có nhiều loại như: area scan camera, line scan camera và network camera.



Hình 2.7 Hệ thống xử lý ảnh công nghiệp - Machine Vision Delta với khả năng kết nối lên đến 8 camera tốc độ cao.

- Nhận diện khuôn mặt.
  - + Hệ thống nhận dạng khuôn mặt là một ứng dụng máy tính tự động xác định hoặc nhận dạng một người nào đó từ một bức hình ảnh kỹ thuật số hoặc một khung hình video từ một nguồn video. Một trong những cách để thực hiện điều này là so sánh các đặc điểm khuôn mặt chọn trước từ hình ảnh và một cơ sở dữ liệu về khuôn mặt.



- Image analysis (1,2): Phân tích hình ảnh là trích xuất thông tin có ý nghĩa từ hình ảnh, chủ yếu từ hình ảnh kỹ thuật số bằng các kỹ thuật xử lý hình ảnh kỹ thuật số. Nhiệm vụ phân tích hình ảnh có thể đơn giản như đọc các thẻ được mã hóa bằng thanh hoặc tinh vi như nhận dạng một người từ khuôn mặt của họ.

Image in → features out

- Computer vision (level 2,3): Thị giác máy tính là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh và, nói chung là dữ liệu đa chiều từ thế giới thực để cho ra các thông tin số hoặc biểu tượng, ví dụ trong các dạng quyết định. Việc phát triển lĩnh vực này có bối cảnh từ việc sao chép các khả năng thị giác con người bởi sự nhận diện và hiểu biết một hình ảnh mang tính điện tử. Các lĩnh vực con của thị giác máy tính bao gồm tái cấu trúc cảnh, dò tìm sự kiện, theo dõi video, nhận diện bố cục đối tượng, học, chỉ mục, đánh giá chuyển động và phục hồi ảnh.

Image in → interpretation out

- Computer graphic: Đồ họa máy tính là một nhánh của khoa học máy tính liên quan đến việc tạo ra hình ảnh với sự trợ giúp của máy tính. Ngày nay, đồ họa máy tính là một công nghệ cốt lõi trong nhiếp ảnh kỹ thuật số, phim ảnh, trò chơi video, điện thoại di động và màn hình máy tính và nhiều ứng dụng chuyên dụng. Rất nhiều phần cứng và phần mềm chuyên dụng đã được phát triển, với màn hình của hầu hết các thiết bị được điều khiển bởi phần cứng đồ họa máy tính.

Number in → image out

- Visualization: Trực quan hóa hoặc trực quan hóa là bất kỳ kỹ thuật nào để tạo hình ảnh, sơ đồ hoặc hình động để truyền đạt một thông điệp. Các trực quan hóa dựa trên mô hình hoặc đặt các lớp dữ liệu lên các hình ảnh thực tế hoặc được xây dựng bằng kỹ thuật số hoặc thực hiện một cấu trúc kỹ thuật số của một đối tượng thực trực tiếp từ dữ liệu khoa học. Các lĩnh vực truyền thống của trực quan khoa học là trực quan hóa dòng chảy, trực quan y tế, trực quan hóa vật lý và trực quan hóa học.

Image in → representation

#### 2.1.4. Một số thư viện nổi tiếng

##### 2.1.4.1. Scikit-image

- Scikit-image sử dụng mảng NumPy làm đối tượng hình ảnh bằng cách chuyển đổi các hình ảnh gốc. Những nparrays này có thể là số nguyên (đã ký hoặc chưa ký) hoặc số float. Và vì NumPy được xây dựng trong lập trình C, nó rất nhanh, làm cho nó trở thành một thư viện hiệu quả để xử lý hình ảnh.
- Trong số các phương pháp khác nhau, các nhà khoa học dữ liệu thường sử dụng kỹ thuật thang độ xám trong đó mỗi pixel là một màu xám.

##### 2.1.4.2. OpenCV

- Được phát hành lần đầu tiên vào năm 2000, OpenCV đã trở thành một thư viện phổ biến do tính dễ sử dụng và dễ đọc của nó. Thư viện tập trung vào xử lý hình ảnh, nhận diện khuôn mặt, phát hiện đối tượng và hơn thế nữa.
- Nó được viết bằng C++ nhưng cũng đi kèm với trình bao bọc Python và có thể hoạt động song song với NumPy, SciPy và Matplotlib. Được hỗ trợ bởi hơn một nghìn người đóng góp trên GitHub, thư viện thị giác máy tính tiếp tục nâng cao để xử lý hình ảnh dễ dàng.

##### 2.1.4.3. Mahotas

- Mahotas cho phép các nhà phát triển sử dụng các tính năng nâng cao của nó như haralick, các mẫu nhị phân cục bộ và hơn thế nữa.
- Nó có thể tính toán hình ảnh 2D và 3D thông qua mô-đun mahotas, features, haralick và thực hiện xử lý hình ảnh nâng cao bằng cách trích xuất thông tin từ hình ảnh.
- Mahotas có hơn 100 chức năng cho khả năng thị giác máy tính có thể cho phép bạn thực hiện các quy trình như đầu nguồn, xử lý hình thái, tích chập.

##### 2.1.4.4. SimpleITK

- Không giống như các thư viện khác coi hình ảnh là mảng, SimpleITK coi hình ảnh là một tập hợp các điểm trên một vùng vật lý trong không gian.

- Vùng chiếm bởi hình ảnh được xác định là ma trận cosine gốc, khoảng cách, kích thước và hướng. Modus operandi này cho phép nó xử lý hình ảnh hiệu quả. Nó hỗ trợ một loạt các kích thước bao gồm 2D, 3D và 4D.

#### 2.1.4.5. SciPy

- SciPy chủ yếu được sử dụng cho toán học và tính toán khoa học, nhưng cũng có thể thực hiện các thuật toán để thao tác hình ảnh bằng cách nhập mô-đun `scipy.ndimage`.
- SciPy có thể thực hiện hình thái nhị phân, đo lường đối tượng, lọc tuyến tính và phi tuyến tính. Bên cạnh đó, người ta có thể vẽ các đường đồng mức, điều chỉnh nội suy, bộ lọc, hiệu ứng, khử nhiễu và trích xuất và phân đoạn tương tự khác trên hình ảnh.

#### 2.1.4.6. Pillow

- Thư viện là phiên bản nâng cao của PIL, được hỗ trợ bởi Tidelift.
- Nó bao gồm các quy trình khác nhau trong xử lý hình ảnh như thao tác điểm, lọc, thao tác và hơn thế nữa. Pillow cũng hỗ trợ một loạt các định dạng hình ảnh, do đó làm cho thư viện phù hợp với công việc xử lý hình ảnh hơn.

#### 2.1.4.7. Picasso

- Đây là thư viện tải và lưu trữ hình ảnh mạnh mẽ dành cho Android, cho phép việc tải hình ảnh trong ứng dụng của bạn được trơn tru — chỉ với một dòng code
- Picasso giúp bạn giải quyết rất nhiều vấn đề phổ biến của việc hiển thị ảnh lên view như:

- + Xử lý `ImageView`, tái sử dụng hoặc hủy tải xuống trong adapter
- + Chuyển đổi hình ảnh phức tạp nhưng lại sử dụng rất ít tài nguyên bộ nhớ.
- + Hỗ trợ memory và disk caching.

#### 2.1.4.8. Matplotlib

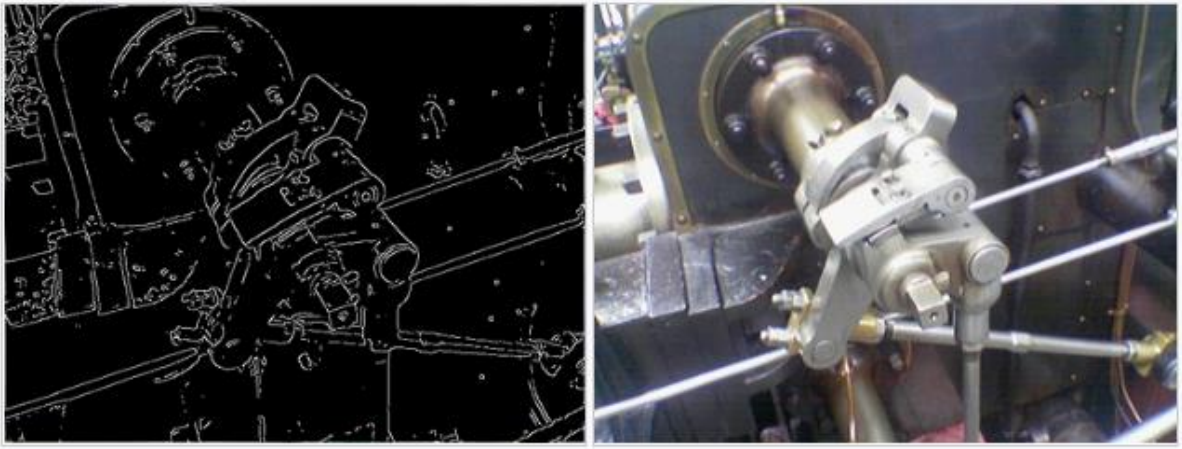
- Matplotlib chủ yếu được sử dụng để hiển thị 2D, nhưng nó cũng có thể được tận dụng để xử lý hình ảnh.
- Mặc dù nó không hỗ trợ tất cả các định dạng tệp, Matplotlib có hiệu quả trong việc thay đổi hình ảnh để trích xuất thông tin từ đó.



## 2.2 . Các phương pháp và kỹ thuật cụ thể

### 2.2.1.Trình bày đặc trưng Canny.

- Phương pháp phát hiện cạnh biên Canny là một toán tử phát hiện biên sử dụng thuật toán nhiều giai đoạn để phát hiện một loạt các biên ảnh. Nó được phát triển bởi John F. Canny vào năm 1986.
- Phát hiện biên Canny là một kỹ thuật để trích xuất thông tin cấu trúc hữu ích từ các đối tượng tầm nhìn khác nhau và giảm đáng kể lượng dữ liệu cần xử lý, các yêu cầu cho việc áp dụng phát hiện biên trên các hệ thống tầm nhìn đa dạng là tương đối giống nhau.



Hình 2.9 Máy dò biên Canny áp dụng cho một bức ảnh màu của động cơ hơi nước

- Các tiêu chí chung để phát hiện biên bao gồm:
  - + Phát hiện biên có tỷ lệ lỗi thấp, điều đó có nghĩa là phát hiện phải bất chính xác càng nhiều biên hiển thị trong ảnh càng tốt
  - + Điểm biên được phát hiện từ người vận hành nên định vị chính xác trên tâm của biên.
  - + Một biên đã cho trong ảnh chỉ nên được đánh dấu một lần và nếu có thể, nhiều hình ảnh sẽ không tạo ra các biên sai.
- Quá trình thuật toán phát hiện biên Canny.

Thuật toán phát hiện biên Canny có thể được chia thành 5 bước:

- 1) Áp dụng bộ lọc Gaussian để làm mịn hình ảnh để loại bỏ nhiễu.
- 2) Tìm độ dốc cường độ của hình ảnh.
- 3) Áp dụng triệt tiêu không tối đa để loại bỏ phản ứng giả để phát hiện biên.



4) Áp dụng ngưỡng kép để xác định các biên tiềm năng.

5) Theo dõi biên bằng độ trễ: Hoàn thiện việc phát hiện các biên bằng cách triệt tiêu tất cả các biên khác yếu và không được kết nối với các biên mạnh.

- Đặc trưng Canny trong thư viện OpenCV.

<code>edges=cv.Canny(image, threshold1, threshold2[, edges[, apertureSize[,L2gradient]]])</code>
--

<code>edges=cv.Canny(dx, dy, threshold1, threshold2[, edges[, L2gradient]])</code>
--

- Hàm tìm các biên trong ảnh đầu vào và đánh dấu chúng trong các biên của bản đồ đầu ra bằng thuật toán Canny. Giá trị nhỏ nhất giữa ngưỡng 1 và ngưỡng 2 được sử dụng cho liên kết biên. Giá trị lớn nhất được sử dụng để tìm các phân đoạn ban đầu của các biên mạnh.

#### 2.2.2. Ứng dụng phát hiện làn đường cho xe tự lái.

- Chiết xuất hình ảnh từ camera hoặc video, sau đó tiến hành xử lý sự biến dạng hình ảnh do hiện tượng méo lăng kính gây ra.
- Áp dụng bộ lọc ảnh màu kết hợp với đặc trưng Canny để phát hiện các biên ảnh và nhận được sự ổn định hơn của các điểm ảnh từ đó xác định các vật thể là làn đường.
- Sử dụng biểu đồ Histogram để cung cấp các thông tin và làm trực quan hình ảnh về làn đường.
- Tạo các điểm đánh dấu để có thể mô phỏng góc nhìn và tầm nhìn của camera.

### 2.3 . Tìm hiểu thư viện OpenCV

#### 2.3.1. Lịch sử phát triển

- OpenCV (Open Source Computer Vision) là một thư viện mã nguồn mở về thị giác máy với hơn 500 hàm và hơn 2500 các thuật toán đã được tối ưu về xử lý ảnh, và các vấn đề liên quan tới thị giác máy. OpenCV được thiết kế một cách tối ưu, sử dụng tối đa sức mạnh của các dòng chip đa lõi... để thực hiện các phép tính toán trong thời gian thực, nghĩa là tốc độ đáp ứng của nó có thể đủ nhanh cho các ứng dụng thông thường. OpenCV là thư viện được thiết kế để chạy trên nhiều nền tảng khác nhau (cross-platform), nghĩa là nó có thể chạy trên hệ điều hành Window, Linux, Mac, iOS ... Việc sử dụng thư viện OpenCV tuân theo các quy định về sử dụng phần mềm mã nguồn mở BSD do đó bạn

có thể sử dụng thư viện này một cách miễn phí cho cả mục đích phi thương mại lẫn thương mại.

- Dự án về OpenCV được khởi động từ những năm 1999, đến năm 2000 nó được giới thiệu trong một hội nghị của IEEE về các vấn đề trong thị giác máy và nhận dạng, tuy nhiên bản OpenCV 1.0 mãi tới tận năm 2006 mới chính thức được công bố và năm 2008 bản 1.1 (pre-release) mới được ra đời. Tháng 10 năm 2009, bản OpenCV thế hệ thứ hai ra đời (thường gọi là phiên bản 2.x), phiên bản này có giao diện của C++ (khác với phiên bản trước có giao diện của C) và có khá nhiều điểm khác biệt so với phiên bản thứ nhất.
- Thư viện OpenCV ban đầu được sự hỗ trợ từ Intel, sau đó được hỗ trợ bởi Willow Garage, một phòng thí nghiệm chuyên nghiên cứu về công nghệ robot. Cho đến nay, OpenCV vẫn là thư viện mở, được phát triển bởi nguồn quỹ không lợi nhuận (none - profit foundation) và được sự hưởng ứng rất lớn của cộng đồng.

### 2.3.2. Các chức năng và phiên bản sử dụng

Bảng 2.1 Các phiên bản OpenCV gần đây		
STT	Phiên bản	Ngày phát hành
1	4.0.0	2018-11-18
2	3.4.5	2018-12-22
3	4.0.1	2018-12-22
4	3.4.6	2019-04-08
5	4.1.0	2019-04-08
6	3.4.7	2019-07-26
7	4.1.1	2019-07-26
8	3.4.8	2019-11-12
9	4.1.2	2019-11-12
10	3.4.9	2019-12-23
11	4.2.0	2019-12-23
12	3.4.10	2020-04-06
13	4.3.0	2020-04-06

### 2.3.3. Chức năng của thư viện OpenCV

#### 2.3.3.1. Ứng dụng của thư viện OpenCV

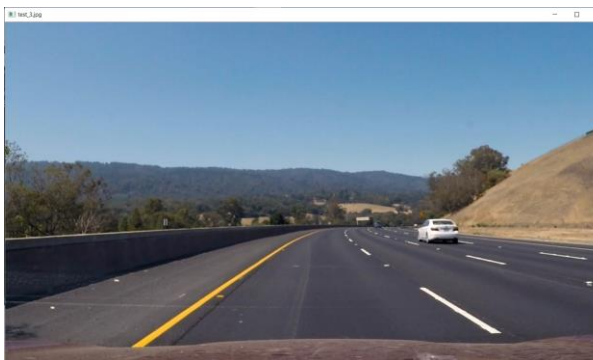
- OpenCV đang được sử dụng rộng rãi trong các ứng dụng bao gồm:
  - + Hình ảnh street view
  - + Kiểm tra và giám sát tự động
  - + Robot và xe hơi tự lái
  - + Phân tích hình ảnh y tế
  - + Tìm kiếm và phục hồi hình ảnh/video
  - + Phim - cấu trúc 3D từ chuyển động

#### 2.3.3.2. Chức năng của thư viện OpenCV

- Image/video I/O, xử lý, hiển thị (core, imgproc, highgui).
  - + Ảnh màu OpenCV tổ chức dữ liệu theo: Height, Width, Channel (gọi tắt HWC).
  - + Ảnh màu OpenCV sắp xếp 3 kênh theo: Blue, Green, Red (BGR).

```
1 import cv2
2 # doc anh
3 img = cv2.imread('test3.jpg')
4 # thay mau blue bang 0
5 #img[:, :, 0] = 0
6 # ghi anh
7 cv2.imwrite('test_3.jpg', img)
8 cv2.imshow('test_3.jpg', img)
9 cv2.waitKey(0)
10
```

Hình 2.10 Lệnh thay đổi màu điểm ảnh blue



Hình 2.11 Ảnh trước khi xử lý



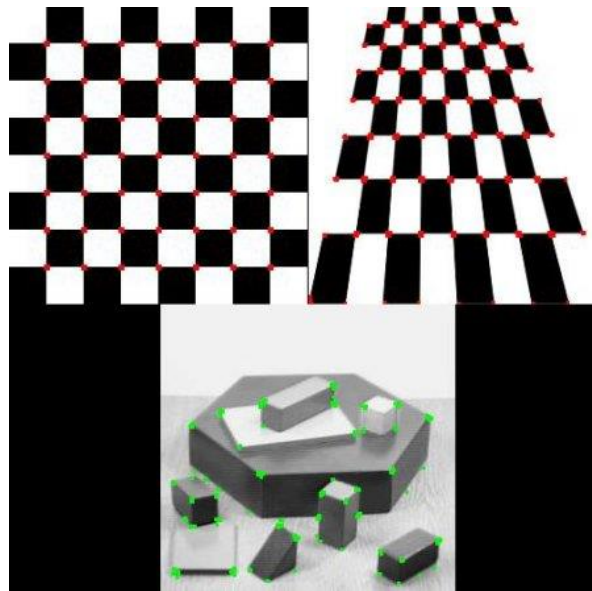
Hình 2.12 Ảnh sau khi xử lý

– Phát hiện và mô tả tính năng

- + OpenCV có chức năng `cv2.cornerHarris ()` cho mục đích này.
- + Đối số của nó là:
  - `img` - Hình ảnh đầu vào, nó phải là loại thang độ xám và kiểu `float32`.
  - `blockSize` - Đó là kích thước của vùng lân cận được xem xét để phát hiện góc
  - `ksize` - Tham số khẩu độ của đạo hàm Sobel được sử dụng.
  - `k` - Harris dò tham số miễn phí trong phương trình.

```
1 import cv2
2 import numpy as np
3
4 filename = 'test3.jpg'
5 img = cv2.imread(filename)
6 gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
7
8 gray = np.float32(gray)
9 dst = cv2.cornerHarris(gray,2,3,0.04)
10
11 #result được mở rộng để đánh dấu các góc, không quan trọng
12 dst = cv2.dilate(dst,None)
13
14 # Ngưỡng cho một giá trị tối ưu, nó có thể thay đổi tùy theo hình ảnh.
15 img[dst>0.01*dst.max()]=[0,0,255]
16
17 cv2.imshow('dst',img)
18 if cv2.waitKey(0) & 0xff == 27:
19     cv2.destroyAllWindows()
```

Hình 2.13 chương trình xử lý bằng `cv2.cornerHarris ()`



Hình 2.14 hình ảnh được xử lý phát hiện góc `cv2.cornerHarris ()`

- Video Analysis (Meanshift and Camshift, Optical Flow, Background Subtraction).
  - + Background Subtraction là một thuật toán phân đoạn nền / tiền cảnh dựa trên hỗn hợp Gaussian. Một đặc điểm quan trọng của thuật toán này là nó chọn số lượng phân phối gaussian thích hợp cho mỗi pixel.
  - + Bóng sẽ được đánh dấu màu xám. detectShadows = True

```

1 import numpy as np
2 import cv2
3
4 cap = cv2.VideoCapture('vtest.avi')
5
6 fgbg = cv2.createBackgroundSubtractorMOG2()
7
8 while(1):
9     ret, frame = cap.read()
10
11     fgmask = fgbg.apply(frame)
12
13     cv2.imshow('frame', fgmask)
14     k = cv2.waitKey(30) & 0xff
15     if k == 27:
16         break
17
18 cap.release()
19 cv2.destroyAllWindows()

```

Hình 2.15 Cài đặt thuật toán BackgroundSubtractorMOG2 với  
cv2.createBackgroundSubtractorMOG2()



Hình 2.16 Khung hình gốc



2.17 Kết quả của  
BackgroundSubtractorMOG2

- + Vùng màu xám hiển thị vùng bóng
- Machine learning & clustering (ml, flann, KNN, K-means, SVM).
  - + K-Means Clustering trong OpenCV: cv2.kmeans ()

- + Phân cụm K-Means là thuật toán học máy không giám sát nhằm phân vùng N quan sát thành các cụm K trong đó mỗi quan sát thuộc về cụm có giá trị trung bình gần nhất.
- + `cv2.TERM_CRITERIA_EPS` - dừng lặp lại thuật toán nếu đạt độ chính xác được chỉ định, epsilon đạt được.
- + `cv2.TERM_CRITERIA_MAX_ITER` - dừng thuật toán sau số lần lặp được chỉ định, `max_iter`.
- + `cv2.TERM_CRITERIA_EPS` + `cv2.TERM_CRITERIA_MAX_ITER` - dừng lặp lại khi bất kỳ điều kiện nào ở trên được đáp ứng.
- + `cv2.KMEANS_PP_CENTERS` và `cv2.KMEANS_RANDOM_CENTERS` - xác định cách lấy trung tâm ban đầu.
- + Ví dụ một ảnh có 3 kênh màu chính là R, G, B để định hình lại hình ảnh thành một mảng có kích thước  $M \times 3$  (M là số pixel trong hình ảnh). Và sau khi phân cụm áp dụng các giá trị centroid (cũng là R, G, B) cho tất cả các pixel, sao cho hình ảnh thu được sẽ có số lượng màu được chỉ định.

```

1 import numpy as np
2 import cv2
3
4 img = cv2.imread('home.jpg')
5 Z = img.reshape((-1,3))
6 # ép kiểu sang np.float32
7 Z = np.float32(Z)
8 # tạo criteria, clusters(K) và kmeans()
9 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
10 K = 8
11 ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)
12
13 center = np.uint8(center)
14 res = center[label.flatten()]
15 res2 = res.reshape((img.shape))
16
17 cv2.imshow('res2',res2)
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
20 |

```

Hình 2.18 cài đặt Color Quantization



Hình 2.19 kết quả sau K lần định lượng lại kênh màu

- Phát hiện đối tượng
  - + Phát hiện khuôn mặt với Haar Cascades.
  - + OpenCV đã chứa nhiều phân loại được đào tạo trước cho khuôn mặt, mắt, nụ cười... Những tệp XML đó được lưu trữ trong nguồn cài đặt `opencv/data/haarcascades/`

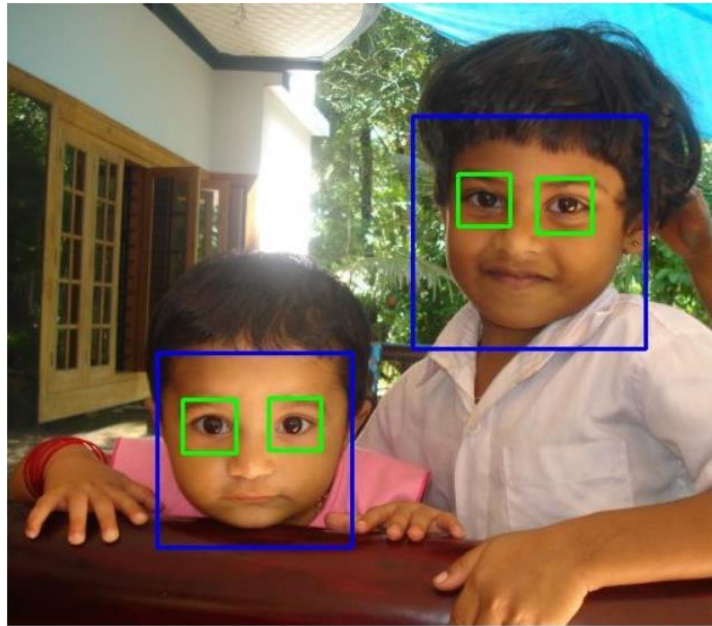
```

1 import numpy as np
2 import cv2
3
4 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
5 eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
6
7 img = cv2.imread('sachin.jpg')
8 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
9 faces = face_cascade.detectMultiScale(gray, 1.3, 5)
10 for (x,y,w,h) in faces:
11     img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
12     roi_gray = gray[y:y+h, x:x+w]
13     roi_color = img[y:y+h, x:x+w]
14     eyes = eye_cascade.detectMultiScale(roi_gray)
15     for (ex,ey,ew,eh) in eyes:
16         cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
17
18 cv2.imshow('img',img)
19 cv2.waitKey(0)
20 cv2.destroyAllWindows()

```

## 2.20 Cài đặt nhận diện khuôn mặt với `cv2.CascadeClassifier()`





2.21 Kết quả nhận diện khuôn mặt và mắt với cv2.CascadeClassifier()

#### 2.3.4. Các nhóm hàm trong thư viện openCV

##### - cv2.Canny()

```
cv2.Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]])  
→ edges
```

- + Hàm tìm các cạnh trong ảnh đầu vào image và đánh dấu chúng trong bản đồ đầu ra edges bằng thuật toán Canny.
- + Giá trị nhỏ nhất giữa threshold1 và threshold2 được sử dụng cho liên kết cạnh. Giá trị lớn nhất được sử dụng để tìm các phân đoạn ban đầu của các cạnh mạnh.
- + image – single-channel 8-bit input image.
- + edges – output edge map; it has the same size and type as image

##### - cv2.GaussianBlur()

```
cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]]) → dst
```

- + Làm mờ hình ảnh bằng bộ lọc Gaussian
- + src - hình ảnh đầu vào; hình ảnh có thể có bất kỳ số lượng các kênh truyền hình, được xử lý một cách độc lập, nhưng độ sâu nên CV\_8U, CV\_16U, CV\_16S, CV\_32F hoặc CV\_64F.
- + dst - hình ảnh đầu ra có cùng kích thước và loại như src.



- cv2.dilate()

```
cv2.dilate(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]) → dst
```

- + Làm giãn một hình ảnh bằng cách sử dụng một yếu tố cấu trúc cụ thể.
- + Hàm làm giãn hình ảnh nguồn bằng cách sử dụng phần tử cấu trúc đã chỉ định để xác định hình dạng của vùng lân cận pixel mà trên đó lấy mức tối đa

- cv2.erode()

```
cv2.erode(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]) → dst
```

- + Làm mờ hình ảnh bằng cách sử dụng một yếu tố cấu trúc cụ thể.
- + Hàm xóa hình ảnh nguồn bằng cách sử dụng phần tử cấu trúc đã chỉ định để xác định hình dạng của vùng lân cận pixel mà trên đó mức tối thiểu.

- cv2.circle()

```
cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]) → None
```

- + Vẽ một vòng tròn.
- + Hàm vẽ một vòng tròn đơn giản hoặc đầy với một tâm và bán kính cho trước.

- cv2.undistort()

```
cv2.undistort(src, cameraMatrix, distCoeffs[, dst[, newCameraMatrix]]) → dst
```

- + Chuyển đổi hình ảnh để bù cho hiện tượng méo ống kính.
- + Những pixel trong ảnh đích, không có pixel tương ứng trong ảnh nguồn, được lấp đầy bằng số không (màu đen).

- cv2.warpPerspective

```
cv2.warpPerspective(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) → dst
```

- + Perspective transformation là phép biến đổi sao cho các điểm nằm trên một đường thẳng ở ảnh đầu vào vẫn nằm trên cùng một đường thẳng ở ảnh đầu ra.
- + Hàm này nhận tham số đầu vào là vị trí của 4 điểm ở bức ảnh đầu vào và vị trí tương ứng của chúng ở bức ảnh đầu ra.

- cv2.addWeighted()

```
cv2.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]]) → dst
```

- + Tính tổng trọng số của hai mảng

## CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG

### 3.1 . Nêu bài toán

- Ngày nay, các nghiên cứu về khả năng tự lái cho một hệ thống trợ lý lái xe tiên tiến (ADAS) đã nhận được nhiều sự chú ý lớn. Một trong những mục tiêu chính của lĩnh vực nghiên cứu này là cung cấp chức năng thông minh và an toàn hơn cho người lái xe bằng cách sử dụng công nghệ thông tin và điện tử. Trong điều kiện đường xá đặc thù thì khả năng nhận biết và phát hiện các biển báo đường, làn đường và đèn giao thông là rất quan trọng và đóng vai trò quan trọng đối với các hệ thống ADAS.
- Nhận thấy tầm quan trọng và tính ứng dụng cao của việc nhận diện làn đường cho xe tự lái. Yêu cầu bài toán đặt ra ở đây là việc nhận dạng được làn đường để hỗ trợ cho hệ thống xe tự lái bằng cách ứng dụng và kết hợp các kiến thức được học và nghiên cứu trong học phần “Nhập môn xử lý ảnh”.

### 3.2 . Các bước thực hiện

#### 3.2.1. Bước 1: Đọc và tiền xử lý hình ảnh.

- Dữ liệu đọc vào là hình ảnh triết xuất từ camera hoặc video.
- Sau đó tiến hành các bước tiền xử lý điều chỉnh và chuyển đổi hình ảnh để bù cho hiện tượng méo ảnh do lăng kính.

+ Ví dụ mô phỏng về kỹ thuật điều chỉnh hiện tượng méo ảnh do lăng kính gây ra.



Hình 3.1 Ảnh trước khi xử lý



Hình 3.2 Ảnh đã qua xử lý

- + Áp dụng hàm `cv2.undistort(img, mtx, dist, None, mtx)` để xử lý. Hàm này chỉ đơn giản là sự kết hợp của sự thống nhất và phép nội suy song tuyến tính.



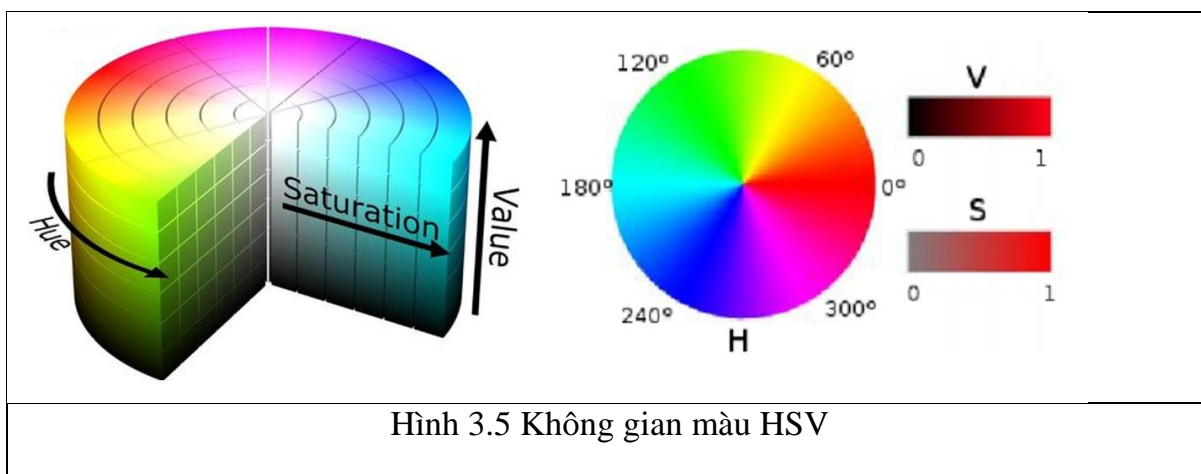
Hình 3.3 Ảnh trước khi xử lý



Hình 3.4 Ảnh đã được xử lý

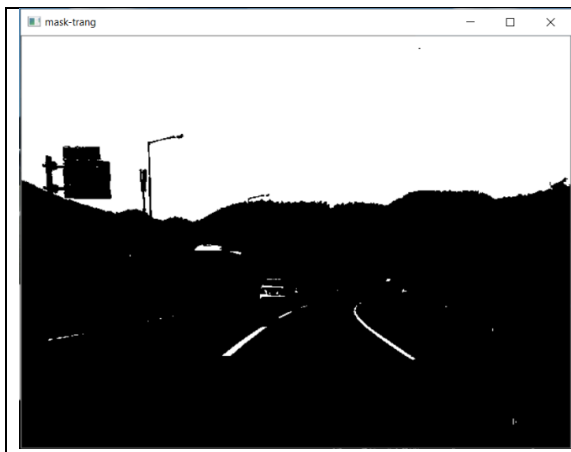
### 3.2.2. Bước 2: Tạo xử lý hình ảnh.

- Các làn đường thường có độ tương phản cao hơn so với mặt đường và thường được đánh dấu bằng màu vàng và màu trắng. Áp dụng kỹ thuật tạo bộ lọc màu với mask màu trắng và mask màu vàng và kết hợp với máy dò cạnh để làm rõ hơn các làn đường.

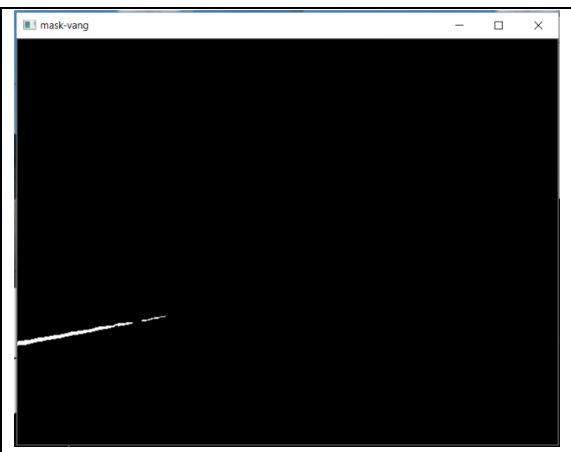


Hình 3.5 Không gian màu HSV

- + Áp dụng `cv2.inRange(src, lower, upper)` dùng để phát hiện một đối tượng dựa trên phạm vi giá trị pixel trong không gian màu HSV.
- + Sau đó tạo 2 lớp Mask trắng và vàng, chỉ giữ lại những chi tiết bên trong vùng ảnh và bỏ hết các phần khác (bôi đen) bằng cách áp dụng hàm `cv2.bitwise_or(maskedWhite, maskedYellow)` tính độ lệch của các bit theo từng mảng màu giữ 2 mảng màu trắng và màu vàng.



Hình 3.6 Mask ảnh trắng



Hình 3.7 Mask ảnh vàng



Hình 3.8 Ảnh áp dụng `cv2.bitwise_or(maskedWhite,maskedYellow)`

- Xử lý phân ngưỡng tách biên cho ảnh.
  - + Trước khi tiến hành xử lý nhận diện biên ảnh, sẽ chuyển ảnh về chế độ ảnh xám với `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`.
  - + Kế tiếp hành nâng cao chất lượng ảnh với `cv2.GaussianBlur(imgGray, (5, 5), 0)`.
  - + Áp dụng đặt trưng Canny() của opencv để nhận diện biên ảnh.
  - + Tiếp tục nâng cao chất lượng ảnh sau nhận diện biên bằng phương pháp tăng độ ổn định cho các điểm ảnh với việc làm giãn nở các điểm ảnh và giảm độ sói mòn điểm ảnh với `imgDial = cv2.dilate(imgCanny,kernel,iterations=1)` với độ giãn nở điểm ảnh là 1 và `imgErode = cv2.erode(imgDial,kernel,iterations=1)` với độ sói mòn là 1.



Hình 3.9 Ảnh được xử lý phát hiện biên

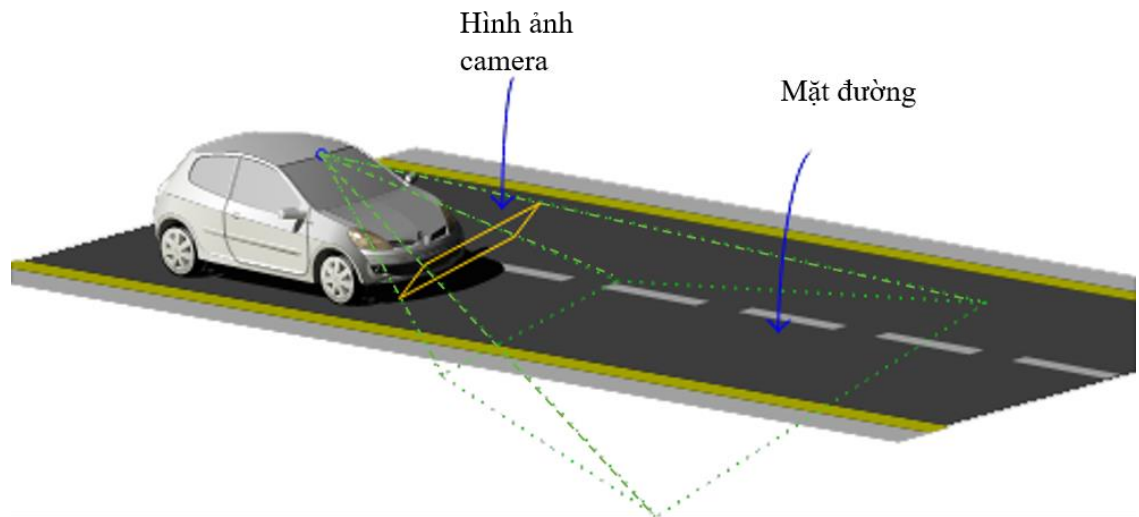
- + Cuối cùng là kết hợp ảnh đã được xử lý bộ lọc màu và nhận diện biên ảnh lại với nhau để tăng độ trung quan.



Hình 3.10 Ảnh kết hợp xử lý bộ lọc màu và phát hiện biên

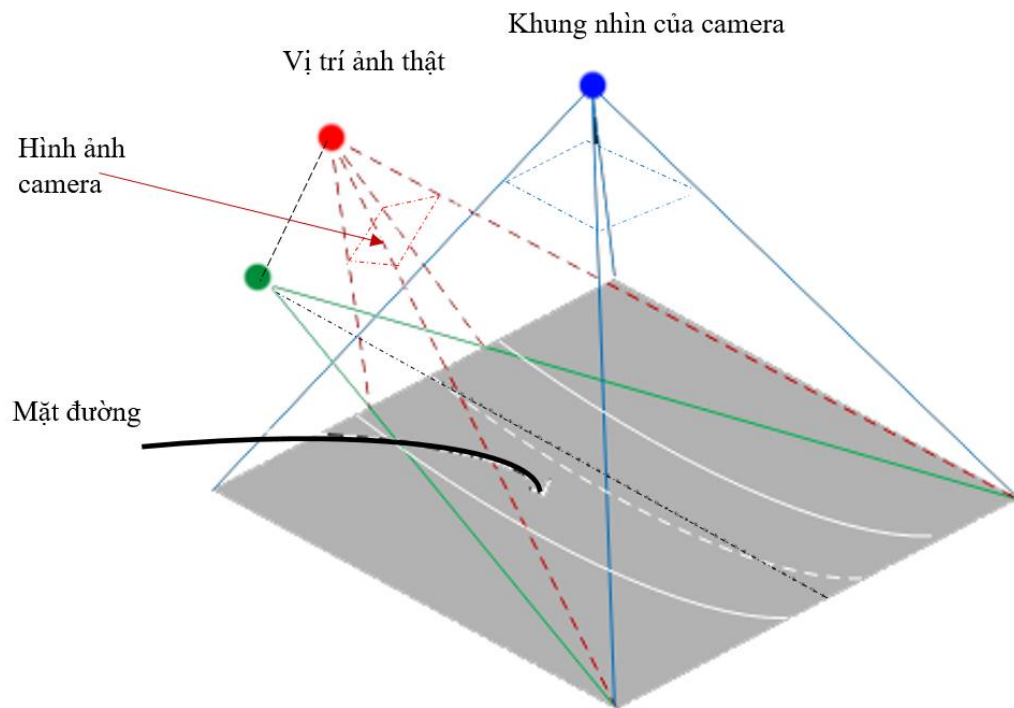
### 3.2.3. Bước 3: Xử lý nhận dạng làn đường

- Chuyển đổi góc nhìn của camera để xác định làn đường từ đó đưa ra các tính toán cần thiết để nhận dạng.
- Bằng cách giả sử rằng làn đường nằm trên một bề mặt 2D phẳng, rồi điều chỉnh một đa thức có thể biểu thị chính xác làn đường trong không gian.



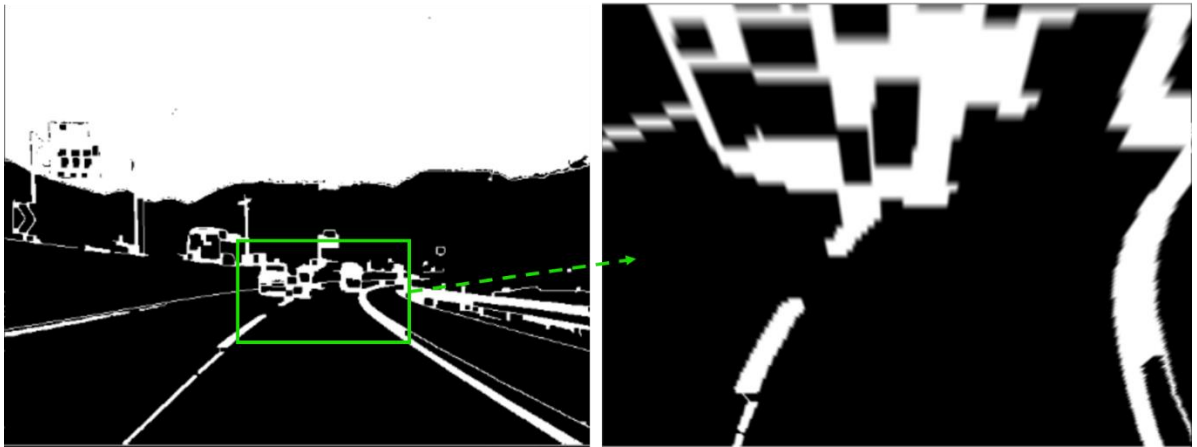
Hình 3.11 Hình ảnh mô phỏng khung nhìn của camera đặt trước xe

- Khi chuyển đổi chế độ xem từ phía trên trong đó chế độ xem camera thực được chuyển thành vị trí ảo với góc nhìn trực tiếp từ trên xuống. Để tìm ra mối quan hệ chuyển đổi giữa hình ảnh xem trước và hình ảnh xem từ phía trên.



Hình 3.12 Mô hình góc nhìn trong không gian từ phía trên của mặt đường

- Bằng cách sử dụng hàm `cv2.getPerspectiveTransform()` để lấy ma trận biến đổi và `cv2.warpPerspective()` để áp dụng nó cho hình ảnh.



Hình 3.13 hình chiếu góc nhìn từ trên xuống

- Áp dụng kỹ thuật phân ngưỡng để tạo độ tương phản hơn cho hình ảnh. Và dùng nó để xác định vị trí các điểm ảnh của phần làn đường.



Hình 3.14 mô phỏng biểu đồ histogram của ảnh

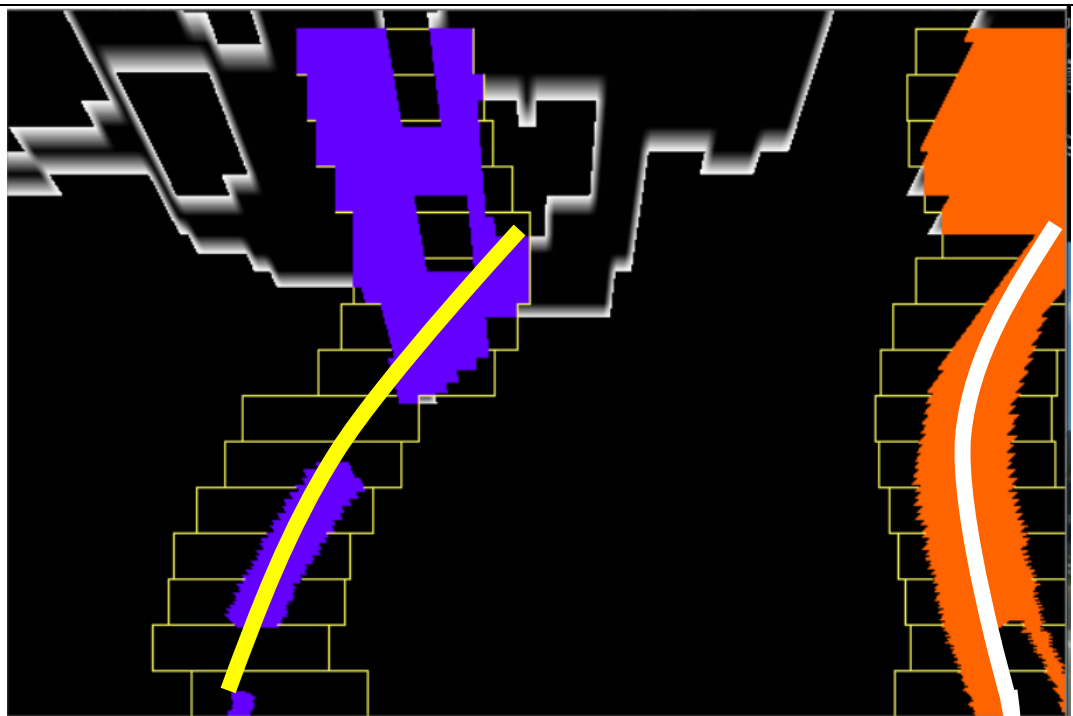


Hình 3.15 Biểu đồ histogram của ảnh

- Biểu đồ histogram được chiếu theo trục Ox. Biểu đồ sẽ biểu diễn các điểm ảnh màu trắng ở hình 3.14 thành các đỉnh. Chọn 2 đỉnh cao nhất thì đây chính là phần làn đường.
- Để phân biệt ranh giới giữa 2 làn đường trái phải chúng em áp dụng thuật toán Sliding Window.
- Để tìm ra hình ảnh làn đường ở các đoạn cua (cong) chúng em áp dụng áp dụng hồi quy đa thức cho từng điểm ảnh của 2 đỉnh trái và phải bằng cách sử dụng np.polyfit() kết hợp với công thức tìm bán kính đường cong để xác định các điểm trên đường cong (làn đường). Vì đường cong được cho bởi phương trình trong hệ tọa độ Descartes:  $y=f(x)$  nên bán kính cong được tính như sau:

$$R = \left| \frac{\left[ 1 + \left( \frac{dy}{dx} \right)^2 \right]^{3/2}}{\frac{d^2y}{dx^2}} \right|$$

- Với hàm  $f(x) = ax^2+bx+c$
- Sau khi nhận dạng được đường cong của các làn đường từ góc nhìn từ trên xuống thì ta tiến hành chuyển đổi lại góc nhìn và tiến hành vẽ mô phỏng lại làn đường.

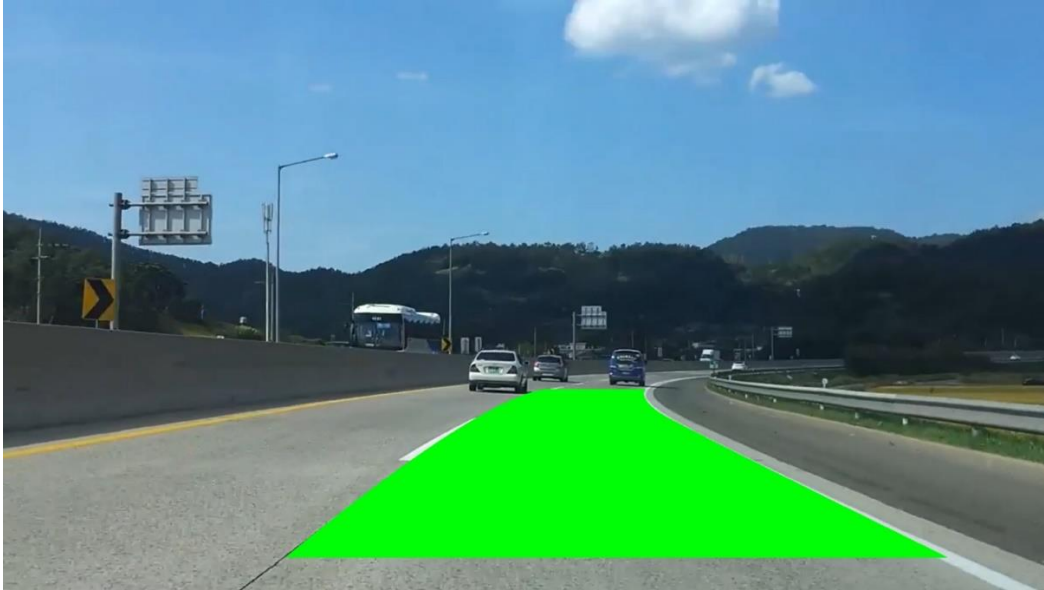


Hình 3.16 biểu đồ đường cong thể hiện làn đường trái và phải



#### 3.2.4. Bước 4: Đưa hình ảnh lên màn hình

- Sau khi qua các bước xử lý nhận dạng làn đường thì sẽ phải chuyển đổi lại góc nhìn như trước và cần được vẽ 1 lớp phủ nhận diện lên mặt đường lớp phủ này được giới hạn bởi 4 điểm tọa độ được xác định bởi số liệu được đưa vào từ bảng điều khiển.



Hình 3.17 Kết quả của quá trình xử lý phát hiện nhận diện làn đường

#### 3.2.5. Bước 4: Tạo bảng điều khiển tọa độ cho lớp phủ làn đường.

- Tạo một cửa sổ mới với `cv2.namedWindow("Control")`. Cửa sổ này sẽ hiển thị 4 Trackbar dùng để điều chỉnh vị trí của 4 tọa độ trên trái, trên phải, dưới trái và dưới phải.
- Sau khi có được tọa độ của 4 điểm giới hạn thì dùng `cv2.circle()` với `cv2.FILLED` để vẽ 4 điểm này.



Hình 3.18 Giao diện bảng điều khiển và 4 điểm tọa độ

### 3.2.6. Bước 6: Tạo giao diện đồ họa

- Phần giao diện chính được chia thành các frame để chứa các khung hình và các nút chức năng, các tùy chọn menu và thông tin về đề tài.

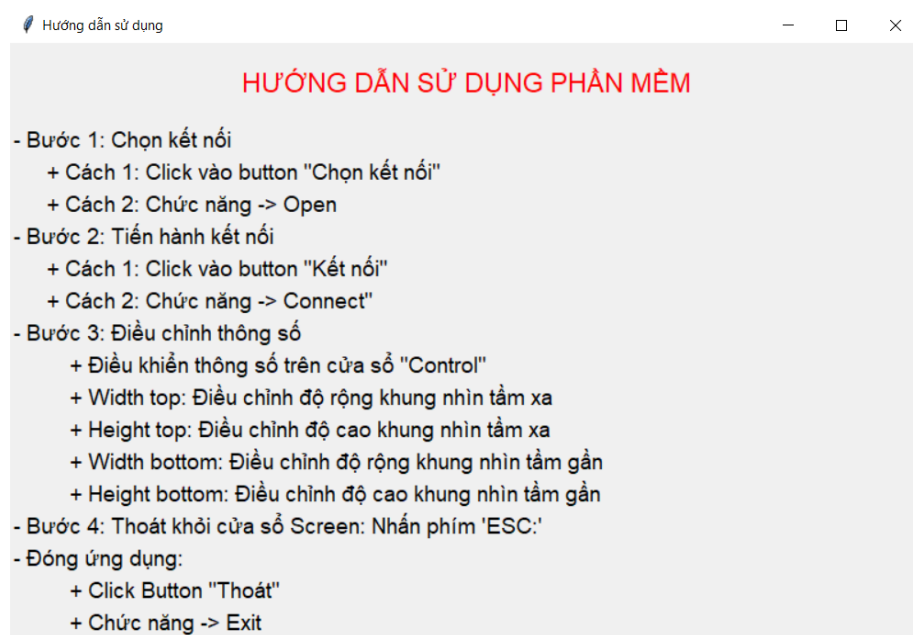


Hình 3.19 giao diện chính

### 3.3 . Hướng dẫn sử dụng

- Các thư viện hỗ trợ bao gồm:
  - + Ứng dụng được viết bằng ngôn ngữ lập trình Python với giao diện đồ họa Tkinter và thư viện đồ họa Opencv.
  - + Python: v3.8.1
  - + OpenCV: v4.2.0
  - + Numpy: v1.18.1
  - + PIL: V7.0.0
- Gồm các chức năng:
  - + Mở chọn video hình ảnh.
  - + Nhận diện làn đường.
  - + Điều chỉnh góc nhận diện.

- + Giao diện giới thiệu về nhón thực hiện và hướng dẫn sử dụng.
- Hướng dẫn sử dụng:
  - + Bước 1: Chọn kết nối
    - Cách 1: Click vào button "Chọn kết nối"
    - Cách 2: Chức năng -> Open
  - + Bước 2: Tiến hành kết nối
    - Cách 1: Click vào button "Kết nối"
    - Cách 2: Chức năng -> Connect
  - + Bước 3: Điều chỉnh thông số
    - Điều khiển thông số trên cửa sổ "Control"
    - Width top: Điều chỉnh độ rộng khung nhìn tầm xa
    - Height top: Điều chỉnh độ cao khung nhìn tầm xa
    - Width bottom: Điều chỉnh độ rộng khung nhìn tầm gần
    - Height bottom: Điều chỉnh độ cao khung nhìn tầm gần
  - + Bước 4: Thoát khỏi cửa sổ Screen: Nhấn phím "ESC":
  - + Đóng ứng dụng:
    - Cách 1: Click Button "Thoát"



Hình 3.20 Cửa sổ hướng dẫn sử dụng

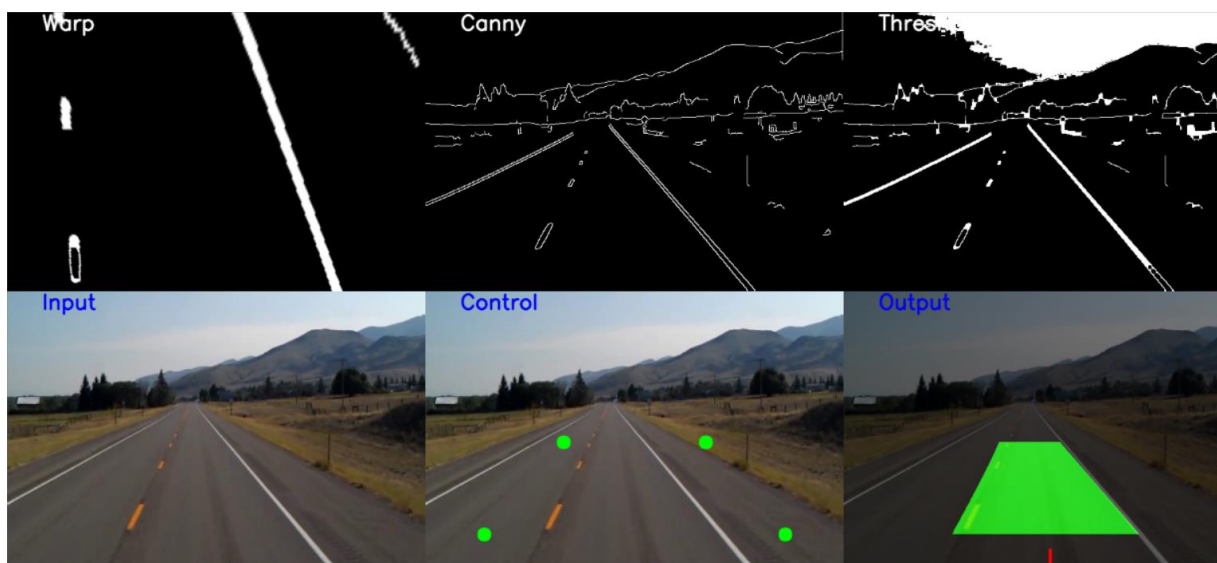
### 3.4 . Đánh giá về kết quả và đề xuất hướng phát triển

#### 3.4.1. Kết quả đạt được

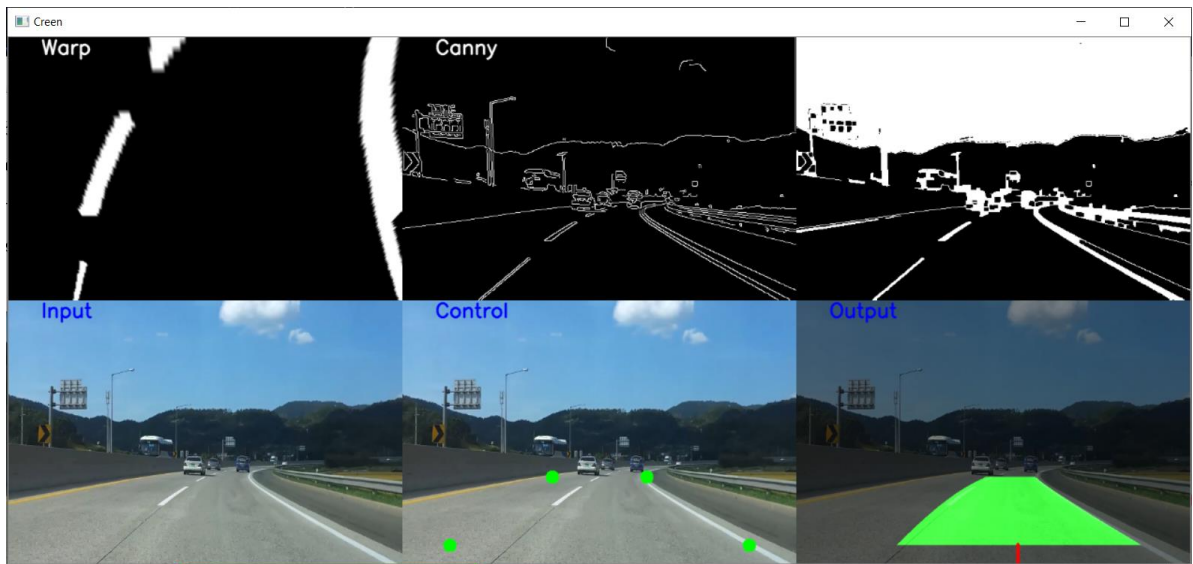
- Xây dựng được ứng dụng nhận diện làn đường dành cho xe tự lái bằng kiến thức đã học.
- Giải quyết được yêu cầu đặt ra của bài toán tìm và nhận diện làn đường.
- Nhận diện được làn đường đối với cả tuyến đường cong.
- Ứng dụng có giao diện đồ họa đơn giản và dễ sử dụng.



Hình 3.21 Giao diện ứng dụng



Hình 3.22 nhận diện làn đường với tuyến đường thẳng



Hình 3.23 nhận diện làn đường với tuyến đường cong

#### 3.4.2. Hạn chế

- Kết quả nhận diện ở các tuyến đường cong còn hạn chế.
- Phát hiện và nhận diện sẽ gặp khó khăn khi có vật thể che đi lane đường, lane mờ.
- Chưa tính được độ lệch của xe với làn đường.

#### 3.4.3. Hướng phát triển

- Cải tiến và tối ưu các thuật toán để có thể nhận diện hiệu quả hơn.
- Nâng cấp ứng dụng có thể nhận diện các vật thể khác như biển báo giao thông, đèn tín hiệu.
- Tối ưu hóa giao diện.

## TÀI LIỆU THAM KHẢO

- 1) Giáo trình: Xử lý ảnh - PGS.TS Nguyễn Quang Hoan
- 2) Giáo trình: OpenCV Computer Vision Projects with Python v2016
- 3) Giáo trình: Xử lý ảnh - Hoàng Văn Hiệp năm 2011
- 4) Sách điện tử: <https://opencv-python-tutroals.readthedocs.io/>
- 5) Bài báo khoa học: A Precise Lane Detection Algorithm Based on Top View Image Transformation and Least-Square Approaches ([dx.doi.org/10.1155/2016/4058093](https://doi.org/10.1155/2016/4058093))