

TRƯỜNG ĐẠI HỌC TRẦN ĐẠI NGHĨA
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO

BÀI TẬP LỚN MÔN HỌC NHẬP MÔN XỬ LÝ ẢNH

**Đề tài: “Tìm hiểu thư viện OpenCV, ngôn ngữ Python
Viết ứng dụng phát hiện đoạn thẳng và hình tròn bằng
phép biến đổi Hough”**

TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2019

BÁO CÁO

BÀI TẬP LỚN MÔN HỌC NHẬP MÔN XỬ LÝ ẢNH

**Đề tài: “Tìm hiểu thư viện OpenCV, ngôn ngữ Python
Viết ứng dụng phát hiện đoạn thẳng và hình tròn bằng
phép biến đổi Hough”**

GVHD: Ngô Thanh Tú

Sinh viên thực hiện:

Nguyễn Lê Xuân Phước

Tôn Nữ Nguyên Hậu

Đỗ Tổng Quốc

Lớp: 16DDS06031

TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2019

MỤC LỤC

DANH MỤC HÌNH ẢNH	5
LỜI MỞ ĐẦU	6
CHƯƠNG 1: TÌM HIỂU NGÔN NGỮ LẬP TRÌNH PYTHON	7
1.1. Giới thiệu ngôn ngữ Python	7
1.1.1. Lịch sử phát triển	7
1.1.2. Phiên bản của Python.....	9
1.1.3. Một điểm khác nhau giữa phiên bản 3.x và 2.x.....	9
1.1.4. Đặc điểm của Python	15
1.2. Hướng dẫn cài đặt Ananconda	24
1.2.1. Giới thiệu Ananconda	24
1.2.2. Download Anaconda và hướng dẫn cài đặt trên HĐH Window.....	25
1.2.3. Hướng dẫn cài đặt thêm thư viện	30
1.3. Hướng dẫn cài đặt IDE Spyder	32
1.3.1. Giới thiệu IDE Spyder.....	32
1.3.2. Hướng dẫn cài đặt IDE Spyder bằng Navigator	33
1.3.4. Hướng dẫn sử dụng Spyder	34
CHƯƠNG 2: TÌM HIỂU MÔN HỌC XỬ LÝ ẢNH	37
2.1. Các khái niệm cơ bản	37
2.1.1. Ảnh.....	37
2.1.2. Ảnh số	38
2.1.3. Hệ thống xử lý ảnh.....	38
2.1.4. Biểu diễn ảnh	39
2.2. Xử lý ảnh là gì?	40
2.2.1. Lịch sử phát triển	40
2.2.2. Các lĩnh vực ứng dụng.....	41
1.2.3. Một số thư viện nổi tiếng trong xử lý ảnh	43
2.3. Phương pháp và kỹ thuật phép biến đổi Hough	45
2.3.1. Tổng quan về biến đổi Hough (HT- Hough transform)	45
2.3.2. Biến đổi Hough là gì?	45
2.3.3. Biến đổi Hough để phát hiện các đường thẳng trong một hình ảnh.....	46
2.3.4. Biến đổi Hough để phát hiện các hình tròn trong một hình ảnh	50

2.3. Giới thiệu thư viện OpenCV	55
2.3.1. Lịch sử phát triển	55
2.3.2. Các phiên bản của thư viện OpenCV	56
2.3.3. Phiên bản OpenCv nhóm đang sử dụng	56
2.3.4. Chức năng thư viện OpenCV.....	56
2.3.5. Sơ lược về cấu trúc của thư viện OpenCV	56
2.3.6. Cài đặt thư viện OpenCV.....	58
CHƯƠNG 3: ỨNG DỤNG PHÁT HIỆN ĐOẠN THẲNG VÀ HÌNH TRÒN BẰNG PHÉP BIẾN ĐỔI HOUGH	60
3.1. Bài Toán	60
3.2. Giao diện chức năng chương trình	60
3.3. Một số kết quả chương trình.	61
KẾT LUẬN	62
TÀI LIỆU THAM KHẢO	63

DANH MỤC HÌNH ẢNH

Hình 1. 1: Guido van Russom.....	7
Hình 1.2:Giao diện trang chủ Anaconda.....	25
Hình 1. 3: Trang tải Anaconda.....	26
Hình 1. 4: Chạy chương trình vừa tải về.....	26
Hình 1. 5: License Agreement	27
Hình 1. 6:Chọn Loại cài đặt	27
Hình 1. 7: Chọn Vị trí cài đặt.....	28
Hình 1. 8: Advanced Options.....	28
Hình 1. 9:Hoàn thành Cài đặt.....	29
Hình 1. 10: Anaconda và JetBrains.....	29
Hình 1. 11: Cài đặt hoàn tất và khởi động chương trình.....	30
Hình 1. 12:Giao diện chính của IDE Spyder	32
Hình 1. 13:Chạy Spyder.....	33
Hình 1. 14:Cài đặt Spyder bằng Anaconda Navigator.....	34
Hình 1. 15: Mở chương trình Spyder.....	34
Hình 1. 16:Giao diện chính Spyder.....	35
Hình 1. 17: Lưu chương trình với đuôi *.py	35
Hình 1. 18: Chạy chương trình.....	36
Hình 2.1: Quá trình chuyển đổi từ ảnh tương tự sang ảnh số	37
Hình 2.2: Mô hình biểu diễn một hệ thống xử lí ảnh.....	39
Hình 2.3: Biểu diễn ảnh bằng ma trận điểm	41
Hình 2.4: Hình ảnh trước và sau khi tăng cường	42
Hình 2.5: Hình ảnh trước và sau khi lọc nhiễu	54
Hình 2.6: Đường thẳng trong hệ tọa độ Polar.....	54
Hình 2.7: Sơ đồ hình sin đi qua điểm (x_0, y_0)	54
Hình 2.8: Sơ đồ hình Sin đi qua 3 điểm.....	54
Hình 2.9: Phát hiện dòng bằng Hough Transform.....	54
Hình 2.10: Hình tròn	54
Hình 2.11: Phát hiện hình tròn bằng Hough	54
Hình 2.12: Phát hiện hình tròn bằng Hough Transform	54
Hình 3.1: Giao diện chính	60
Hình 3.2: Kết quả thực hiện phát hiện đoạn thẳng.....	61
Hình 3.3: Kết quả thực hiện tìm hình tròn bằng phép biến đổi Hough	61

LỜI MỞ ĐẦU

Xử lý ảnh là một lĩnh vực mang tính khoa học và công nghệ. Nó là một trong những chuyên ngành quan trọng của công nghệ thông tin hiện nay được áp dụng trong những lĩnh vực khác nhau như: y học, toán học, tìm kiếm tội phạm và nhiều lĩnh vực khoa học khác....

Các phương pháp xử lý ảnh bắt đầu từ các ứng dụng chính là nâng cao chất lượng ảnh và phân tích ảnh. Các phương pháp xử lý ảnh trong tìm kiếm hình dạng trong hình ảnh máy tính để có thể tự động nhận dạng khuôn mặt đang được áp dụng phổ biến trong quốc phòng an ninh. Để có thể đối sánh được hình dạng thì phải trích trợn được các đặc trưng bất biến của hình dạng. Chính vì vậy mà nhóm em lựa chọn đề tài “ Phát hiện đường thẳng và hình tròn bằng phép biến đổi Hough (Hough Transform)” để tìm hiểu các phương pháp trích chọn đặc trưng bất biến của ảnh, các đường thẳng, hình tròn Hough.

Nội dung gồm các 3 chương:

Chương 1: Giới thiệu ngôn ngữ Python

Chương 2: Tổng quan về xử lý ảnh

Chương 3: Giới thiệu về thư viện OpenCV

Chương 4: Viết ứng dụng phát hiện đoạn thẳng và hình tròn bằng phép biến đổi Hough

CHƯƠNG 1: TÌM HIỂU NGÔN NGỮ LẬP TRÌNH PYTHON

1.1. Giới thiệu ngôn ngữ Python

Python là một ngôn ngữ lập trình thông dịch do Guido van Rossum tạo ra năm 1990. Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Theo đánh giá của Eric S. Raymond, Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu, như nhận định của chính Guido van Rossum trong một bài phỏng vấn ông.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, nó đã “bành trướng” sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

1.1.1. Lịch sử phát triển

Python được hình thành vào cuối những năm 1980, và việc thực hiện nó vào tháng 12 năm 1989 bởi Guido van Rossum tại Centrum Wiskunde & Informatica (CWI) ở Hà Lan như là một kế thừa cho ngôn ngữ ABC (tự lấy cảm hứng từ SETL) có khả năng xử lý ngoại lệ và giao tiếp với Hệ điều hành Amoeba. Van Rossum là tác giả chính của Python, và vai trò trung tâm của ông trong việc quyết định hướng phát triển của Python.



Hình 1. 1: Guido van Russom

Sự phát triển Python đến nay có thể chia làm các giai đoạn:

Python 1: bao gồm các bản phát hành 1.x. Giai đoạn này, kéo dài từ đầu đến cuối thập niên 1990. Từ năm 1990 đến 1995, Guido làm việc tại CWI (Centrum voor Wiskunde en Informatica - Trung tâm Toán-Tin học tại Amsterdam, Hà Lan). Vì vậy, các phiên bản Python đầu tiên đều do CWI phát hành. Phiên bản cuối cùng phát hành tại CWI là 1.2.

Vào năm 1995, Guido chuyển sang CNRI (Corporation for National Research Initiatives) ở Reston, Virginia. Tại đây, ông phát hành một số phiên bản khác. Python 1.6 là phiên bản cuối cùng phát hành tại CNRI.

Sau bản phát hành 1.6, Guido rời bỏ CNRI để làm việc với các lập trình viên chuyên viết phần mềm thương mại. Tại đây, ông có ý tưởng sử dụng Python với các phần mềm tuân theo chuẩn GPL. Sau đó, CNRI và FSF (Free Software Foundation - Tổ chức phần mềm tự do) đã cùng nhau hợp tác để làm bản quyền Python phù hợp với GPL. Cùng năm đó, Guido được nhận Giải thưởng FSF vì Sự phát triển Phần mềm tự do (Award for the Advancement of Free Software).

Phiên bản 1.6.1 ra đời sau đó là phiên bản đầu tiên tuân theo bản quyền GPL. Tuy nhiên, bản này hoàn toàn giống bản 1.6, trừ một số sửa lỗi cần thiết.

Python 2: vào năm 2000, Guido và nhóm phát triển Python dời đến BeOpen.com và thành lập BeOpen PythonLabs team. Phiên bản Python 2.0 được phát hành tại đây. Sau khi phát hành Python 2.0, Guido và các thành viên PythonLabs gia nhập Digital Creations.

Python 2.1 ra đời kế thừa từ Python 1.6.1 và Python 2.0. Bản quyền của phiên bản này được đổi thành Python Software Foundation License. Từ thời điểm này trở đi, Python thuộc sở hữu của Python Software Foundation (PSF), một tổ chức phi lợi nhuận được thành lập theo mẫu Apache Software Foundation.

Python 3, còn gọi là Python 3000 hoặc Py3K: Dòng 3.x sẽ không hoàn toàn tương thích với dòng 2.x, tuy vậy có công cụ hỗ trợ chuyển đổi từ các phiên bản 2.x sang 3.x. Nguyên tắc chủ đạo để phát triển Python 3.x là "bỏ cách làm việc cũ nhằm hạn chế trùng lặp về mặt chức năng của Python". Trong PEP (Python Enhancement Proposal) có mô tả chi tiết các thay đổi trong Python. Các đặc điểm mới của Python 3.0 sẽ được trình bày phần cuối bài này.

1.1.2. Phiên bản của Python

- *CPython*

Đây là phiên bản đầu tiên và được duy trì lâu nhất của Python, được viết trong C. Những đặc điểm của ngôn ngữ mới xuất hiện đầu tiên từ đây

- *Jython*

Là phiên bản Python trong môi trường Java. Bản này có thể được sử dụng như là một ngôn ngữ script cho những ứng dụng Java. Nó cũng thường được sử dụng để tạo ra những tests thử nghiệm cho thư viện Java

- *Python for .NET*

Phiên bản này thật ra sử dụng phiên bản Cpython, nhưng nó là một ứng dụng .NET được quản lý, và tạo ra thư viện .NET sẵn dùng. Bản này được xây dựng bởi Brian Lloyd.

- *IronPython*

Là một bản Python tiếp theo của .NET, không giống như Python.NET đây là một bản Python hoàn chỉnh, tạo ra IL, và biên dịch mã Python trực tiếp vào một tập hợp .NET.

- *PyPy*

PyPy được viết trên Python, thậm chí cả việc thông dịch từng byte mã cũng được viết trên Python. Nó sử dụng Cpython như một trình thông dịch cơ sở. Một trong những mục đích của dự án cho ra đời phiên bản này là để khuyến khích sự thử nghiệm bản thân ngôn ngữ này làm cho nó dễ dàng hơn để sửa đổi thông dịch (bởi vì nó được viết trên Python).

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

- *Division operator*

Thay đổi này đặc biệt nguy hiểm nếu bạn đang thực thi code Python 3 trong Python 2, vì thay đổi trong hành vi phân chia số nguyên thường có thể không được chú ý (nó không làm tăng cú pháp SyntaxError).

```
print 7 / 5
print -7 / 5
Output in Python 2.x:
1
-2
Output in Python 3.x:
1.4
-1.4
```

- *Print function*

Đây là một trong những sự thay đổi được biết đến nhiều nhất từ bản Python 2.x lên Python 3.x. Python 2.x không có vấn đề với các lệnh bỏ sung, nhưng ngược lại, Python 3.x sẽ tăng cú pháp SyntaxError nếu chúng ta gọi hàm in mà không có dấu ngoặc đơn.

```
print 'Hello, Geeks'      # Python 3.x doesn't support
print('Hope You like these facts')
Output in Python 2.x:
Hello, Geeks
Hope You like these facts

Output in Python 3.x:
File "a.py", line 1
    print 'Hello, Geeks'
                        ^
SyntaxError: invalid syntax
```

- *Unicode*

Trong Python 2.x, kiểu mặc định của String là ASCII, nhưng ở Python 3.x kiểu mặc định của String là Unicode và 2 lớp byte: byte và bytearray.

```
print(type('default string '))
print(type(u'string with b '))
Output in Python 2.x (Unicode and str are different):
<type 'str'>
<type 'unicode'>

Output in Python 3.x (Unicode and str are same):
<class 'str'>
<class 'str'>
```

- *Xrange*

Việc sử dụng xrange () rất phổ biến trong Python 2.x để tạo một đối tượng có thể lặp lại, ví dụ: trong vòng lặp for hoặc list / set-dictionary-comprehension. xrange-iterable không phải hoàn toàn có nghĩa là bạn có thể lặp lại nó vô hạn.

Trong Python 3, range () đã được thực hiện giống như hàm xrange () sao cho hàm xrange () chuyên dụng không còn tồn tại nữa (xrange () tăng một NameError trong Python 3).

```
for x in xrange(1, 5):
    print(x),

for x in range(1, 5):
    print(x),
Output in Python 2.x:
1 2 3 4 1 2 3 4
Output in Python 3.x:
```

```
NameError: name 'xrange' is not defined
```

- *Error Handling*

Đây là một thay đổi nhỏ trên phiên bản 3.x, từ khoá `as` đã trở thành bắt buộc Kiểm tra không có từ khoá `as` trong 2 phiên bản Python

```
try:
    trying_to_check_error
except NameError, err:
    print err, 'Error Caused'    # Would not work in Python 3.x
Output in Python 2.x:
name 'trying_to_check_error' is not defined Error Caused

Output in Python 3.x :
File "a.py", line 3
    except NameError, err:
                   ^
SyntaxError: invalid syntax
```

Kiểm tra khi có từ khoá `as` trong 2 phiên bản Python

```
try:
    trying_to_check_error
except NameError as err: # 'as' is needed in Python 3.x
    print (err, 'Error Caused')
Output in Python 2.x:
(NameError("name 'trying_to_check_error' is not defined"),, 'Error Caused')

Output in Python 3.x:
name 'trying_to_check_error' is not defined Error Caused
```

- *The next() function and next() method*

`Next()` hay `(.next ())` là một hàm thường được sử dụng (method), đây là một thay đổi cú pháp khác (hay đúng hơn là thay đổi trong thực thi) sử dụng tốt trong python 2 nhưng bị loại bỏ trong python3

Vd: python 2.x:

```
my_generator = (letter for letter in 'abcdefg')

next(my_generator)
my_generator.next()
-----
result:
'a'
'b'
```

python3.x:

```
my_generator = (letter for letter in 'abcdefg')
next(my_generator)

-----
'a'

my_generator.next()

-----
AttributeError                                Traceback (most recent call last)

<ipython-input-14-125f388bb61b> in <module>()
----> 1 my_generator.next()

AttributeError: 'generator' object has no attribute 'next'
```

- *Các biến vòng lặp và rò rỉ namespace chung*

Trong Python 3.x cho các biến vòng lặp không bị rò rỉ vào không gian tên chung nữa List comprehensions không còn hỗ trợ mẫu cú pháp [... for var in item1, item2, ...]. Sử dụng [... for var in (item1, item2, ...)] thay thế. Cũng lưu ý rằng việc hiểu danh sách có ngữ nghĩa khác nhau: chúng gần với cú pháp cho biểu thức máy hiểu bên trong một hàm tạo danh sách (và đặc biệt là các biến điều khiển vòng lặp không còn bị rò rỉ ra ngoài phạm vi. ”

Vd: Python 2.x

```
i = 1
print 'before: i =', i
print 'comprehension: ', [i for i in range(5)]

print 'after: i =', i
-----
before: i = 1
comprehension:  [0, 1, 2, 3, 4]
after: i = 4
```

Python 3.x:

```
i = 1
print('before: i =', i)

print('comprehension:', [i for i in range(5)])
print('after: i =', i)
-----
before: i = 1
comprehension: [0, 1, 2, 3, 4]
after: i = 1
```

- So sánh khác loại

Ở python 2 ,có thể so sánh list với string,... nhưng ở python 3 thì không

Vd: python 2.x

```
print "[1, 2] > 'foo' = ", [1, 2] > 'foo'
print "(1, 2) > 'foo' = ", (1, 2) > 'foo'
print "[1, 2] > (1, 2) = ", [1, 2] > (1, 2)
-----
[1, 2] > 'foo' = False
(1, 2) > 'foo' = True
[1, 2] > (1, 2) = False
```

Python 3.x

```
print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))
-----
TypeError                                Traceback (most recent call last)

<ipython-input-16-a9031729f4a0> in <module>()
      1 print('Python', python_version())
----> 2 print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
      3 print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
      4 print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))
      TypeError: unorderable types: list() > str()
```

- Phân tích cú pháp đầu vào của người dùng thông qua input()

Hàm input () được cố định trong Python 3 để nó luôn lưu trữ đầu vào của người dùng làm các đối tượng str. Để tránh hành vi nguy hiểm trong Python 2 để đọc trong các loại khác so với chuỗi, chúng ta phải sử dụng raw_input () để thay thế.

Vd: Python 2.x

```
Python 2.7.6
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

>>> my_input = input('enter a number: ')

enter a number: 123

>>> type(my_input)
<type 'int'>

>>> my_input = raw_input('enter a number: ')
enter a number: 123

>>> type(my_input)
```

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

Python 3.x

```
Python 3.4.1
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

>>> my_input = input('enter a number: ')

enter a number: 123

>>> type(my_input)
<class 'str'>
```

Trả về các đối tượng có thể lặp lại thay vì danh sách

- *Một số hàm và phương thức trả về các đối tượng có thể lặp lại trong Python 3 thay vì các danh sách trong Python 2*

Vd: Python 2.x

```
print range(3)
print type(range(3))
-----
[0, 1, 2]
<type 'list'>
```

Python 3

```
print(range(3))
print(type(range(3)))
print(list(range(3)))
-----
range(0, 3)
<class 'range'>
[0, 1, 2]
```

Một số hàm và phương thức thường được sử dụng không trả về danh sách nữa trong Python 3:

```
zip()
map()
filter()
dictionary's .keys() method
```

```
dictionary's .values() method
```

```
dictionary's .items() method
```

1.1.4. Đặc điểm của Python

- Dễ học, dễ đọc

Python được thiết kế để trở thành một ngôn ngữ dễ học, mã nguồn dễ đọc, bố cục trực quan, dễ hiểu, thể hiện qua các điểm sau:

- Từ khóa

- Python tăng cường sử dụng từ khóa tiếng Anh, hạn chế các kí hiệu và cấu trúc cú pháp so với các ngôn ngữ khác.
- Python là một ngôn ngữ phân biệt kiểu chữ HOA, chữ thường.
- Như C/C++, các từ khóa của Python đều ở dạng chữ thường.

- Khối lệnh

Trong các ngôn ngữ khác, khối lệnh thường được đánh dấu bằng cặp kí hiệu hoặc từ khóa. Ví dụ, trong C/C++, cặp ngoặc nhọn { } được dùng để bao bọc một khối lệnh. Python, trái lại, có một cách rất đặc biệt để tạo khối lệnh, đó là thụt các câu lệnh trong khối vào sâu hơn (về bên phải) so với các câu lệnh của khối lệnh cha chứa nó. Ví dụ, giả sử có đoạn mã sau trong C/C++:

```
#include <math.h>
//...
delta = b * b - 4 * a * c;
if (delta > 0) {
// Khoi lenh moi bat dau tu ki tu { den }
x1 = (- b + sqrt(delta)) / (2 * a);
x2 = (- b - sqrt(delta)) / (2 * a);
printf("Phuong trinh co hai nghiem phan biet:\n");
printf("x1 = %f; x2 = %f", x1, x2);
}
```

Đoạn mã trên có thể được viết lại bằng Python như sau:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
```

```
import math
#...
delta = b * b - 4 * a * c
if delta > 0:
    # Khởi lệnh mới, thụt vào đầu dòng
    x1 = (- b + math.sqrt(delta)) / (2 * a)
    x2 = (- b - math.sqrt(delta)) / (2 * a)
    print "Phương trình có hai nghiệm phân biệt:"
print "x1 = ", x1, "; ", "x2 = ", x2
```

Ta có thể sử dụng dấu tab hoặc khoảng trống để thụt các câu lệnh vào.

- Các bản hiện thực

Python được viết từ những ngôn ngữ khác, tạo ra những bản hiện thực khác nhau. Bản hiện thực Python chính, còn gọi là CPython, được viết bằng C, và được phân phối kèm một thư viện chuẩn lớn được viết hỗn hợp bằng C và Python. CPython có thể chạy trên nhiều nền và khả chuyển trên nhiều nền khác. Dưới đây là các nền trên đó, CPython có thể chạy.

- Các hệ điều hành họ Unix: AIX, Darwin, FreeBSD, Mac OS X, NetBSD, Linux, OpenBSD, Solaris,...
- Các hệ điều hành dành cho máy desktop: Amiga, AROS, BeOS, Mac OS 9, Microsoft Windows, OS/2, RISC OS.
- Các hệ thống nhúng và các hệ đặc biệt: GP2X, Máy ảo Java, Nokia 770 Internet Tablet, Palm OS, PlayStation 2, PlayStation Portable, Psion, QNX, Sharp Zaurus, Symbian OS, Windows CE/Pocket PC, Xbox/XBMC, VxWorks.
- Các hệ máy tính lớn và các hệ khác: AS/400, OS/390, Plan 9 from Bell Labs, VMS, z/OS.

Ngoài CPython, còn có hai hiện thực Python khác: Jython cho môi trường Java và IronPython cho môi trường .NET và Mono.

- Khả năng mở rộng

Python có thể được mở rộng: nếu ta biết sử dụng C, ta có thể dễ dàng viết và tích hợp vào Python nhiều hàm tùy theo nhu cầu. Các hàm này sẽ trở thành hàm xây dựng sẵn (built-in) của Python. Ta cũng có thể mở rộng chức năng của trình thông dịch, hoặc liên kết các chương trình Python với các thư viện chỉ ở dạng nhị phân (như các thư viện đồ họa do nhà sản xuất thiết bị cung cấp). Hơn thế nữa, ta cũng có thể liên kết trình thông dịch của Python với các ứng dụng viết từ C và sử dụng nó như là một mở rộng hoặc một ngôn ngữ dòng lệnh phụ trợ cho ứng dụng đó.

- Trình thông dịch

Python là một ngôn ngữ lập trình dạng thông dịch, do đó có ưu điểm tiết kiệm thời gian phát triển ứng dụng vì không cần phải thực hiện biên dịch và liên kết. Trình thông dịch có thể được sử dụng để chạy file script, hoặc cũng có thể được sử dụng theo cách tương tác. Ở chế độ tương tác, trình thông dịch Python tương tự shell của các hệ điều hành họ Unix, tại đó, ta có thể nhập vào từng biểu thức rồi gõ Enter, và kết quả thực thi sẽ được hiển thị ngay lập tức. Đặc điểm này rất hữu ích cho người mới học, giúp họ nghiên cứu tính năng của ngôn ngữ; hoặc để các lập trình viên chạy thử mã lệnh trong suốt quá trình phát triển phần mềm. Ngoài ra, cũng có thể tận dụng đặc điểm này để thực hiện các phép tính như với máy tính bỏ túi.

- Lệnh và cấu trúc điều khiển

Mỗi câu lệnh trong Python nằm trên một dòng mã nguồn. Ta không cần phải kết thúc câu lệnh bằng bất kỳ ký tự gì. Cũng như các ngôn ngữ khác, Python cũng có các cấu trúc điều khiển. Chúng bao gồm:

Cấu trúc rẽ nhánh: cấu trúc if (có thể sử dụng thêm elif hoặc else), dùng để thực thi có điều kiện một khối mã cụ thể.

Cấu trúc lặp, bao gồm:

- Lệnh while: chạy một khối mã cụ thể cho đến khi điều kiện lặp có giá trị false.
- Vòng lặp for: lặp qua từng phần tử của một dãy, mỗi phần tử sẽ được đưa vào biến cục bộ để sử dụng với khối mã trong vòng lặp.

Python cũng có từ khóa class dùng để khai báo lớp (sử dụng trong lập trình hướng đối tượng) và lệnh def dùng để định nghĩa hàm.

- Hệ thống kiểu dữ liệu

Python sử dụng hệ thống kiểu duck typing, còn gọi là latent typing (tự động xác định kiểu). Có nghĩa là, Python không kiểm tra các ràng buộc về kiểu dữ liệu tại thời điểm dịch, mà là tại thời điểm thực thi. Khi thực thi, nếu một thao tác trên một đối tượng bị thất bại, thì có nghĩa là đối tượng đó không sử dụng một kiểu thích hợp.

Python cũng là một ngôn ngữ định kiểu mạnh. Nó cấm mọi thao tác không hợp lệ, ví dụ cộng một con số vào chuỗi ký tự.

Sử dụng Python, ta không cần phải khai báo biến. Biến được xem là đã khai báo nếu nó được gán một giá trị lần đầu tiên. Căn cứ vào mỗi lần gán, Python sẽ tự động xác định kiểu dữ liệu của biến. Python có một số kiểu dữ liệu thông dụng sau:

- int, long: số nguyên (trong phiên bản 3.x long được nhập vào trong kiểu int). Độ dài của kiểu số nguyên là tùy ý, chỉ bị giới hạn bởi bộ nhớ máy tính.
- float: số thực
- complex: số phức, chẳng hạn $5+4j$
- list: dãy trong đó các phần tử của nó có thể được thay đổi, chẳng hạn `[8, 2, 'b', -1.5]`. Kiểu dãy khác với kiểu mảng (array) thường gặp trong các ngôn ngữ lập trình ở chỗ các phần tử của dãy không nhất thiết có kiểu giống nhau. Ngoài ra phần tử của dãy còn có thể là một dãy khác.
- tuple: dãy trong đó các phần tử của nó không thể thay đổi.
- str: chuỗi kí tự. Từng kí tự trong chuỗi không thể thay đổi. Chuỗi kí tự được đặt trong dấu nháy đơn, hoặc nháy kép.
- dict: từ điển, còn gọi là "hashtable": là một cặp các dữ liệu được gắn theo kiểu {từ khóa: giá trị}, trong đó các từ khóa trong một từ điển nhất thiết phải khác nhau. Chẳng hạn `{1: "Python", 2: "Pascal"}`
- set: một tập không xếp theo thứ tự, ở đó, mỗi phần tử chỉ xuất hiện một lần.

Ngoài ra, Python còn có nhiều kiểu dữ liệu khác. Xem thêm trong phần "Các kiểu dữ liệu" bên dưới.

- Module

Python cho phép chia chương trình thành các module để có thể sử dụng lại trong các chương trình khác. Nó cũng cung cấp sẵn một tập hợp các modules chuẩn mà lập trình viên có thể sử dụng lại trong chương trình của họ. Các module này cung cấp nhiều chức năng hữu ích, như các hàm truy xuất tập tin, các lời gọi hệ thống, trợ giúp lập trình mạng (socket),...

- Đa năng

Python là một ngôn ngữ lập trình đơn giản nhưng rất hiệu quả.

- So với Unix shell, Python hỗ trợ các chương trình lớn hơn và cung cấp nhiều cấu trúc hơn.
- So với C, Python cung cấp nhiều cơ chế kiểm tra lỗi hơn. Nó cũng có sẵn nhiều kiểu dữ liệu cấp cao, ví dụ như các mảng (array) linh hoạt và từ điển (dictionary) mà ta sẽ phải mất nhiều thời gian nếu viết bằng C.

Python là một ngôn ngữ lập trình cấp cao có thể đáp ứng phần lớn yêu cầu của lập trình viên:

- Python thích hợp với các chương trình lớn hơn cả AWK và Perl.
- Python được sử dụng để lập trình Web. Nó có thể được sử dụng như một ngôn ngữ kịch bản.

- Python được thiết kế để có thể nhúng và phục vụ như một ngôn ngữ kịch bản để tùy biến và mở rộng các ứng dụng lớn hơn.
- Python được tích hợp sẵn nhiều công cụ và có một thư viện chuẩn phong phú, Python cho phép người dùng dễ dàng tạo ra các dịch vụ Web, sử dụng các thành phần COM hay CORBA, hỗ trợ các loại định dạng dữ liệu Internet như email, HTML, XML và các ngôn ngữ đánh dấu khác. Python cũng được cung cấp các thư viện xử lý các giao thức Internet thông dụng như HTTP, FTP,...
- Python có khả năng giao tiếp đến hầu hết các loại cơ sở dữ liệu, có khả năng xử lý văn bản, tài liệu hiệu quả, và có thể làm việc tốt với các công nghệ Web khác.
- Python đặc biệt hiệu quả trong lập trình tính toán khoa học nhờ các công cụ Python Imaging Library, pyVTK, MayaVi 3D Visualization Toolkits, Numeric Python, ScientificPython,...
- Python có thể được sử dụng để phát triển các ứng dụng desktop. Lập trình viên có thể dùng wxPython, PyQt, PyGtk để phát triển các ứng dụng giao diện đồ họa (GUI) chất lượng cao. Python còn hỗ trợ các nền tảng phát triển phần mềm khác như MFC, Carbon, Delphi, X11, Motif, Tk, Fox, FLTK, ...
- Python cũng có sẵn một unit testing framework để tạo ra các bộ test (test suites).

- Multiple paradigms (đa biến hóa)

Python là một ngôn ngữ đa biến hóa (multiple paradigms). Có nghĩa là, thay vì ép buộc mọi người phải sử dụng duy nhất một phương pháp lập trình, Python lại cho phép sử dụng nhiều phương pháp lập trình khác nhau: hướng đối tượng, có cấu trúc, chức năng, hoặc chỉ hướng đến một khía cạnh. Python kiểu kiểu động và sử dụng bộ thu gom rác để quản lý bộ nhớ. Một đặc điểm quan trọng nữa của Python là giải pháp tên động, kết nối tên biến và tên phương thức lại với nhau trong suốt thực thi của chương trình.

- Sự tương đương giữa true và một giá trị khác 0

Cũng như C/C++, bất kỳ một giá trị khác 0 nào cũng tương đương với true và ngược lại, một giá trị 0 tương đương với false. Như vậy:

```
if a != 0:
```

tương đương với:

```
if a:
```

- Cú pháp

Sau đây là cú pháp cơ bản nhất của ngôn ngữ Python:

+ Toán tử

```
+ - * / // (chia làm tròn) % (phần dư) ** (lũy thừa)
~ (not) & (and) | (or) ^ (xor)
<< (left shift) >> (right shift)
== (bằng) <= >= != (khác)
```

Python sử dụng kí pháp trung tố thường gặp trong các ngôn ngữ lập trình khác.

+ Các kiểu dữ liệu

🔗 Kiểu số

```
1234585396326 (số nguyên dài vô hạn) -86.12 7.84E-04
2j 3 + 8j (số phức)
```

🔗 Kiểu chuỗi (string)

```
"Hello" "It's me" '"OK"-he replied'
```

🔗 Kiểu bộ (tuple)

```
(1, 2.0, 3) (1,) ("Hello",1,())
```

🔗 Kiểu danh sách (list)

```
[4.8, -6] ['a', 'b']
```

🔗 Kiểu từ điển (dictionary)

```
{"Vietnam":"Hanoi", "Netherlands":"Amsterdam", "France":"Paris"}
```

+ Chú thích

```
# dòng chú thích
```

+ Lệnh gán

```
tên biến = biểu thức
x = 23.8
y = -x ** 2
z1 = z2 = x + y
loiChao = "Hello!"

i += 1 # tăng biến i thêm 1 đơn vị
```

+ In giá trị

```
print biểu thức
print (7 + 8) / 2.0
```

```
print (2 + 3j) * (4 - 6j)
```

+ *Nội suy chuỗi (string interpolation)*

```
print "Hello %s" %("world!")  
print "i = %d" %i  
print "a = %.2f and b = %.3f" %(a,b)
```

+ *Cấu trúc rẽ nhánh*

🔗 Dạng 1:

```
if biểu_thức_điều_kiện:  
    # lệnh ...
```

🔗 Dạng 2:

```
if biểu_thức_điều_kiện:  
    # lệnh ...  
else:  
    # lệnh ...
```

🔗 Dạng 3:

```
if biểu_thức_điều_kiện_1:  
    # lệnh ... (được thực hiện nếu biểu_thức_điều_kiện_1 là đúng/true)  
elif biểu_thức_điều_kiện_2:  
    # lệnh ... (được thực hiện nếu biểu_thức_điều_kiện_1 là sai/false,  
    nhưng biểu_thức_điều_kiện_2 là đúng/true)  
else:  
    # lệnh ... (được thực hiện nếu tất cả các biểu thức điều kiện đi kèm if  
    và elif đều sai)
```

+ *Cấu trúc lặp*

```
while biểu_thức_đúng:  
    # lệnh ...  
    for phần_tử in dãy:  
        # lệnh ...  
        L = ["Ha Noi", "Hai Phong", "TP Ho Chi Minh"]  
        for thanhPho in L:  
            print thanhPho  
  
for i in range(10):  
    print i
```

+ *Hàm*

```
def tên_hàm (tham_biến_1, tham_biến_2, tham_biến_n):  
    # lệnh ...  
    return giá_trị_hàm  
def binhPhuong(x):
```

```
return x*x
```

+ Hàm với tham số mặc định:

```
def luyThua(x, n=2):  
    """Lũy thừa với số mũ mặc định là 2"""  
    return x**n  
  
print luyThua(3) # 9  
print luyThua(2,3) # 8
```

+ Lớp

```
class Tên_Lớp_1:  
    # ...  
  
class Tên_Lớp_2(Tên_Lớp_1):  
    """Lớp 2 kế thừa lớp 1"""  
    x = 3 # biến thành viên của lớp  
    #  
    def phương_thức(self, tham_biến):  
    # ...  
  
# khởi tạo  
a = Tên_Lớp_2()  
print a.x  
print a.phương_thức(m) # m là giá trị gán cho tham biến  
List Comprehension
```

+ List Comprehension

Là dạng cú pháp đặc biệt (syntactic sugar) (mới có từ Python 2.x) cho phép thao tác trên toàn bộ dãy (list) mà không cần viết rõ vòng lặp. Chẳng hạn y là một dãy mà mỗi phần tử của nó bằng bình phương của từng phần tử trong dãy x:

```
y = [xi**2 for xi in x]
```

+ Xử lý lỗi

```
try:  
    câu_lệnh  
except Loại_Lỗi:  
    thông_báo_lỗi
```

+ Tốc độ thực hiện

Là một ngôn ngữ thông dịch, Python có tốc độ thực hiện chậm hơn nhiều lần so với các ngôn ngữ biên dịch như Fortran, C, v.v... Trong số các ngôn ngữ thông dịch, Python được đánh giá nhanh hơn Ruby và Tcl, nhưng chậm hơn Lua.

+ Các đặc điểm mới trong Python 3.x

Nội dung phần này được trích từ tài liệu của Guido van Rossum. Phần này không liệt kê đầy đủ tất cả các đặc điểm; chi tiết xin xem tài liệu nói trên.

+ Một số thay đổi cần lưu ý nhất

Lệnh print trở thành hàm print. Theo đó sau print() ta cần nhớ gõ vào cặp ngoặc ():

```
print("Hello")  
print(2+3)
```

Trả lại kết quả không còn là list trong một số trường hợp.

- dict.keys(), dict.items(), dict.values() kết quả cho ra các "view" thay vì list.
- map và filter trả lại các iterator.
- range bây giờ có tác dụng như xrange, và không trả lại list.

✂ So sánh

Không còn hàm cmp, và cmp(a, b) có thể được thay bằng (a > b) - (a < b)

✂ Số nguyên

- Kiểu long được đổi tên thành int.
- 1/2 cho ta kết quả là số thực chứ không phải số nguyên.
- Không còn hằng số sys.maxint
- Kiểu bát phân được kí hiệu bằng 0o thay vì 0, chẳng hạn 0o26.

Phân biệt văn bản - dữ liệu nhị phân thay vì Unicode - chuỗi 8-bit

- Tất cả chuỗi văn bản đều dưới dạng Unicode, nhưng chuỗi Unicode mã hóa lại là dạng dữ liệu nhị phân. Dạng mặc định là UTF-8.
- Không thể viết u"a string" để biểu diễn chuỗi như trong các phiên bản 2.x

+ Các thay đổi về cú pháp

✂ Cú pháp mới

- Các tham biến chỉ chấp nhận keyword: Các tham biến phía sau *args phải được gọi theo dạng keyword.
- Từ khóa mới nonlocal. Muốn khai báo một biến x với có phạm vi ảnh hưởng rộng hơn, nhưng chưa đến mức toàn cục, ta dùng nonlocal x.
- Gán giá trị vào các phần tử tuple một cách thông minh, chẳng hạn có thể viết (a, *rest, b) = range(5) để có được a = 0; b = [1,2,3]; c = 4.

- Dictionary comprehension, chẳng hạn {k: v for k, v in stuff} thay vì dict(stuff).
- Kiểu nhị phân, chẳng hạn b110001.

🔪 Cú pháp được thay đổi

- raise [biểu_thức [from biểu_thức]]
- except lệnh as biến
- Sử dụng metaclass trong đối tượng:

```
class C(metaclass=M):
    pass
```

Cách dùng biến `__metaclass__` không còn được hỗ trợ.

+ *Cú pháp bị loại bỏ*

- Không còn dấu ``, thay vì đó, dùng repr.
- Không còn so sánh <> (dùng !=).

Không còn các lớp kiểu classic.

1.2. Hướng dẫn cài đặt Anaconda

1.2.1. Giới thiệu Anaconda

Anaconda là nền tảng mã nguồn mở về Khoa học dữ liệu trên Python thông dụng nhất hiện nay. Anaconda hướng đến việc quản lí các package một cách đơn giản, phù hợp với mọi người. Hệ thống quản lí package của Anaconda là Conda. Anaconda Với hơn 11 triệu người dùng, Anaconda là cách nhanh nhất và dễ nhất để học Khoa học dữ liệu với Python hoặc R trên Windows, Linux và Mac OS X. Lợi ích của Anaconda:

- Dễ dàng tải 1500+ packages về Python/R cho data science
- Quản lý thư viện, môi trường và dependency giữa các thư viện dễ dàng
- Dễ dàng phát triển mô hình machine learning và deep learning với scikit-learn, tensorflow, keras
- Xử lý dữ liệu tốc độ cao với numpy, pandas
- Hiện thị kết quả với Matplotlib, Bokeh

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 quang trọng hơn là nó được cài đặt khi bạn cài đặt Anaconda.

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

[Download](#)

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with *scikit-learn*, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews



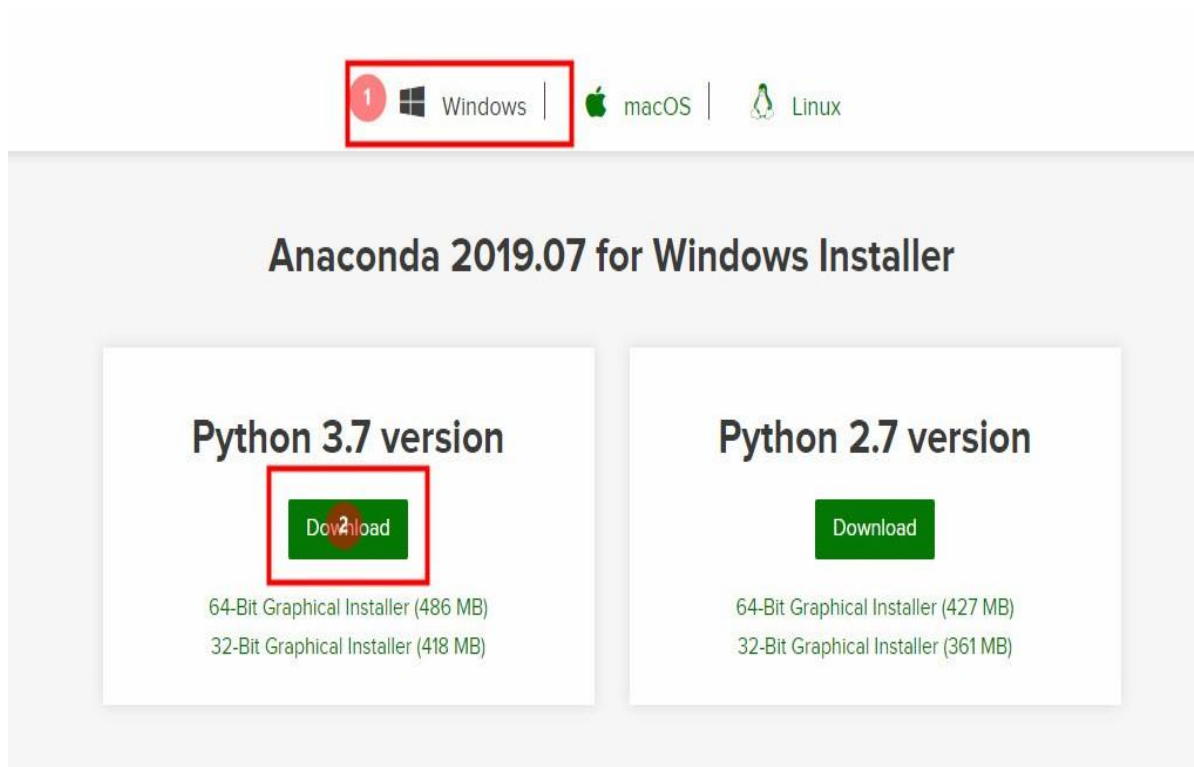
Hình 1.2: Giao diện trang chủ Anaconda

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

1.2.2. Download Anaconda và hướng dẫn cài đặt trên HĐH Window

- ✎ Download Anaconda (python3) cho HĐH Window về tại <https://www.anaconda.com/distribution/>
- ✎ Hiện tại phiên bản Python mà ta sử dụng là python3.7 bản phân phối Anaconda 2019.07



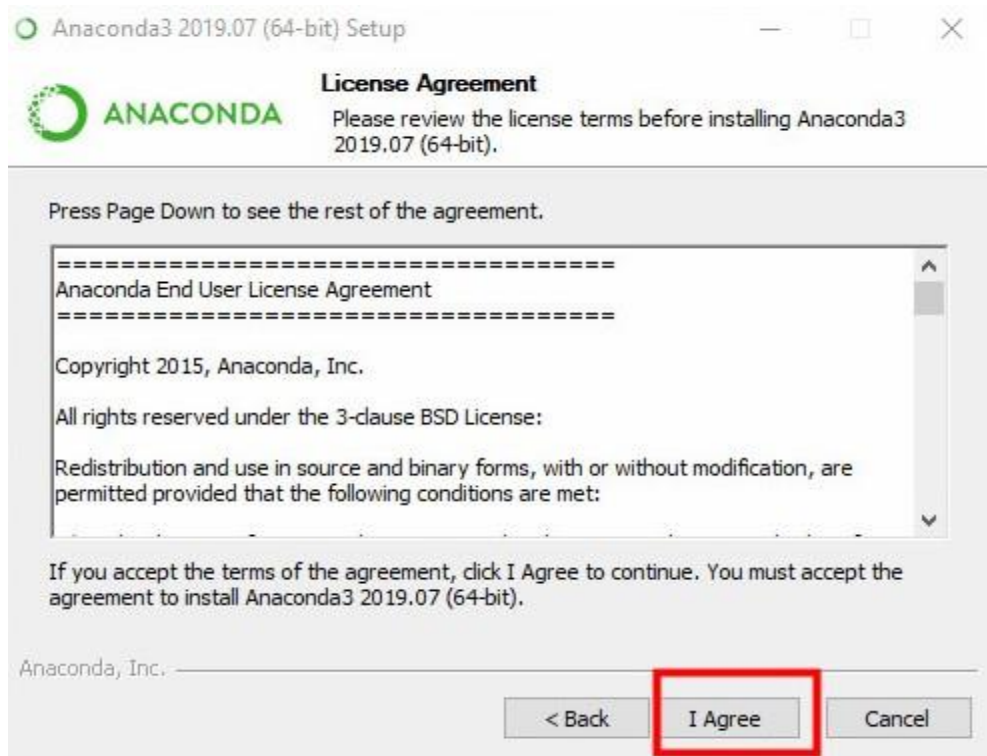
Hình 1.3: Trang tải Anaconda

Sau khi tải về xong bạn chạy file “[Anaconda3-2019.07-Windows-x86_64.exe](#)”



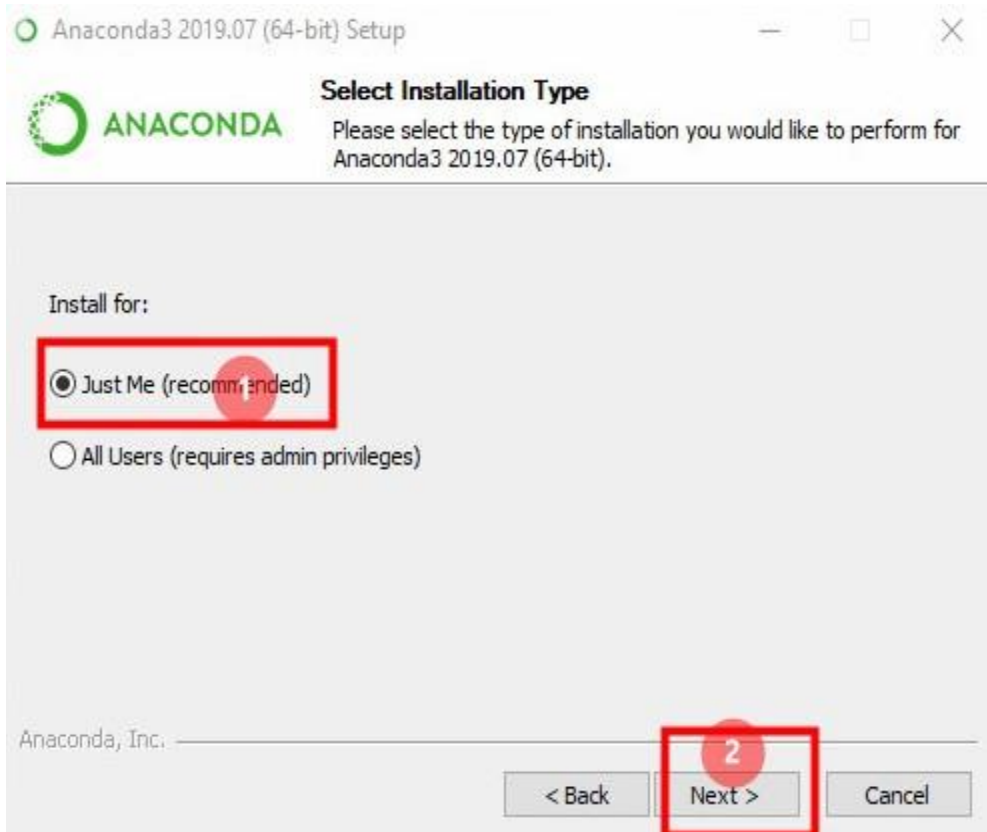
Hình 1. 4: Chạy chương trình vừa tải về

- Click Next



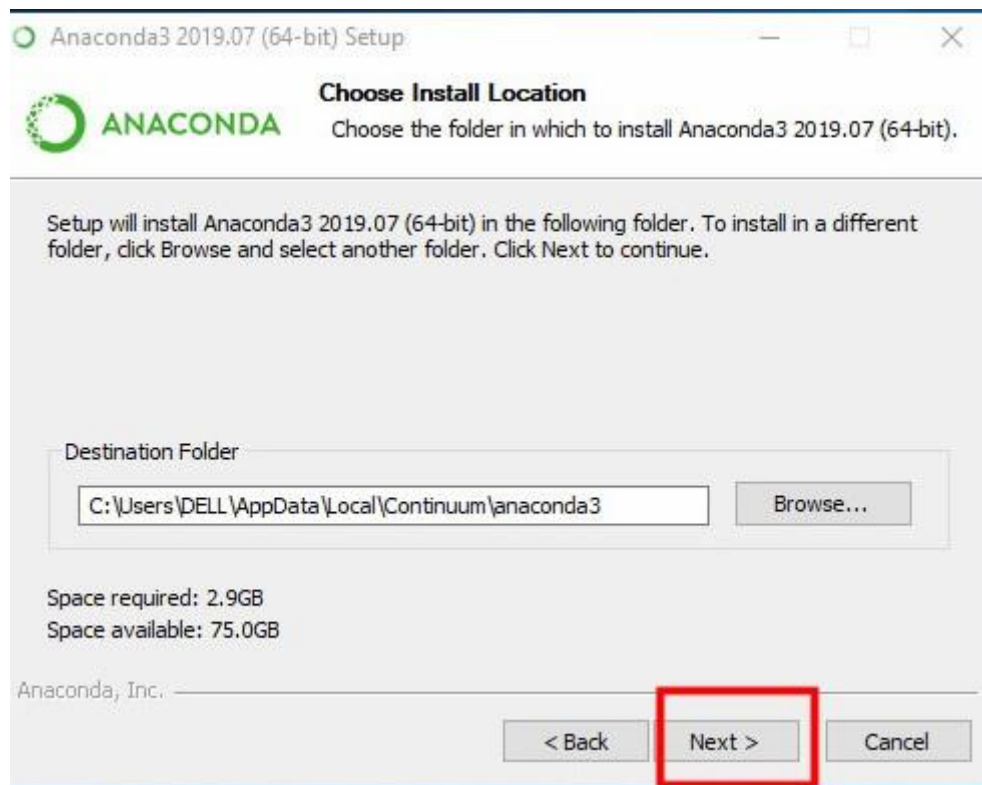
Hình 1. 5: License Agreement

- Click **I Agree**



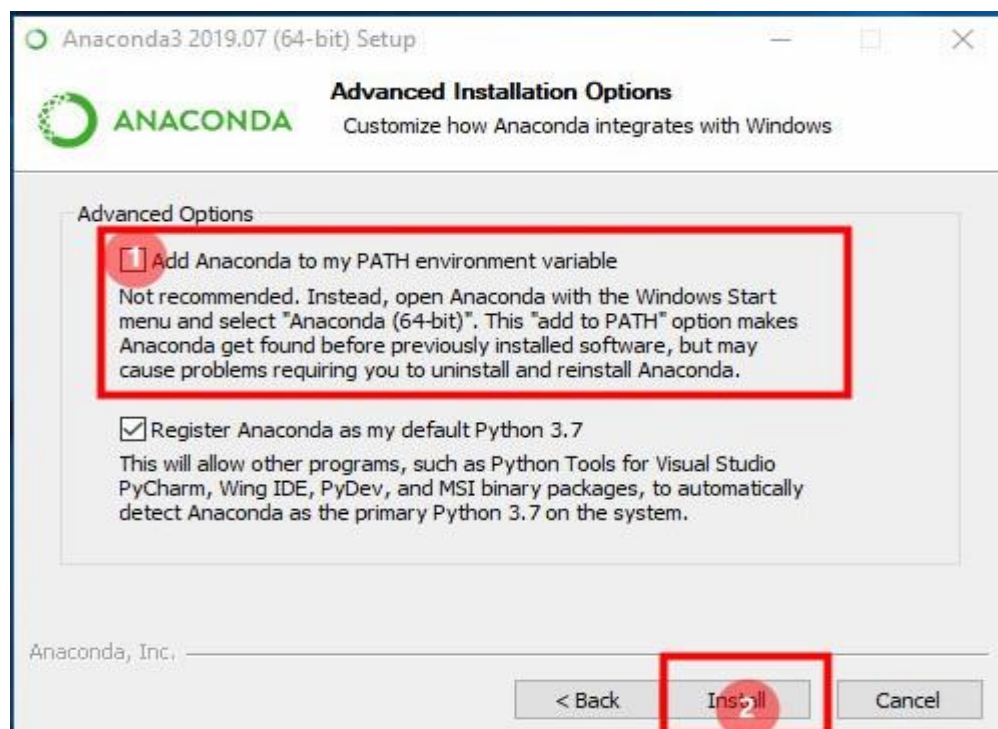
Hình 1.6: Chọn Loại cài đặt

- Bạn chọn “Just Me” sau đó Click **Next**



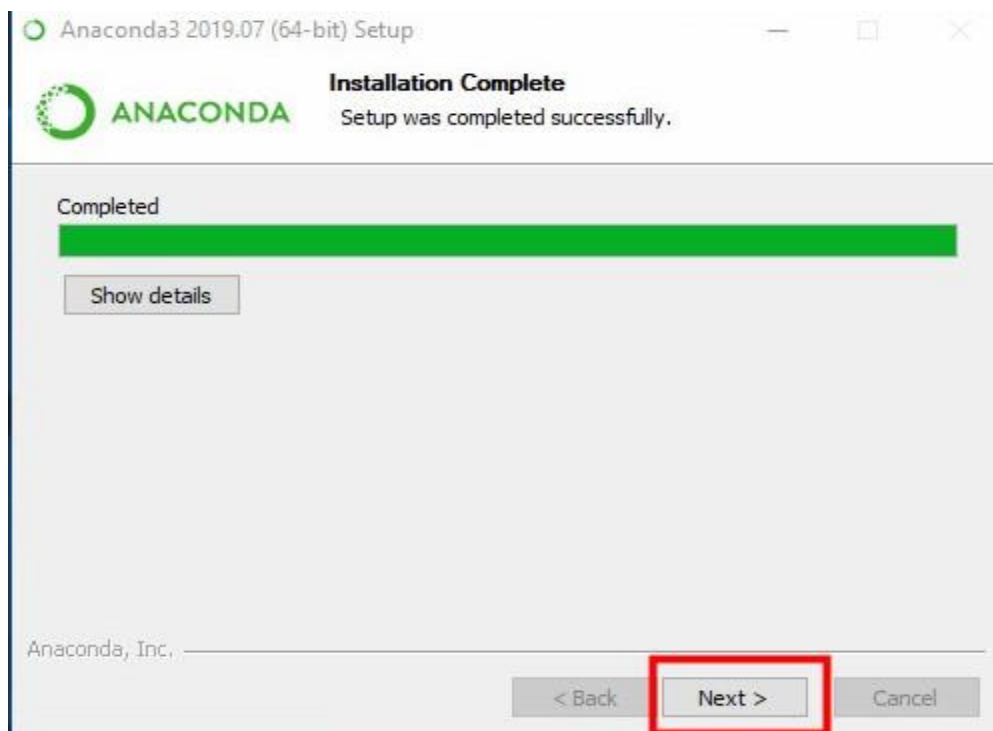
Hình 1. 7: Chọn Vị trí cài đặt

- Click **Next**

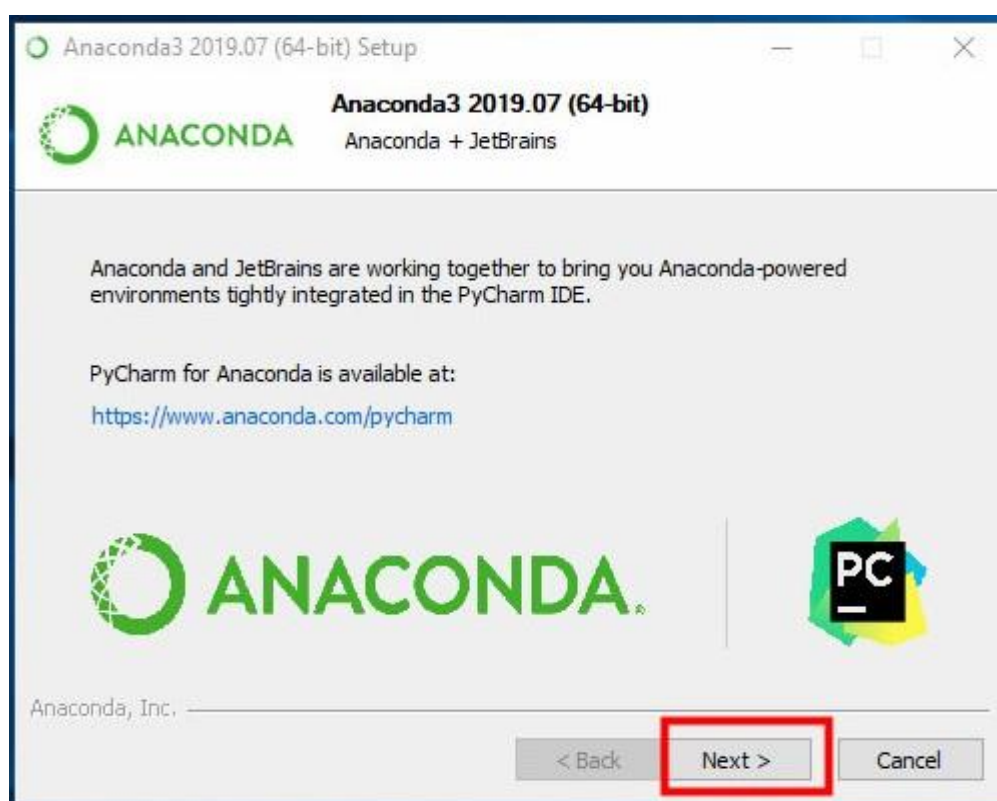


Hình 1. 8: Advanced Options

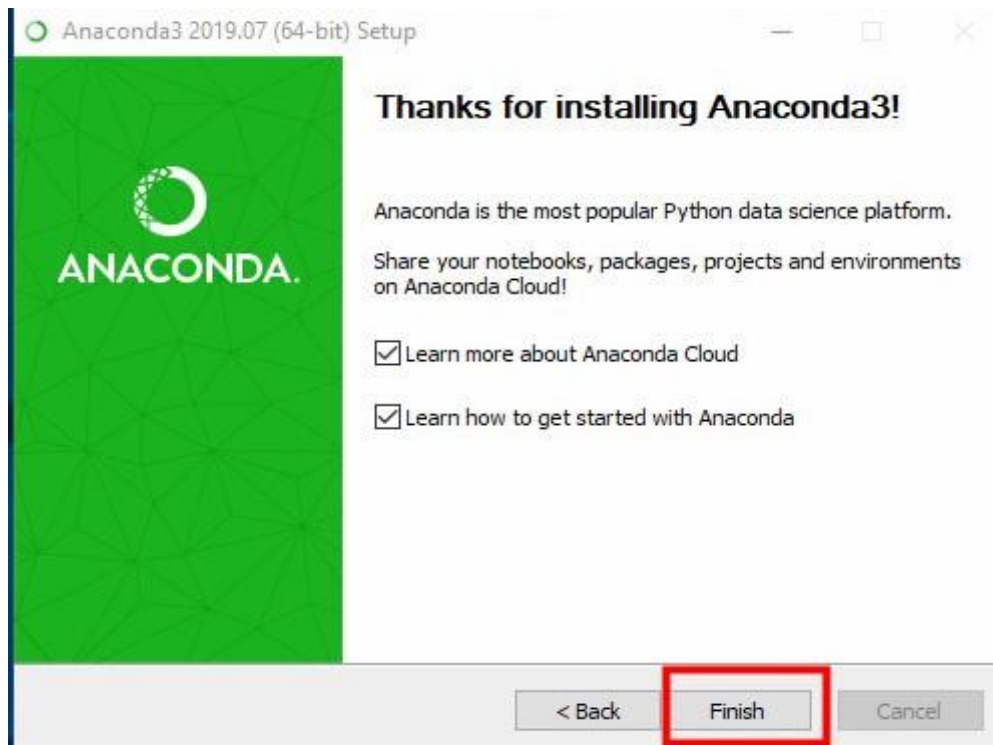
- Lưu ý: hãy chắc là bạn chọn tùy chọn được khoanh đỏ bên dưới để thêm các câu lệnh Anaconda vào **System Environment Variable (PATH)** của Windows
- Sau đó Click **Next**



- *Hình 1. 9: Hoàn thành Cài đặt*
- *Tiếp tục Click Next*



- *Hình 1. 10: Anaconda và JetBrains*
- *Click Next*



Hình 1.11: Cài đặt hoàn tất và khởi động chương trình

- Tiếp theo bạn click **Finish** để hoàn tất việc cài đặt

1.2.3. Hướng dẫn cài đặt thêm thư viện

Anaconda đã có sẵn khá là nhiều thư viện python như : [Numpy](#), [Scipy](#), [Matplotlib](#), [sklearn](#),...

Để kiểm tra python của Anaconda đã có thư viện nào đó, chúng ta sẽ thử import nó trong Console

```

Anaconda Prompt (Anaconda3) - python
(base) C:\Users\vinh-pc>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>

```

Không có lỗi được thông báo nghĩa là python đã biết được thư viện này. Để kiểm tra thư viện này ở đâu, sau khi *import*, ta truy xuất đường dẫn của thư viện như sau:

```

Anaconda Prompt (Anaconda3) - python
(base) C:\Users\vinh-pc>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__file__
'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\numpy\\__init__.py'
>>>

```

Thư viện Numpy của tôi nằm ở đường dẫn
'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\'.

Anaconda đã có sẵn thư viện Numpy

```
Anaconda Prompt (Anaconda3) - python
(base) C:\Users\vinh-pc>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import Scipy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'Scipy'
```

Nếu như Python trả về lỗi Import như trên thì có nghĩa trong Anaconda chúng ta chưa có thư viện đó.

Ở phần trên python của tôi chưa có thư viện *Scipy*, nên tôi phải đi cài đặt nó. Vì tôi sử dụng Anaconda cho lập trình python nên tôi cần phải (1) cài đặt thư viện mới vào đường dẫn *libs python* của Anaconda hoặc (2) chỉ cho python của Anaconda biết về đường dẫn tới thư viện mới này.

Với Anaconda, việc cài đặt 1 thư viện đang được hỗ trợ cực kỳ đơn giản, tôi chỉ cần dùng tools *pip* hoặc *conda* mà Anaconda đã cài sẵn. Cụ thể, ở đây tôi muốn cài thư viện *Scipy* tôi truy cập vào trang chủ của [Scipy](#). Trang này ghi rằng chúng ta có thể cài bằng *pip* hoặc *conda*.

Chúng ta sẽ bật Anaconda Prompt (Anaconda3) lên và gõ `conda install -c anaconda Scipy`. Conda sẽ tự động tìm thư viện *Scipy* và cài vào đường dẫn Anaconda giúp chúng ta.

```
Anaconda Prompt (Anaconda3)
(base) C:\Users\vinh-pc>conda install -c anaconda Scipy
```

Chờ cho thư viện và các thư viện liên quan hoàn tất cài đặt, chúng ta vào spyder kiểm tra lại đã có *Scipy* chưa. Và python trả về đã có *Scipy* trong Anaconda. Và chúng ta đã có thể sử dụng *Scipy*

```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\vinh-pc> python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import scipy
>>> scipy.__file__
'C:\ProgramData\Anaconda3\lib\site-packages\scipy\__init__.py'
>>>
```

Với 1 thư viện chưa có trên Anaconda, cách cài đặt sẽ phức tạp hơn chút nhưng hầu hết các thư viện lớn thường dùng đều có thể cài đặt thông qua Anaconda, nên chúng ta không phải lo lắng lắm.

Ngoài ra chúng ta có thể cài đặt thêm các thư viện bằng Anaconda Navigator

1.3. Hướng dẫn cài đặt IDE Spyder

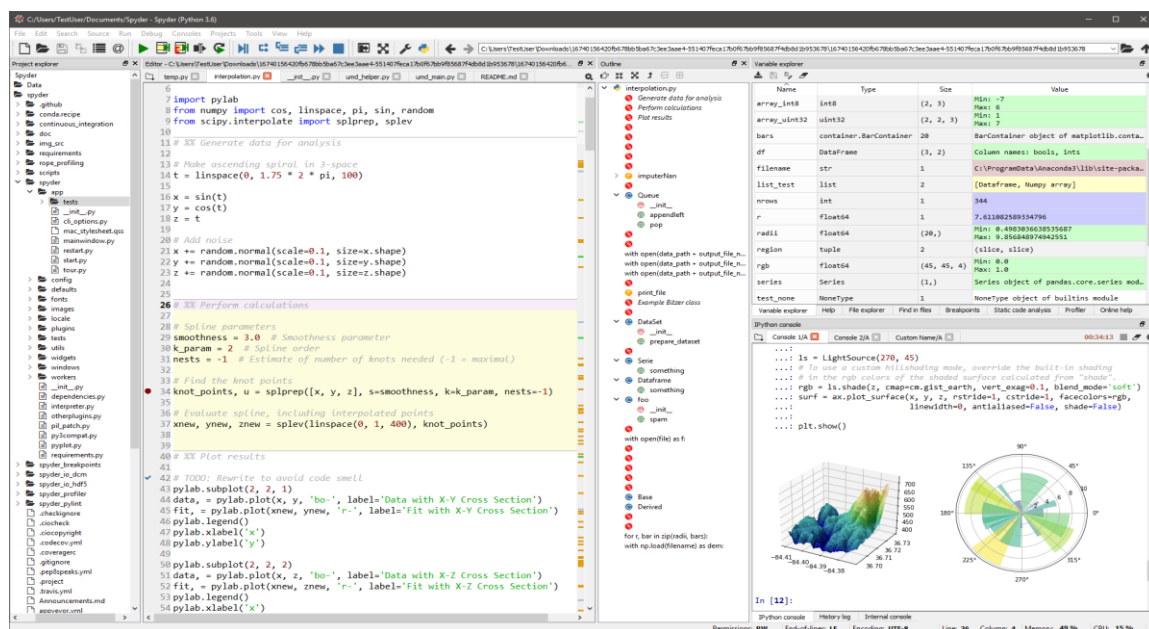
1.3.1. Giới thiệu IDE Spyder

Spyder là một môi trường phát triển Python mã nguồn mở được tối ưu hóa cho các bài toán liên quan đến khoa học dữ liệu. Spyder đi kèm với phân phối quản lý gói Anaconda. Spyder là công cụ thường dùng của các nhà khoa học dữ liệu sử dụng Python. Spyder tích hợp tốt với các thư viện khoa học dữ liệu Python phổ biến như SciPy, NumPy và Matplotlib.

Spyder có hầu hết các tính năng của một “IDE phổ biến”, chẳng hạn như trình soạn thảo mã với chức năng đánh dấu cú pháp mạnh mẽ, tự động hoàn thành mã và thậm chí là trình duyệt tài liệu được tích hợp.

Một tính năng đặc biệt không có trong các môi trường phát triển Python khác là tính năng “khám phá biến” của Spyder cho phép hiển thị dữ liệu bằng cách sử dụng bố cục bảng ngay bên trong IDE. Điều này làm nó trông khá gọn gàng. Nếu bạn thường xuyên làm các bài toán khoa học dữ liệu làm việc bằng cách sử dụng Python, thì đây là một tính năng độc đáo. Việc tích hợp IPython/Jupyter là một đặc điểm nổi bật khác.

Nhìn chung Spyder có nhiều chức năng nổi trội cơ bản hơn các IDE khác. Điều đặc biệt khác là Spyder miễn phí trên Windows, macOS, và Linux và nó là phần mềm mã nguồn mở hoàn toàn.



Hình 1. 12: Giao diện chính của IDE Spyder

✓ **Ưu điểm:** Tối ưu nhiều tính chất phù hợp cho các hoạt động khoa học dữ liệu sử dụng phân phối Python Anaconda.

✓ **Nhược điểm:** Các nhà phát triển Python có kinh nghiệm hơn có thể cảm thấy Spyder quá đơn giản để làm việc hàng ngày và thay vào đó chọn một IDE hoàn chỉnh hơn hoặc một giải pháp biên tập có khả năng tùy chỉnh.

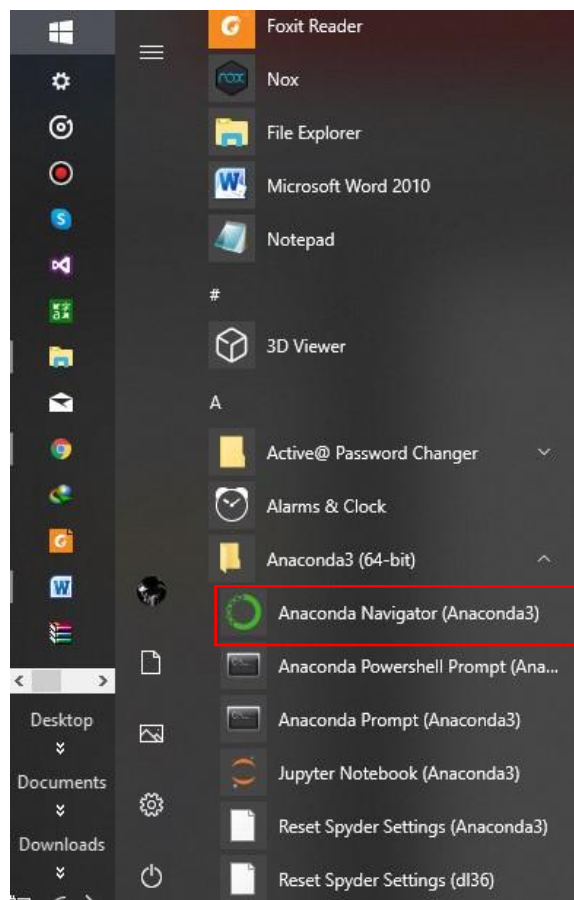
1.3.2. Hướng dẫn cài đặt IDE Spyder bằng Navigator

Spyder tương đối dễ cài đặt trên Windows, Linux và macOS. Chỉ cần chắc chắn để đọc và làm theo các hướng dẫn cẩn thận.

Spyder được bao gồm theo mặc định trong bản phân phối Anaconda Python, đi kèm với mọi thứ bạn cần để bắt đầu trong gói tất cả trong một (thường là khi cài đặt Anaconda thì Spyder đã được mặc định cài đặt).

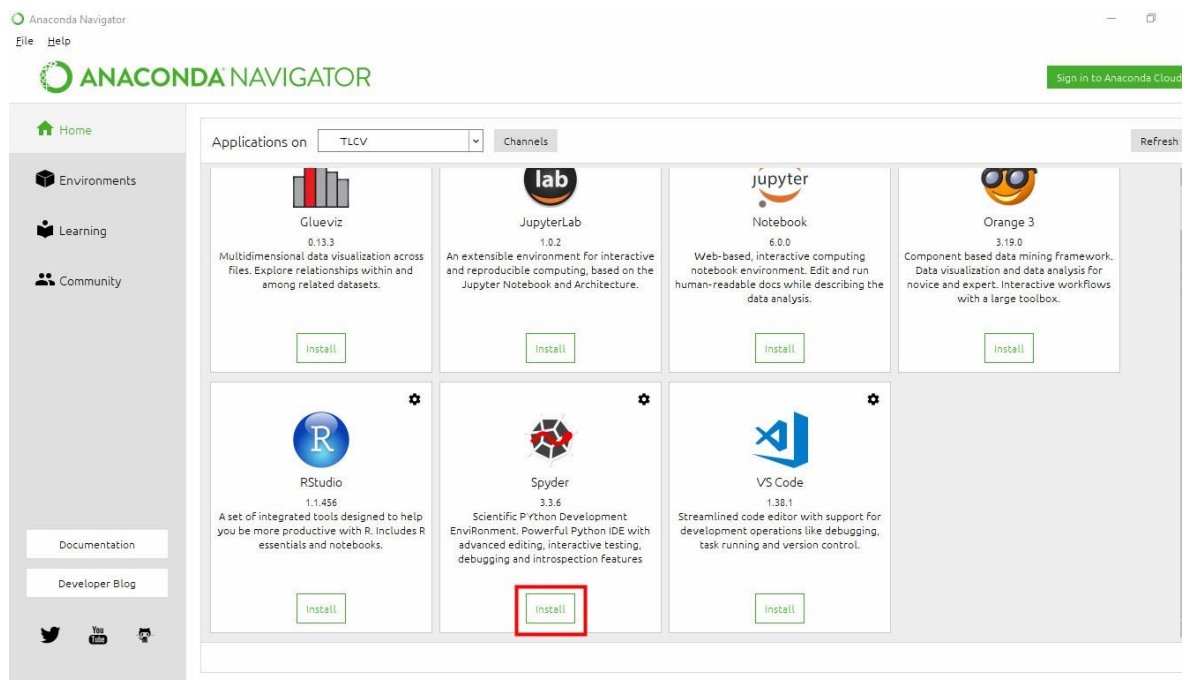
Đây là cách dễ nhất để cài đặt Spyder cho bất kỳ nền tảng được hỗ trợ nào và là cách khuyên bạn nên tránh các sự cố không mong muốn. Bạn nên cài đặt thông qua phương pháp này; nó thường có ít khả năng gây ra những cạm bẫy tiềm tàng cho những người không phải là chuyên gia và có thể cung cấp hỗ trợ hạn chế nếu gặp rắc rối.

Đầu tiên bạn cần khởi động Anaconda Navigator



Hình 1. 13:Chạy Spyder

Sau đó bạn vào phần Home rồi cài đặt Spyder theo phiên bản mà Anaconda hỗ trợ

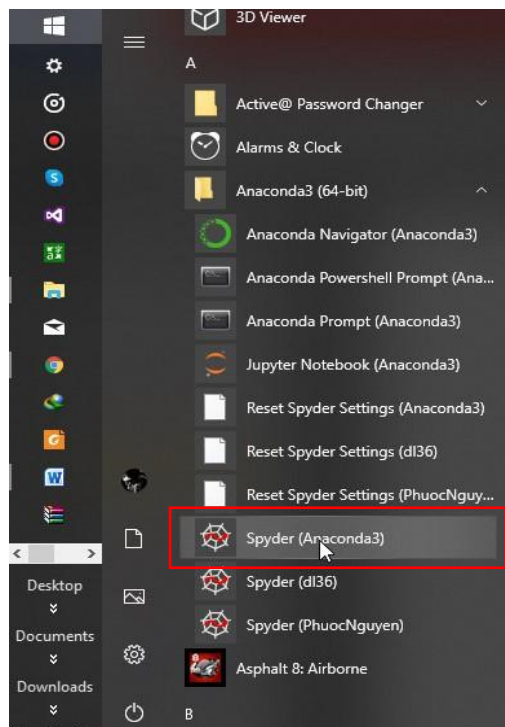


Hình 1. 14: Cài đặt Spyder bằng Anaconda Navigator

Chờ ít phút để chương trình cài đặt hoàn thành.

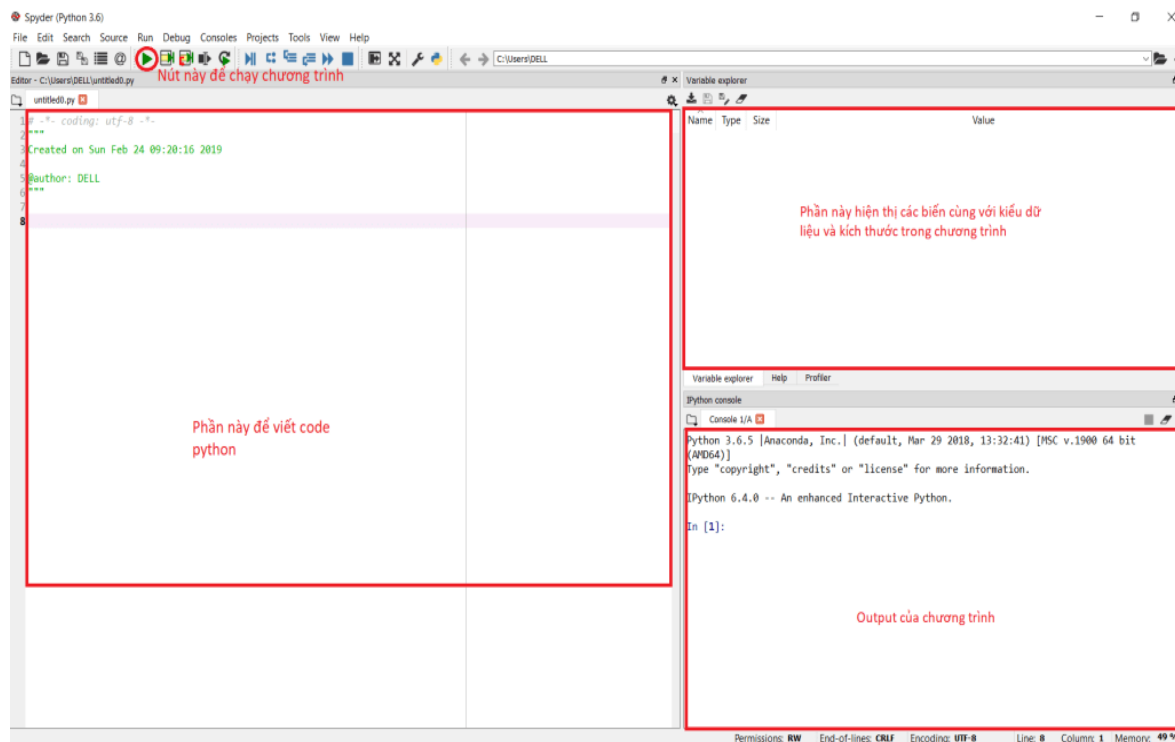
1.3.4. Hướng dẫn sử dụng Spyder

Sau khi cài đặt Spyder hoàn thành, mở Spyder lên và sử dụng.



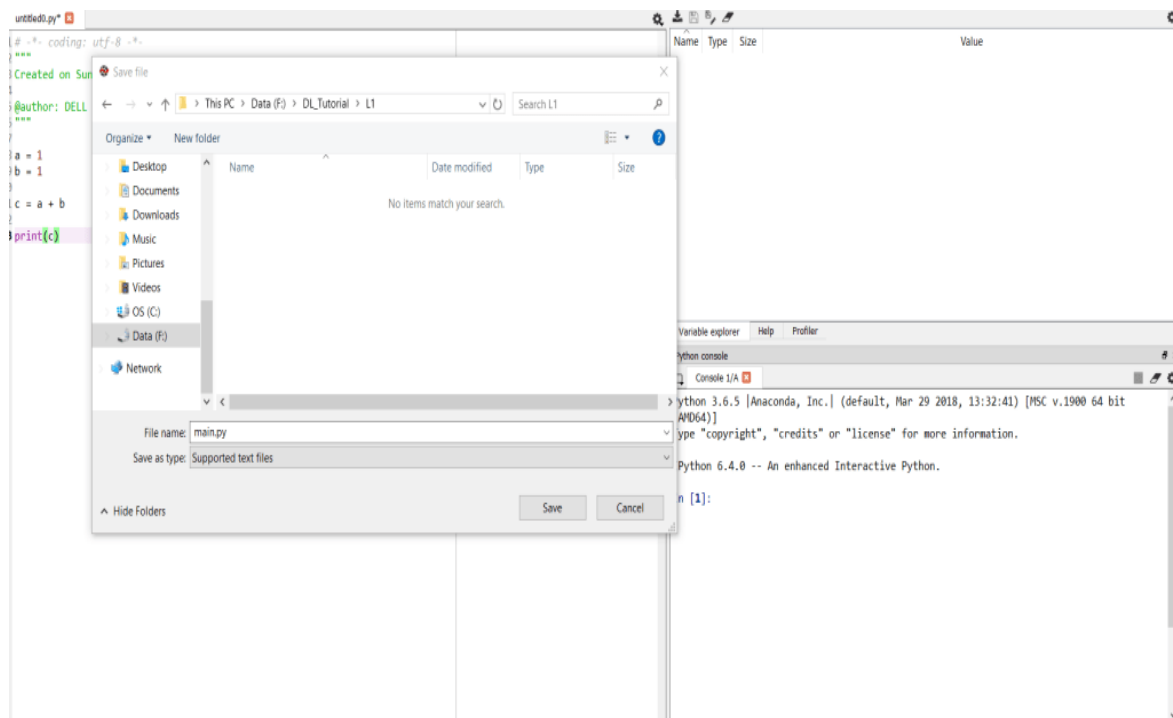
Hình 1. 15: Mở chương trình Spyder

Giao diện chính của Spyder



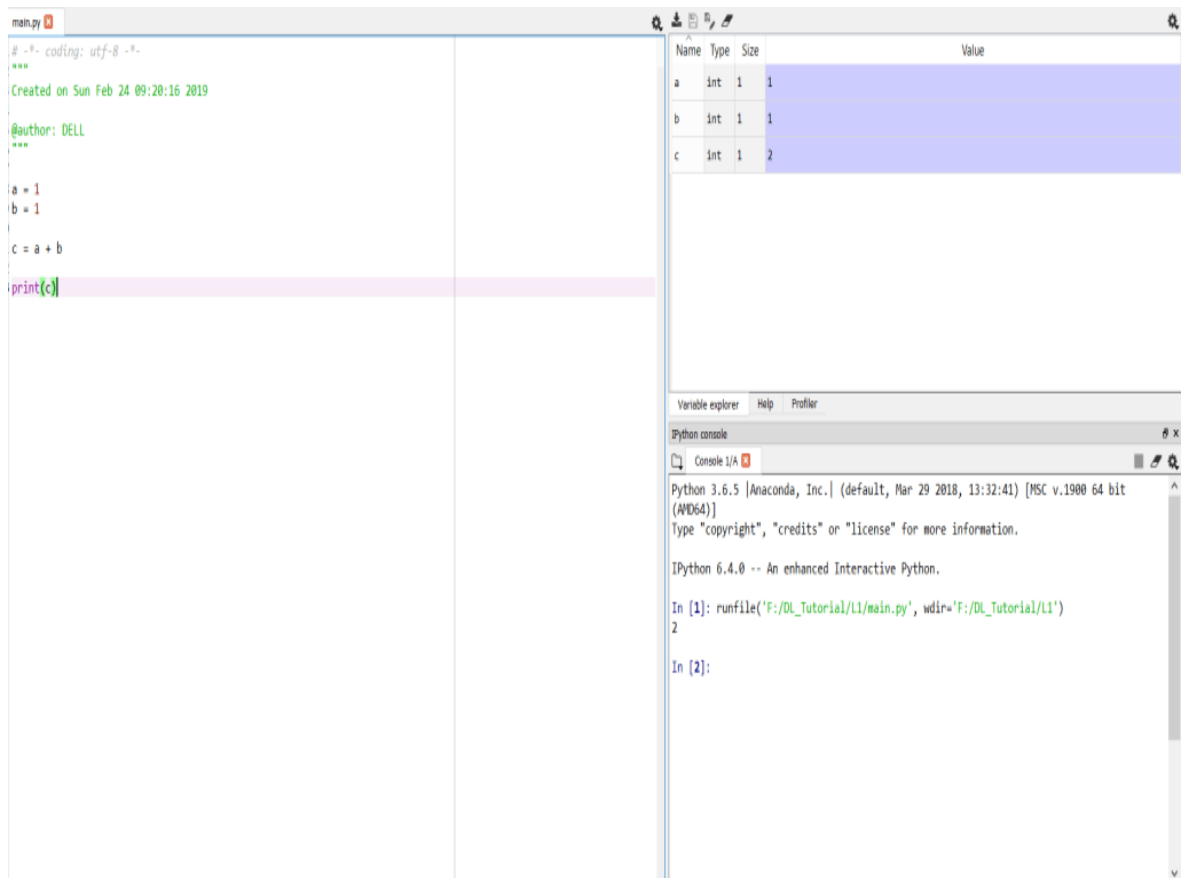
Hình 1. 15: Giao diện chính Spyder

Khi bạn viết code xong ở phần viết code, ấn nút chạy chương trình (hoặc F5) thì bạn cần phải lưu file trước nếu file chưa được lưu.



Hình 1. 17: Lưu chương trình với đuôi *.py

Chọn thư mục để lưu vào viết tên file, tên file **luôn có .py đằng sau**, ví dụ như trong hình là **main.py**



Hình 1. 18: Chạy chương trình

CHƯƠNG 2: TÌM HIỂU MÔN HỌC XỬ LÝ ẢNH

2.1. Các khái niệm cơ bản

2.1.1. Ảnh

Ảnh có thể được hiểu là thông tin (về đường nét, hình khối, màu sắc...) của vật thể hay quang cảnh được chiếu sáng mà con người cảm nhận và quan sát được bằng mắt và hệ thống thần kinh thị giác...

Đối tượng chính của xử lý ảnh chính là ảnh chụp tự nhiên. Quá trình xử lý ảnh được biểu hiệu là xử lý nội dung thông qua dữ liệu ảnh, qua đó nâng cao chất lượng ảnh hiển thị đạt một yêu cầu cảm quan nào đó.

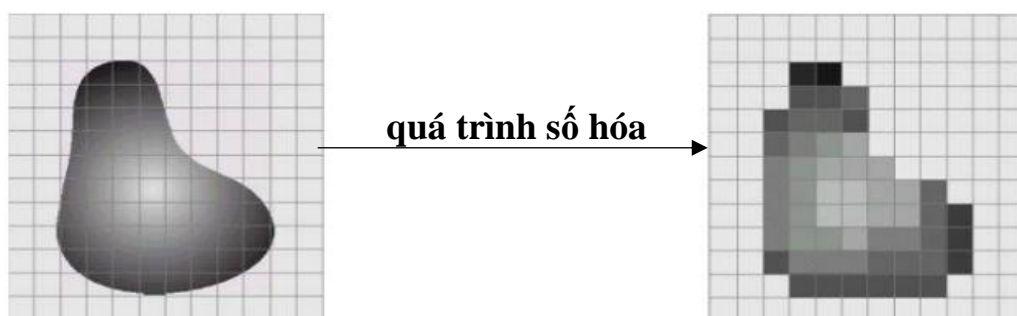
Tín hiệu ảnh thuộc loại *tín hiệu đa chiều*: tọa độ (x,y,z) , độ sáng (λ) , thời gian (t) .

Ảnh tĩnh trong không gian 2 chiều được định nghĩa là một hàm 2 biến $S(x,y)$, với S là giá trị biên độ (được biểu diễn bằng màu sắc) tại vị trí không gian (x,y) .

Có hai loại ảnh cần phải phân biệt:

- ❖ Một là ảnh tương tự $S(x,y)$: (x,y) liên tục, S liên tục.
- ❖ Hai là ảnh số $S(m,n)$: (m,n) rời rạc, S rời rạc.

Quá trình chuyển đổi từ ảnh tương tự sang ảnh số được mô tả như sau:



Ảnh tương tự

Ảnh số hóa

Hình 2. 1: Quá trình chuyển đổi từ ảnh tương tự sang ảnh số

- ❖ Quá trình số hóa có thể được hiểu đơn giản qua các bước:
 - Bước 1: Ảnh tương tự được chia thành M hàng, N cột.
 - Bước 2: Lấy giao của từng hàng và từng cột gọi là pixel

Giá trị biên độ của pixel tại tọa độ nguyên (m,n) là $S(m,n)$: là trung bình độ sáng trong pixel đó. $S(m,n) \leq L$ (L số mức xám dùng biểu diễn ảnh).

2.1.2. Ảnh số

Chúng ta thường nghe người ta gọi ảnh, kèm theo kích thước của nó, tôi lấy ví dụ một ảnh có kích thước 640px * 480 px. Bởi vì cơ bản, ảnh là một ma trận 2 chiều có kích thước với số cột - chiều rộng (640) và số dòng - chiều cao (480), mỗi phần tử trong ma trận là một pixel - điểm ảnh.

Đơn vị cơ bản nhất của ảnh là điểm ảnh (pixel). Do đó, tùy vào giá trị của điểm ảnh, mà ảnh sẽ có kết quả hiển thị khác nhau. Mỗi điểm ảnh có giá trị được lưu trữ bằng số lượng byte (depth) và số kênh màu khác nhau. Có các loại ảnh khác nhau, gồm:

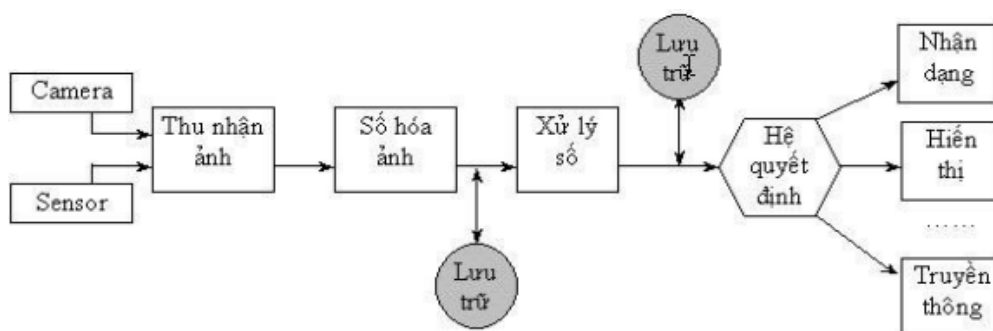
Ảnh nhị phân: Mỗi điểm ảnh có thể nhận một trong hai giá trị là 0 hoặc 1. Thông thường, khi thực hiện thao tác với ảnh nhị phân, người ta sử dụng ảnh xám để lưu trữ. Bằng cách này, ta có cái nhìn trực quan hơn khi thao tác. Giá trị độ xám bằng 0 (biểu diễn cho giá trị nhị phân 0) - màu đen, và giá trị độ xám bằng 255 (biểu diễn cho giá trị nhị phân 1) - màu trắng.

Ảnh xám: Mỗi điểm ảnh trong ảnh xám có một kênh màu duy nhất, thông thường giá trị của kênh màu này được lưu trữ bởi 8 bits, vì thế, ta có thể gọi ảnh xám này có 256 mức xám, mỗi điểm ảnh có thể nhận các giá trị màu từ 0 → 255.



Ảnh màu: Mỗi điểm ảnh là sự kết hợp của 3 hay 4 kênh màu và tùy vào hệ màu. Hệ màu được sử dụng phổ biến là RGB. Mỗi kênh màu được lưu trữ bởi n bits. Vậy, với ảnh màu có 3 kênh màu gồm: R - Red, G - Green, B - Blue; thì số lượng bits để biểu diễn mỗi điểm ảnh là 3n bits và số lượng màu mà điểm ảnh này có thể hiển thị là $2^{(3n)}$.

2.1.3. Hệ thống xử lý ảnh



Hình 2.2: Mô hình biểu diễn một hệ thống xử lý ảnh

Trong đó:

- *Thu nhận ảnh*: Là quá trình thu ảnh từ các thiết bị hỗ trợ như:
 - + Camera (tương tự, số).
 - + Từ vệ tinh qua các bộ cảm ứng (Sensors).
 - + Các máy quét ảnh (Scanners).
- *Số hóa ảnh*: Là quá trình biến đổi ảnh tương tự thành ảnh rời rạc để xử lý bằng máy tính thông qua quá trình lấy mẫu (rời rạc về mặt không gian) và lượng tử hóa (rời rạc về mặt biên độ).
- *Xử lý số*: Là một quá trình gồm nhiều công đoạn nhỏ như tăng cường ảnh (Enhancement), khôi phục ảnh (Restoration), phát hiện biên (Edge Detection), phân vùng ảnh (Segmentation), trích chọn các đặc tính (Feature Extraction)...
- *Hệ quyết định*: Tùy thuộc vào mục đích của ứng dụng mà chuyển sang giai đoạn khác là hiển thị, nhận dạng, phân lớp, truyền thông...

2.1.4. Biểu diễn ảnh

Ảnh có thể xem là một hàm 2 biến chứa các thông tin như biểu diễn của một ảnh.

Các mô hình biểu diễn ảnh cho ta một *mô tả logic* hay *định lượng* của hàm này dựa vào các phần tử đặc trưng của ảnh đó là pixel.

Giá trị pixel có thể là một giá trị vô hướng, hoặc là 1 vector (3 thành phần trong trường hợp ảnh màu).

Ta có thể biểu diễn ảnh bằng *hàm toán học*, hoặc *các ma trận điểm*. Trong mô hình toán học, ảnh hai chiều được biểu diễn nhờ các hàm hai biến, đó là:

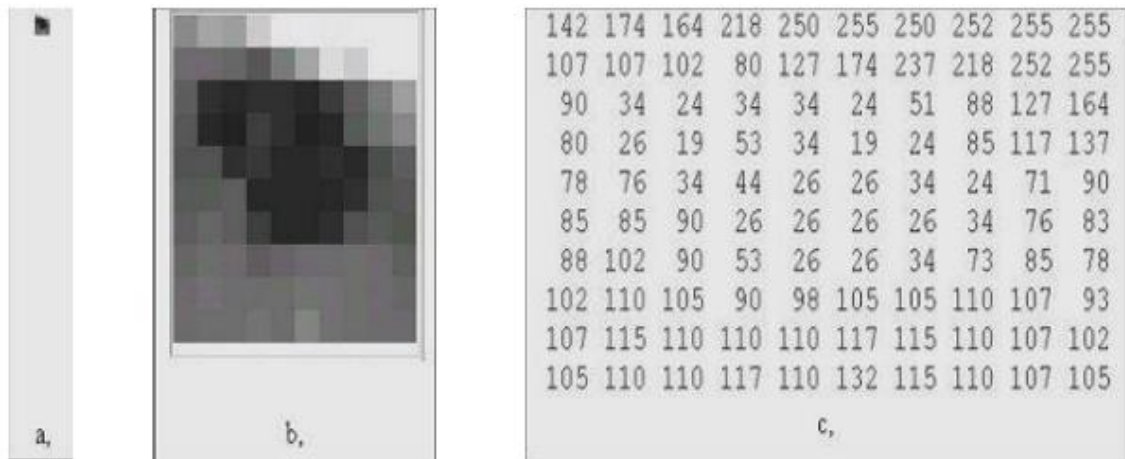
$$S(m,n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} S(k,l) \delta(m-k, n-l) \quad \text{với } 0 \leq m, k \leq M-1, \quad 0 \leq n, l \leq N-1$$

❖ Biểu diễn bằng hàm toán học

Trong các hàm toán học biểu diễn ảnh:

- (m,n) : Tọa độ của Pixel trong miền không gian (2D)
- $S(m,n)$: Độ sáng (Mức xám) của pixel (m,n) .
- $[0 - Lmax]$: Thang mức xám. $Lmax$ thường là 255, nghĩa là chúng ta đang sử dụng thang mức xám 8 bit.

❖ Biểu diễn bằng ma trận điểm



a, Ảnh thật 10x10 b, Ảnh được zoom c, Mô tả ảnh bằng ma trận điểm

Hình 2. 3: Biểu diễn ảnh bằng ma trận điểm

2.2. Xử lý ảnh là gì?

Xử lý ảnh (Xử lý ảnh số) hay còn được gọi là xử lý tín hiệu số là một môn học hết sức cơ bản cho xử lý tín hiệu chung, các khái niệm về tích chập, các biến đổi Fourier, biến đổi Laplace, các bộ lọc hữu hạn... Đồng hành cùng xử lý tín hiệu số là các công cụ toán học hay kiến thức cần thiết như Đại số tuyến tính, Xác suất – thống kê, Trí tuệ nhân tạo, Mạng nơ – ron nhân tạo,...

Ngày nay xử lý ảnh đã được áp dụng rất rộng rãi trong đời sống như: photoshop, nén ảnh, nén video, nhận dạng biển số xe, nhận dạng khuôn mặt, nhận dạng chữ viết, xử lý ảnh thiên văn, ảnh y tế,....

2.2.1. Lịch sử phát triển

Ứng dụng đầu tiên của xử lý ảnh số được biết đến là nâng cao chất lượng ảnh báo được truyền qua cáp từ Luân đôn đến New York vào những năm 1920. Vấn đề nâng cao chất lượng ảnh có liên quan tới phân bố mức sáng và độ phân giải của ảnh. Và một trong những ứng dụng đầu tiên của hình ảnh kỹ thuật số là trong tin tức- ngành công nghiệp giấy.

- ✚ Hình ảnh cáp Bartlane dịch vụ truyền dẫn
- ✚ Hình ảnh được chuyển bằng cáp ngầm giữa Luân Đôn và New York
- ✚ Hình ảnh được mã hóa để chuyển cáp và được xây dựng lại ở cuối nhận trên một máy in điện báo

Giữa đến cuối những năm 1920: Những cải tiến cho Hệ thống Bartlane dẫn đến chất lượng cao hơn hình ảnh

- ✚ Sinh sản mới quy trình dựa trên ảnh kỹ thuật
- ✚ Số lượng tăng lên của tông màu trong hình ảnh sao chép

Những năm 1960: Những cải tiến trong điện toán công nghệ và sự khởi đầu của cuộc đua không gian dẫn đến một sự đột biến của công việc trong hình ảnh kỹ thuật số

Chế biến

- ✚ 1964: Máy tính được sử dụng để nâng cao chất lượng hình ảnh mặt trăng chụp bởi đầu dò *Ranger 7*
- ✚ Kỹ thuật như vậy đã được sử dụng trong các nhiệm vụ không gian khác bao gồm cả cuộc đổ bộ Apollo

Những năm 1970: Xử lý hình ảnh kỹ thuật số bắt đầu được sử dụng trong các ứng dụng y tế

- ✚ 1979: Ngài Godfrey N.Hounsfield & Giáo sư Allan M.Cormack chia sẻ giải Nobel Giải thưởng về y học cho phát minh chụp cắt lớp, công nghệ đằng sau: Máy tính hướng trục. Chụp cắt lớp (CAT) quét

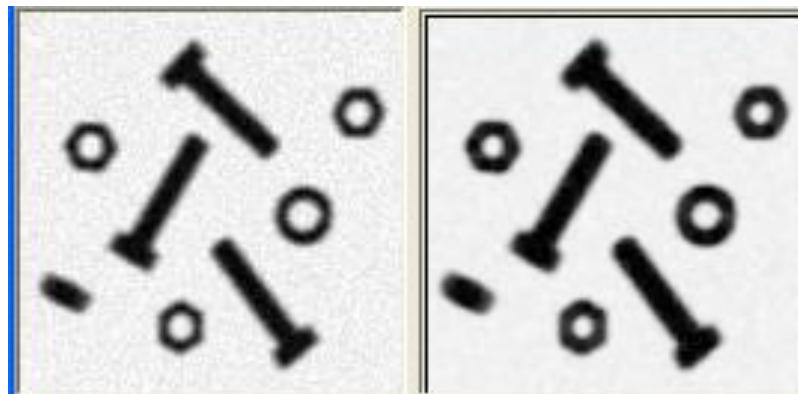
Những năm 1980 - Ngày nay: Việc sử dụng hình ảnh kỹ thuật số kỹ thuật chế biến đã bùng nổ và bây giờ chúng được sử dụng cho tất cả các loại nhiệm vụ các loại khu vực

- ✚ Cải thiện / phục hồi hình ảnh
- ✚ Hiệu ứng nghệ thuật
- ✚ Trục quan y tế
- ✚ Kiểm tra công nghiệp
- ✚ Thực thi pháp luật
- ✚ Giao diện máy tính của con người

2.2.2. Các lĩnh vực ứng dụng

a) Tăng cường ảnh

Mục đích: Tăng cường các thuộc tính cảm nhận, làm cho ảnh tốt lên theo một ý nghĩa nào đó, tiện phục vụ cho các xử lý tiếp theo.



Hình 2.4 : Hình ảnh trước và sau khi tăng cường

Các thao tác: Thay đổi độ tương phản, thay đổi màu sắc, thay đổi cường độ sáng, lọc nhiễu, nội suy, làm trơn ảnh,...

Các phương pháp chính: Các thao tác trên điểm (Point operation), các thao tác không gian (Spatial operation)

b) Khôi phục ảnh

Mục đích: Khôi phục ảnh ban đầu, loại bỏ các biến dạng ra khỏi ảnh tùy theo nguyên nhân gây biến dạng.

Các thao tác: Lọc nhiễu, giảm độ méo,...

Các phương pháp: Lọc ngược, lọc thích nghi (Wiener), khôi phục ảnh từ các hình chiếu.



Hình 2.5 : Hình ảnh trước và sau khi lọc nhiễu

c) Biến đổi ảnh

Mục đích: Biến đổi thể hiện của ảnh dưới các góc nhìn khác nhau tiến cho việc xử lý, phân tích ảnh.

Các phương pháp: Biến đổi Fourier, Sin, Cosin, tích chập, tích Kronecker, KL (Karhunen Loeve), Hadamard,

d) Phân tích ảnh

Mục đích: Tìm ra các đặc trưng của ảnh, xây dựng quan hệ giữa chúng dựa vào các đặc trưng cục bộ.

Các thao tác: Tìm biên, tách biên, làm mảnh đường biên, phân vùng ảnh, phân loại đối tượng.

Các phương pháp: Phương pháp phát hiện biên cục bộ, dò biên theo qui hoạch động, phân vùng theo miền đồng nhất, phân vùng dựa theo đường biên...

e) Nén ảnh

Mục đích: Giảm không gian lưu trữ, thuận tiện truyền thông trên mạng.

Phương pháp: Nén không mất thông tin, nén mất thông tin

- Nén không mất thông tin (nén chính xác): Khai thác các thông tin dư thừa.
- Nén mất thông tin: Khai thác các thông tin dư thừa và các thông tin không liên quan.

Hiện nay có một số chuẩn nén hay dùng: JPEG, MPEG (JPEG-2000, MPEG-4).

f) Nhận dạng ảnh:

Nhận dạng ảnh là quá trình liên quan đến mô tả đối tượng mà người ta muốn đặc tả nó. Thường đi sau quá trình trích chọn các đặc tính của đối tượng.

Có 2 kiểu mô tả đối tượng:

- Mô tả theo tham số (nhận dạng theo tham số).
- Mô tả theo cấu trúc (nhận dạng theo cấu trúc).

Ứng dụng: nhận dạng đối tượng, mặt, vân tay, văn bản...

1.2.3. Một số thư viện nổi tiếng trong xử lý ảnh

a) Scikit –Image

Scikit –Image là một gói Python mã nguồn mở hoạt động với các mảng Numpy. Nó thực hiện các thuật toán và tiện ích để sử dụng trong nghiên cứu, giáo dục và các ứng dụng công nghiệp. Đây là một thư viện khá đơn giản và dễ hiểu, ngay cả đối với những người chưa quen với môi trường của Python. Mã này có chất lượng cao, được đánh giá ngang hàng với các thư viện nổi tiếng khác.

b) Numpy

Numpy là một trong những thư viện cốt lõi trong lập trình Python và cung cấp hỗ trợ cho các mảng. Một hình ảnh về cơ bản là một mảng NumPy tiêu chuẩn chứa các pixel của các điểm dữ liệu. Do đó, bằng cách sử dụng các thao tác NumPy cơ bản, chẳng hạn như cắt, mặt nạ và lập chỉ mục ưa thích, bạn có thể sửa đổi các giá trị pixel của hình ảnh. Hình ảnh có thể được tải bằng cách sử dụng `skimage` và hiển thị bằng `Matplotlib`.

d) Scipy

SciPy là một mô-đun khoa học cốt lõi khác của Python (như NumPy) và có thể được sử dụng cho các tác vụ xử lý và xử lý ảnh cơ bản. Cụ thể, mô-đun `scipy.ndimage` (trong SciPy v1.1.0) cung cấp các hàm hoạt động trên mảng NumPy n chiều. Gói hiện bao gồm các chức năng lọc tuyến tính và phi tuyến tính, hình thái nhị phân, nội suy B-spline và đo lường đối tượng.

d) PIL/Pillow

PIL (Thư viện hình ảnh Python) là một thư viện miễn phí cho ngôn ngữ lập trình Python có thêm hỗ trợ mở, thao tác và lưu nhiều định dạng tệp hình ảnh khác nhau. Tuy nhiên, sự phát triển của nó đã bị đình trệ, với bản phát hành cuối cùng vào năm 2009. May mắn thay, có **Pillow**, một nhánh của PIL được phát triển tích cực, dễ cài đặt hơn, chạy trên tất cả các hệ điều hành chính và hỗ trợ Python 3. Thư viện chứa hình ảnh cơ bản chức năng xử lý, bao gồm các hoạt động điểm, lọc với một tập hợp các hạt tích chập tích hợp và chuyển đổi không gian màu.

e) OpenCV

OpenCV (Thư viện thị giác máy tính nguồn mở) là một trong những thư viện được sử dụng rộng rãi nhất cho các ứng dụng thị giác máy tính. OpenCV-Python là API Python cho OpenCV. OpenCV-Python không chỉ nhanh, vì nền bao gồm mã được viết bằng C / C ++, mà còn dễ mã hóa và triển khai (do trình bao bọc Python ở nền trước). Điều này làm cho nó trở thành một lựa chọn tuyệt vời để thực hiện các chương trình thị giác máy tính chuyên sâu.

Ngoài ra còn có 1 số thư viện nổi tiếng khác như: SimpleCV, Mahotas, SimpleITK, pgmagick, pycaigo, matplotlib,...

f) Tkinter

Tkinter là một gói trong Python có chứa module Tk hỗ trợ cho việc lập trình GUI. Tk ban đầu được viết cho ngôn ngữ Tcl. Sau đó Tkinter được viết ra để sử dụng Tk bằng trình thông dịch Tcl trên nền Python. Ngoài Tkinter ra còn có một số công cụ khác giúp tạo một ứng dụng GUI viết bằng Python như wxPython, PyQt, và PyGTK.

g) Math

Thư viện math trong Python hỗ trợ rất nhiều hàm tính toán liên quan đến toán học.

2.3. Phương pháp và kỹ thuật phép biến đổi Hough

2.3.1. Tổng quan về biến đổi Hough (HT- Hough transform)

Biến đổi Hough (HT) (Hough, 1962) là một kỹ thuật mà nằm trong hình dạng hình ảnh. Đặc biệt, HT đã được sử dụng để trích chọn đường thẳng, hình tròn và hình elip (hoặc cắt hình nón). Trong trường hợp đường thẳng, xác định toán học của nó tương đương với biến đổi Radon (Deans, 1981). HT được giới thiệu bởi Hough (Hough, 1962) và sau đó được sử dụng để tìm đường bong bóng chứ không phải là hình dạng trong hình ảnh. Tuy nhiên, Rosenfeld ghi nhận lợi thế tiềm năng của HT như một thuật toán xử lý hình ảnh (Rosenfeld, 1969). HT do đó đã được thực hiện để tìm đường thẳng trong hình ảnh (Duda, 1972) và nó đã được mở rộng rất nhiều, vì HT có nhiều lợi thế và nhiều tuyến đường tiềm năng để cải thiện.

Ưu điểm chính của HT là nó có thể cung cấp các kết quả tương tự như đối với đối sánh mẫu nhưng nhanh hơn (Princen, 1992), (Sklansky, 1978), (Stockman,1977). Điều này đạt được bởi một tái định dạng lại của quá trình đối sánh mẫu, dựa trên một phương pháp tiếp cận thu thập dấu hiệu là các bình chọn trong một mảng tích lũy. Việc thực hiện HT xác định một ánh xạ từ các điểm ảnh vào một không gian tích lũy (không gian Hough). Ánh xạ được thực hiện trong một cách tính toán hiệu quả dựa trên các hàm mô tả hình dạng mục tiêu. Ánh xạ này đòi hỏi ít tài nguyên hơn nhiều so với đối sánh mẫu. Tuy nhiên, nó vẫn đòi hỏi lưu trữ quan trọng và yêu cầu tính toán cao. Những vấn đề này được giải quyết sau, kể từ khi họ cung cấp tập trung cho sự phát triển liên tục của HT. Tuy nhiên, thực tế là HT tương đương với đối sánh mẫu đã đưa ra đủ động lực cho kỹ thuật này là trong số phổ biến nhất của tất cả các kỹ thuật trích chọn hình dạng hiện có.

2.3.2. Biến đổi Hough là gì?

- Biến đổi Hough là một phương pháp trích xuất tính năng để phát hiện các hình dạng đơn giản như hình tròn, đường vv trong hình ảnh.
- Một hình dạng đơn giản của người Viking là một hình dạng có thể được biểu thị chỉ bằng một vài tham số. Ví dụ: một dòng có thể được biểu thị bằng hai tham số (độ dốc, giao thoa) và một vòng tròn có ba tham số - tọa độ của tâm và bán kính (x , y , r). Biến đổi Hough thực hiện một công việc tuyệt vời trong việc tìm kiếm các hình dạng như vậy trong một hình ảnh.
- Ưu điểm chính của việc sử dụng biến đổi Hough là nó không nhạy cảm với tắc.

2.3.3. Biến đổi Hough để phát hiện các đường thẳng trong một hình ảnh

2.3.3.1. Hough Line Transform

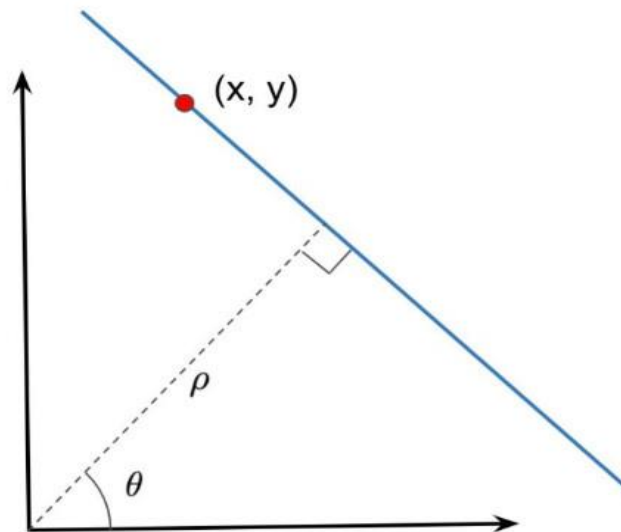
The Hough Line Transform là một biến đổi được sử dụng để phát hiện các đường thẳng.

Để áp dụng Transform đầu tiên chúng ta sẽ xem xét việc tìm kiếm các đường thẳng trong một hình ảnh

2.3.3.2. Hoạt động

Như các bạn đã biết một đường thẳng trong không gian hình ảnh có thể được thể hiện bằng 2 biến:

- Trong hệ tọa độ Descartes: Parameters: (m, b)
- Trong hệ tọa độ Polar: Parameters (r, θ)



Hình 2.6 : Đường thẳng trong hệ tọa độ Polar

Đối vs Hough Transform chúng ta biểu diễn các đường thẳng trong hệ tọa độ Polar. Do đó phương ta có phương trình đường thẳng:

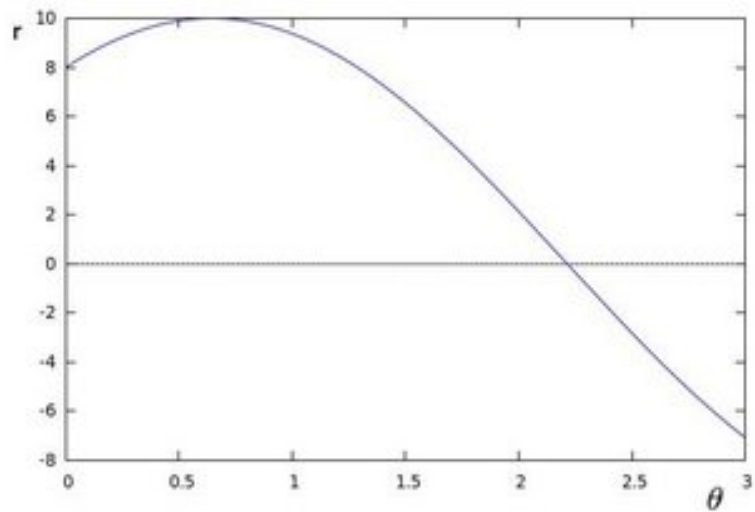
$$y = \left(-\frac{\cos \theta}{\sin \theta} \right) x + \left(\frac{r}{\sin \theta} \right)$$

Từ hình 2 suy ra: $r = x \cdot \cos \theta + y \cdot \sin \theta$

Tại mỗi điểm (x0, y0), chúng ta xác định được đường thẳng đi qua điểm đó là: $r \theta = x_0 \cdot \cos \theta + y_0 \cdot \sin \theta$

Có nghĩa là mỗi cặp (r, θ) đại diện cho mỗi đường thẳng đi qua (x0, y0)

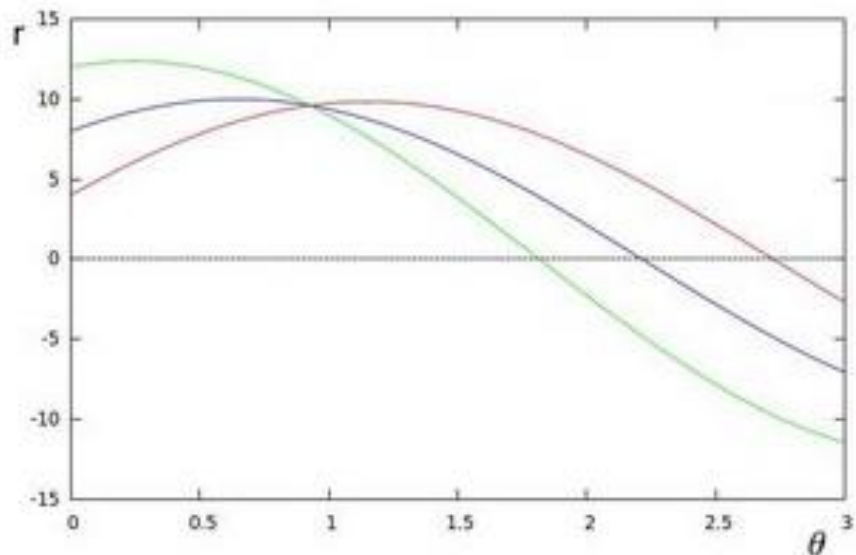
Nếu chúng ta cho cặp (x_0, y_0) , chúng ta sẽ vẽ các đường đi qua nó, chúng ta có được đường hình sin. Ví dụ với $x_0=8, y_0=6$ chúng ta sẽ có sơ đồ sau (trong 1 mặt phẳng $\theta - r$):



Hình 2.7: Sơ đồ hình sin đi qua điểm (x_0, y_0)

Xét trong khoảng $r>0$ và $0<\theta<2\pi$.

Chúng ta có thể thực hiện thao tác tương tự như trên cho tất cả các điểm trong một hình ảnh. Nếu các đường sin của hai điểm khác nhau giao nhau trong mặt phẳng $\theta - r$, điều đó có nghĩa 2 điểm cùng thuộc 1 đường thẳng. Ví dụ: vẽ thêm 2 điểm $x_1=4, y_1=9$ và $x_2=12, y_2=3$. Ta được sơ đồ:



Hình 2.8 : Sơ đồ hình Sin đi qua 3 điểm

Ba đường hình sin trên giao nhau tại điểm (0.925,9.6), các tọa độ này là tham số (θ, r)

Điều này có nghĩa là một đường thẳng được phát hiện bằng cách tìm số giao điểm giữa các đường hình sin. Càng nhiều đường giao nhau thì đường thẳng đó đại diện cho các giao điểm của các đường cong hình sin. Nói chung chúng ta có thể xác định số giao điểm tối thiểu cần thiết để phát hiện 1 đường thẳng.

Đây là những điều mà Hough Line Transform thực hiện. Nó xác định giao điểm của các đường cong của tất cả các điểm trong ảnh. Nếu số lượng giao điểm cao hơn mức tối thiểu chúng ta đưa ra thì sẽ phát hiện được 1 đường thẳng với các tham số (θ, r) của các điểm giao nhau.

2.3.3.3 Hough Line: Cách phát hiện các dòng bằng OpenCV

Trong OpenCV, phát hiện dòng bằng Hough Transform được triển khai trong chức năng **HoughLines** và **HoughLinesP** [Biến đổi xác suất Hough].

Đầu tiên phát hiện các cạnh bằng Canny và chuyển ảnh về dạng xám:

```
src = cv.imread(f)
dst = cv.Canny(src, 50, 200)
cdst = cv.cvtColor(dst, cv.COLOR_GRAY2BGR)
```

Bây giờ chúng ta sẽ giải thích hai hàm này:

HoughLines()

```
lines = cv.HoughLines(dst, 1, math.pi/180.0, 50, np.array([]), 0, 0)
```

Với các đối số:

- dst: kết quả phát hiện cạnh bằng Canny, hình ảnh ở dạng xám
- lines: vector lưu giữ các tham số (r, θ) của các đường được phát hiện
- rho: độ phân giải của r (pixel): 1 pixel
- theta: độ phân giải của θ (radian): $\text{math.pi}/180$
- threshold: số giao điểm tối thiểu để phát hiện đường thẳng: 50
- *srn* and *stn*: tham số mặc định là 0

Sau đó hiển thị kết quả bằng cách vẽ các đường

```
if lines is not None:
    a,b,c = lines.shape
    for i in range(a):
        rho = lines[i][0][0]
```



```

theta = lines[i][0][1]
a = math.cos(theta)
b = math.sin(theta)
x0, y0 = a*rho, b*rho
pt1 = ( int(x0+1000*(-b)), int(y0+1000*(a)) )
pt2 = ( int(x0-1000*(-b)), int(y0-1000*(a)) )
cv.line(cdst, pt1, pt2, (0, 0, 255), 3, cv.LINE_AA)

```

HoughLinesP()

```
lines = cv.HoughLines(dst, 1, math.pi/180.0, 40, np.array([]), 50, 10)
```

Với các đối số:

- dst: kết quả phát hiện cạnh bằng Canny, hình ảnh ở dạng xám
- lines: vector lưu giữ các tham số (r,θ) của các đường được phát hiện
- rho: độ phân giải của r (pixel): 1 pixel
- theta: độ phân giải của θ (radian): math.pi/180
- threshold: số giao điểm tối thiểu để phát hiện đường thẳng
- minLinLength: số giao điểm tối thiểu tạo thành một đường thẳng (50), nếu nhỏ hơn thì bỏ qua
- minLinGap: khoảng cách tối đa giữa 2 điểm được xét trong cùng 1 đường:10

Sau đó hiển thị kết quả bằng cách vẽ các đường

```

a,b,c = lines.shape
for i in range(a):
    cv.line(cdst, (lines[i][0][0], lines[i][0][1]),
    (lines[i][0][2], lines[i][0][3]), (0, 0, 255), 3, cv.LINE_AA)

```

Cuối cùng hiển thị ảnh gốc và các đường thẳng được phát hiện:

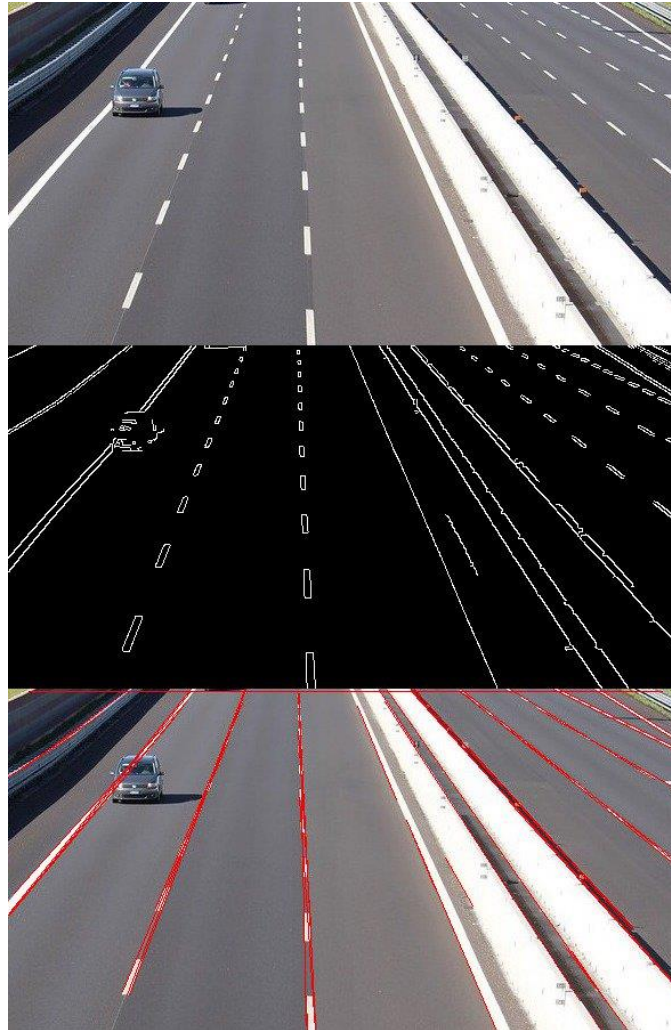
```

cv.imwrite("done.jpg",cdst)
im3 = Image.open("done.jpg")
im3 = ImageTk.PhotoImage(im3)
global label2
label2 = Label(image = im3)
label2.image = im3
label2.place(x=720,y=250)

```

Kết quả phát hiện dòng

Dưới đây chúng tôi hiển thị kết quả của việc sử dụng biến đổi Hough để phát hiện dòng. Hãy nhớ rằng chất lượng của các đường được phát hiện phụ thuộc rất nhiều vào chất lượng của bản đồ cạnh. Do đó, trong thế giới thực, biến đổi Hough được sử dụng khi bạn có thể kiểm soát môi trường và do đó có được các bản đồ cạnh nhất quán hoặc khi bạn có thể huấn luyện máy dò cạnh cho loại cạnh cụ thể mà bạn đang tìm kiếm.



Hình 2.9: Phát hiện dòng bằng Hough Transform

2.3.4. Biến đổi Hough để phát hiện các hình tròn trong một hình ảnh

2.3.4.1. Hough Circle Transform

- Hoạt động tương tự Hough Line Transformb được giải thích ở phần trước
- Trong trường hợp xác định hình tròn chúng ta cần 3 tham số

C:(xcenter,ycenter,r)

Trong đó (x_{center}, y_{center}) là tâm điểm của hình tròn còn r là bán kính của hình tròn đó



Hình 2.10 : Hình tròn

Phương pháp gradient Hough, được tạo thành từ hai giai đoạn chính. Giai đoạn đầu tiên liên quan đến phát hiện cạnh và tìm kiếm các trung tâm vòng tròn có thể và giai đoạn thứ hai tìm thấy bán kính tốt nhất cho mỗi trung tâm ứng cử viên

2.3.4.2. Hoạt động

- Tải 1 ảnh và làm mờ nó, giảm tiếng ồn
- Áp dụng biến đổi Hough Circle Transform cho hình ảnh mờ
- Hiển thị vòng tròn được phát hiện

Đầu tiên chuyển đổi thành thang độ xám, áp dụng medianBlur để giảm nhiễu và phát hiện đường tròn giả

```
src = cv.imread(f)
gray = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
gray = cv.medianBlur(gray, 5)
rows = gray.shape[0]
```

Sau đó áp dụng Hough Circle Transform

```
circles = cv.HoughCircles(gray, cv.HOUGH_GRADIENT, 1, rows / 8,
param1=100, param2=30, minRadius=0, maxRadius=0)
```

```
if circles is not None:
```

```
    circles = np.uint16(np.around(circles))
```

```
    for i in circles[0, :]:
```

```
        center = (i[0], i[1])
```

```
        # circle center
```

```
        cv.circle(src, center, 1, (0, 100, 100), 3)
```

```
        # circle outline
```

```
radius = i[2]
cv.circle(src, center, radius, (255, 0, 255), 3)
```

Cuối cùng hiển thị ảnh gốc và các hình tròn phát hiện

```
cv.imwrite("done_circle.jpg",src)
im4 = Image.open("done_circle.jpg")
im4 = ImageTk.PhotoImage(im4)
global label3
label3 = Label(image = im4)
label3.image = im4
label3.place(x=720,y=250)
```

2.3.4.3. *Hough circle: Cách phát hiện các hình tròn bằng OpenCV*

Trong trường hợp biến đổi Hough dòng, chúng tôi yêu cầu hai tham số, (θ, ρ) nhưng để phát hiện các vòng tròn, chúng tôi yêu cầu ba tham số

- (x, y) tọa độ tâm của đường tròn.
- bán kính.

Như bạn có thể tưởng tượng, một máy dò vòng tròn sẽ yêu cầu bộ tích lũy 3D - một cho mỗi tham số.

Phương trình của một đường tròn được cho bởi

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \quad (2)$$

- Các bước sau đây được thực hiện để phát hiện các vòng tròn trong một hình ảnh:

1. Tìm các cạnh trong ảnh đã cho với sự trợ giúp của các máy dò cạnh (Canny).
2. Để phát hiện các vòng tròn trong một hình ảnh, chúng tôi đặt ngưỡng cho giá trị tối đa và tối thiểu của bán kính.
3. Bảng chứng được thu thập trong một mảng tích lũy 3D cho sự hiện diện của các vòng tròn với các trung tâm và bán kính khác nhau.

Hàm **HoughCircles** được sử dụng trong OpenCV để phát hiện các vòng tròn trong ảnh. Nó nhận các tham số sau:

- **image:** Hình ảnh đầu vào.
- **method:** Phương pháp phát hiện.
- **dp:** tỷ lệ nghịch của độ phân giải tích lũy và độ phân giải hình ảnh.
- **mindst:** khoảng cách tối thiểu giữa các trung tâm od vòng tròn phát hiện.
- **param_1 and param_2:** Đây là các tham số cụ thể của phương thức.
- **min_Radius:** bán kính tối thiểu của vòng tròn được phát hiện.
- **max_Radius:** bán kính tối đa được phát hiện.

Python:

```
import cv2
import numpy as np
#import sys

def onTrackbarChange(max_slider):
    cimg = np.copy(img)

    p1 = max_slider
    p2 = max_slider * 0.4

    #Phát hiện các vòng tròn bằng cách sử dụng biến đổi HoughCircles
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1,
cimg.shape[0]/64, param1=p1, param2=p2, minRadius=25, maxRadius=50)

    # Nếu phát hiện ít nhất 1 vòng tròn
    if circles is not None:
        cir_len = circles.shape[1] # chiều dài được tìm thấy
        circles = np.uint16(np.around(circles))
        for i in circles[0, :]:
            # Vẽ vòng tròn bên ngoài
            cv2.circle(cimg, (i[0], i[1]), i[2], (0, 255, 0), 2)
            # Vẽ tâm của vòng tròn
            cv2.circle(cimg, (i[0], i[1]), 2, (0, 0, 255), 3)
        else:
            cir_len = 0 # không phát hiện vòng tròn
#         print(str(cir_len))

    # Hiển thị hình ảnh đầu ra
    cv2.imshow('Image', cimg)

    # Cạnh hình ảnh để gỡ lỗi
    edges = cv2.Canny(gray, p1, p2)
    cv2.imshow('Edges', edges)

if __name__ == "__main__":
    # Read image
    img = cv2.imread('HoughCircles.jpg')

    # Chuyển đổi sang thang màu xám
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Tạo cửa sổ hiển thị
    cv2.namedWindow("Edges")
    cv2.namedWindow("Image")

    # Trackbar sẽ được sử dụng để thay đổi ngưỡng cho cạnh
    initThresh = 105
```

```

maxThresh = 200

# Tạo trackbar
cv2.createTrackbar("Threshold", "Image", initThresh, maxThresh,
onTrackbarChange)
onTrackbarChange(initThresh)

while True:
    key = cv2.waitKey(1)
    if key == 27:
        break
cv2.destroyAllWindows()

```

Hàm `HoughCircles` có sẵn khả năng phát hiện canny, do đó không bắt buộc phải phát hiện các cạnh rõ ràng trong nó. Kết quả phát hiện vòng tròn

Kết quả phát hiện vòng tròn sử dụng biến đổi Hough được hiển thị bên dưới. Chất lượng kết quả phụ thuộc rất nhiều vào chất lượng của các cạnh bạn có thể tìm thấy, và cũng phụ thuộc vào mức độ kiến thức bạn có về kích thước của vòng tròn bạn muốn phát hiện.



Hình 2.11: Phát hiện hình tròn bằng Hough



Hình 2.12: Phát hiện hình tròn bằng Hough Transform

2.3. Giới thiệu thư viện OpenCV

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở về thị giác máy tính và học máy. Thư viện được xây dựng để cung cấp nền tảng cho các ứng dụng thị giác máy tính nhằm đẩy mạnh sự phát triển về hàm lượng tri thức máy tính trong các sản phẩm thương mại. Nhờ giấy phép bản quyền BSD và được nhiều công ty lớn hàng đầu thế giới như Google, Yahoo, Microsoft, Intel, IBM,... cùng đóng góp xây dựng thư viện, OpenCV là một trong những công cụ mạnh và được sử dụng rộng rãi trong trường học cũng như các công ty khởi nghiệp.

Thư viện OpenCV bao gồm nhiều giao diện dành cho C++, C, Python, Java, MATLAB và hỗ trợ cho các hệ điều hành khác nhau như Windows, Linux, Android, MacOS. Trong phiên bản OpenCV 3.1, giao diện sử dụng cho CUDA và OpenCL cũng đã được phát triển hoàn thiện.

OpenCV được viết nguyên bản bằng ngôn ngữ C++.

Trang chính thức: <http://opencv.org>

Nhóm phát triển OpenCV: <http://itseez.com>

2.3.1. Lịch sử phát triển

OpenCV được bắt đầu tại Intel vào năm 1999 bởi Gary Bradsky và bản phát hành đầu tiên ra mắt vào năm 2000. Vadim Pisarevsky đã cùng Gary Bradsky quản lý nhóm OpenCV phần mềm Intel của Nga. Vào năm 2005, OpenCV đã được sử dụng trên Stanley, chiếc xe đã chiến thắng Thử thách lớn DARPA năm 2005. Sau đó, sự phát triển tích cực của nó tiếp tục dưới sự hỗ trợ của Willow Garage, với Gary Bradsky và Vadim Pisarevsky dẫn dắt dự án. Ngay bây giờ, OpenCV hỗ trợ rất nhiều thuật toán liên quan đến Thị giác máy tính và Học máy và nó đang mở rộng từng ngày.

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở về thị giác máy tính và học máy. Thư viện được xây dựng để cung cấp nền tảng cho các ứng dụng thị giác máy tính nhằm đẩy mạnh sự phát triển về hàm lượng tri thức máy tính trong các sản phẩm thương mại. Nhờ giấy phép bản quyền BSD và được nhiều công ty lớn hàng đầu thế giới như Google, Yahoo, Microsoft, Intel, IBM,... cùng đóng góp xây dựng thư viện, OpenCV là một trong những công cụ mạnh và được sử dụng rộng rãi trong trường học cũng như các công ty khởi nghiệp.

Thư viện OpenCV bao gồm nhiều giao diện dành cho C++, C, Python, Java, MATLAB và hỗ trợ cho các hệ điều hành khác nhau như Windows, Linux, Android, MacOS. Trong phiên bản OpenCV 3.1, giao diện sử dụng cho CUDA và OpenCL cũng đã được phát triển hoàn thiện. OpenCV được viết nguyên bản bằng ngôn ngữ C++.

2.3.2. Các phiên bản của thư viện OpenCV

- Bản 3.0 RC tháng 6 năm 2015.
- Bản 3.1 tháng 12 năm 2015.
- Bản 3.2 tháng 12 năm 2016.
- Bản 3.3 tháng 8 năm 2017.
- Bản 3.3.1 tháng 10 năm 2017.
- Bản 3.4 tháng 12 năm 2017.
- Bản 3.4.1 tháng 2 năm 2018.
- Bản 3.4.2 tháng 7 năm 2018

2.3.3. Phiên bản OpenCv nhóm đang sử dụng

Phiên bản OpenCV nhóm đang sử dụng là bản 3.4.2

2.3.4. Chức năng thư viện OpenCV

Image/video I/O, xử lý, hiển thị (core, imgproc, highgui) Phát hiện các vật thể (objdetect, features2d, nonfree) Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab) Computational photography (photo, video, superres) Machine learning & clustering (ml, flann) CUDA acceleration (gpu)

Nhiều thuật toán chụp ảnh tính toán (HDR, in mờ, bộ lọc nhận biết cạnh, siêu ảnh, siêu tốc); Thuật toán theo dõi và dòng quang; Mô tả dòng, KAZE/AKAZE; Tối ưu hóa sử dụng chung (leo đồi, lập trình tuyến tính).; Mô đun khớp hình dạng 2D, mô đun khớp bề mặt 3D; Mô đun RGB-D.

2.3.5. Sơ lược về cấu trúc của thư viện OpenCV

Thư viện OpenCV có thể được chia thành 2 phần (module) chính:

Phần căn bản (basic) là mã nguồn được nhóm phát triển xây dựng và kiểm toàn diện, gồm các thuật toán đã được thế giới công nhận và đánh giá dựa trên cơ sở lý thuyết chắc chắn.

Phần mở rộng (contribution) được nhiều tổ chức khoa học khác nhau trên thế giới đóng góp, gồm nhiều thuật toán cập nhật được xây dựng dựa trên các công trình nghiên cứu.

Do vậy các thuật toán trong phần mở rộng có độ ổn định và tối ưu không cao. Từ phiên bản 3.0 phần mở rộng được tách riêng không còn được gộp chung với thư viện mặc định.

OpenCV là thư viện mã nguồn mở được đóng gói thành tập tin nén. Tùy vào phiên bản dành cho các hệ điều hành khác nhau mà tập tin nén này có định dạng tương ứng. Thư viện OpenCV cung cấp cho người dùng các cấu trúc dữ liệu, đối tượng và hàm bằng cách khai báo **nguyên mẫu (prototype)**

của chúng trong các tập tin thư viện C/C++ (*.h, *.hpp,...) và định nghĩa chi tiết trong các tập tin mã nguồn (*.c, *.cpp **hoặc** *.py).

Thư viện mã nguồn mở OpenCV gồm các thành phần nhỏ sau đây (tính đến phiên bản 3.1.0):

- Các thành phần chính:

- o core. Các hàm cơ bản
- o imgproc. Xử lý ảnh
- o imgcodecs. Đọc và ghi tập tin ảnh
- o videoio. Đọc và ghi đa phương tiện
- o highgui. Giao diện người dùng bậc cao
- o video. Phân tích video
- o calib3d. Hiệu chuẩn thiết bị ghi hình và tái cấu trúc 3D
- o features2d. Bộ khung các đặc trưng 2D
- o objdetect. Nhận dạng đối tượng
- o ml. Học máy
- o flann. Gom nhóm và tìm kiếm trong nhiều chiều
- o photo. Các thuật toán chụp ảnh
- o stitching. Vá ảnh
- o cudaarithm. Các toán tử trên ma trận
- o cudabgsegm. Phân đoạn nền ảnh
- o cudacodec. Mã hóa/giải mã video
- o cudafeatures2d. Phát hiện và mô tả đặc trưng
- o cudafilters. Lọc ảnh
- o cudaimgproc. Xử lý ảnh với CUDA
- o cudalegacy. Hỗ trợ Legacy
- o cudaobjdetect. Phát hiện đối tượng
- o cudaoptflow. Dòng quang học
- o cudastereo. Thư viện âm thanh
- o cudawarping. Bẻ cong ảnh
- o cudev. Tầng thiết bị
- o shape. Khác biệt hình học và so sánh
- o superres. Siêu phân giải
- o videostab. Ổn định video
- o viz. Hiển thị 3D

- Các thành phần mở rộng:

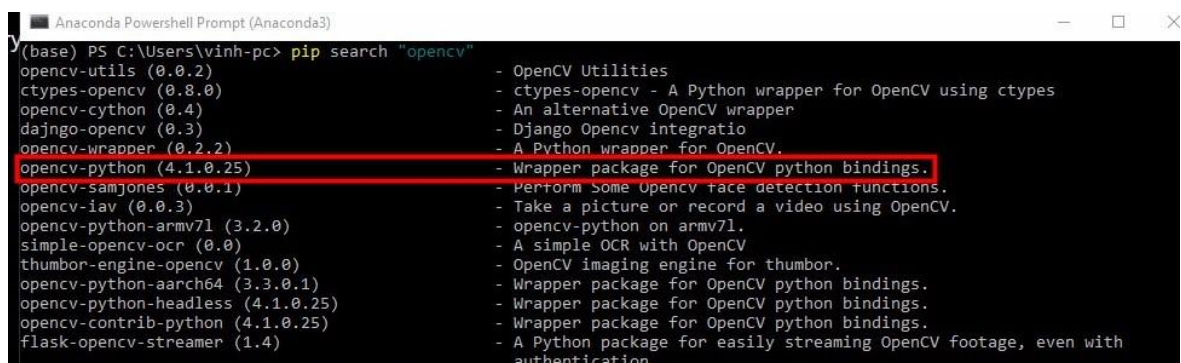
- o aruco. Phát hiện ArUco Marker
- o bgsegm. Các phương pháp phân đoạn đối tượng mới
- o bioinspired. Các mô hình thị giác sinh học và các công cụ phái sinh
- o ccalib. Tùy chọn hiệu chuẩn mẫu cho tái cấu trúc 3D

- o cvv. GUI cho thao tác bắt lỗi trực quan của các chương trình thị giác máy tính
- o datasets. Thư viện hỗ trợ làm việc với các bộ dữ liệu khác nhau
- o dnn. Bộ thư viện mạng neural học sâu
- o dpm. Các mô hình biến dạng từng phần
- o face. Nhận dạng mặt người
- o fuzzy. Xử lý ảnh dựa vào lý thuyết mờ
- o hdf. Nhập xuất dữ liệu cho định dạng HDF (Hierarchical Data Format)
- o line_descriptor. Mô tả nhị phân cho các đường thẳng trích chọn từ một ảnh
- o matlab. Liên kết với MATLAB
- o optflow. Các thuật toán dòng quan học
- o plot. Hàm vẽ cho dữ liệu ma trận
- o reg. Đăng kí ảnh
- o rgbd. Xử lý chiều sâu trong hệ màu RGB
- o saliency. API xử lý vùng lỗi trên ảnh
- o sfm. Cấu trúc từ chuyển động
- o stereo. Các thuật toán tính hiệu nổi tương tự
- o structured_light. API cho tính cấu trúc của ánh sáng
- o surface_matching. Khớp bề mặt
- o text. Phát hiện và nhận dạng chữ trong cảnh tự nhiên
- o tracking. Các phương pháp theo vết
- o xfeatures2d. Tính năng đặc trưng 2D mở rộng
- o ximgproc. Xử lý ảnh mở rộng
- o xobjdetect. Phát hiện đối tượng mở rộng
- o xphoto. Các phương pháp xử lý hình ảnh nâng cao

2.3.6. Cài đặt thư viện OpenCV

Pip là hệ thống quản lý các package của Python. Mình có thể tìm kiếm các package mà pip hỗ trợ thông qua lệnh **pip search**.

pip search "opencv"

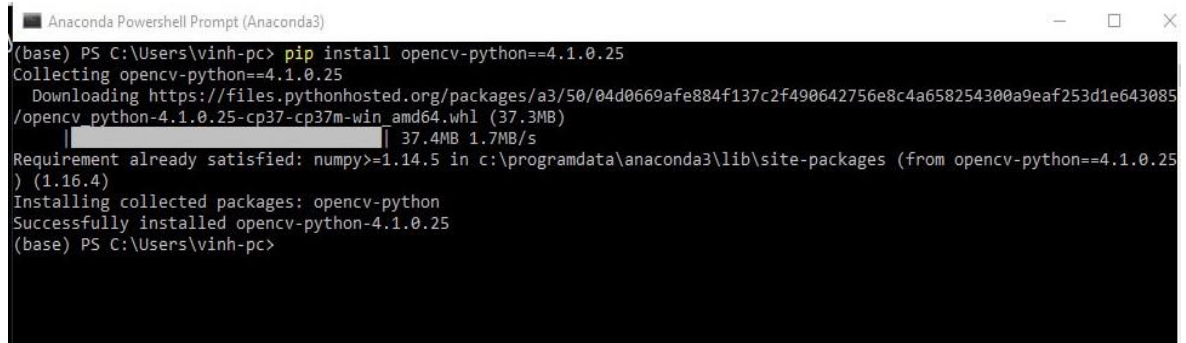


```

Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\vinh-pc> pip search "opencv"
opencv-utils (0.0.2) - OpenCV Utilities
ctypes-opencv (0.8.0) - ctypes-opencv - A Python wrapper for OpenCV using ctypes
opencv-cython (0.4) - An alternative OpenCV wrapper
django-opencv (0.3) - Django Opencv integratio
opencv-wrapper (0.2.2) - A Python wrapper for OpenCV.
opencv-python (4.1.0.25) - Wrapper package for OpenCV python bindings.
opencv-samjones (0.0.1) - Perform some opencv face detection functions.
opencv-iav (0.0.3) - Take a picture or record a video using OpenCV.
opencv-python-armv7l (3.2.0) - opencv-python on armv7l.
simple-opencv-ocr (0.0) - A simple OCR with OpenCV
thumbor-engine-opencv (1.0.0) - OpenCV imaging engine for thumbor.
opencv-python-aarch64 (3.3.0.1) - Wrapper package for OpenCV python bindings.
opencv-python-headless (4.1.0.25) - Wrapper package for OpenCV python bindings.
opencv-contrib-python (4.1.0.25) - Wrapper package for OpenCV python bindings.
flask-opencv-streamer (1.4) - A Python package for easily streaming OpenCV footage, even with authentication
  
```

Như hình trên mình thấy, pip hiện tại hỗ trợ package **opencv-python** (phiên bản 4.1.0.25), nên mình có thể cài đặt OpenCV cho Python thông qua pip. Sử dụng lệnh **pip install** để cài đặt OpenCV.

`pip install opencv-python==3.4.2.16`



```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\vinh-pc> pip install opencv-python==4.1.0.25
Collecting opencv-python==4.1.0.25
  Downloading https://files.pythonhosted.org/packages/a3/50/04d0669afe884f137c2f490642756e8c4a658254300a9eaf253d1e643085
/opencv_python-4.1.0.25-cp37-cp37m-win_amd64.whl (37.3MB)
    | 37.4MB 1.7MB/s
Requirement already satisfied: numpy>=1.14.5 in c:\programdata\anaconda3\lib\site-packages (from opencv-python==4.1.0.25)
(1.16.4)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.1.0.25
(base) PS C:\Users\vinh-pc>
```

Kiểm tra lại, nếu có thể import được OpenCV mà không báo lỗi thì việc cài đặt thành công.

CHƯƠNG 3: ỨNG DỤNG PHÁT HIỆN ĐOẠN THẲNG VÀ HÌNH TRÒN BẰNG PHÉP BIẾN ĐỔI HOUGH

3.1. Bài Toán

Ngày nay các thiết bị sử dụng thuật toán xử lý ảnh được sử dụng ngày càng rộng rãi, với nhiều mục đích khác nhau. Bên cạnh đó, cũng đã có rất nhiều nghiên cứu được đưa ra về vấn đề phát hiện hiện đường thẳng và đường tròn bằng phép biến đổi Hough. Chương trình ứng dụng phát hiện đường thẳng và đường tròn bằng phép biến đổi Hough:

Với một ảnh đầu vào, bài toán đặt ra là làm thế nào để tìm được các đặc trưng bất biến của ảnh dựa vào biên của hình dạng. Khi tìm được biên của đối tượng trong ảnh thì trích chọn được các đặc trưng bất biến của hình dạng như các đường thẳng, hình tròn. Từ đó ta đặt ra giải pháp như sau: sử dụng các phép toán để tìm biên của ảnh (canny, sobel...), sử dụng phép biến đổi hough cho đường thẳng và biến đổi hough cho hình tròn để trích chọn các đặc trưng của đối tượng như trích chọn được các đường thẳng, hình tròn bình chọn tìm thấy trong đối tượng của hình ảnh. Cuối cùng ta sẽ trích chọn đặc trưng của đối tượng đáp ứng được yêu cầu bài toán đặt ra.

Đầu vào: là một ảnh bất kỳ

Đầu ra: ảnh đã được trích chọn các đặc trưng.

3.2. Giao diện chức năng chương trình

Chương trình được xây dựng bằng spyder (bản 3.6), để minh họa các phép toán như: phép biến đổi hough cho đường thẳng, phép biến đổi hough cho hình tròn.



Hình 4.1: Giao diện chính

- Nút *Phát hiện đường thẳng*: dùng để xử lý bằng phép biến đổi Hough đường thẳng
- Nút *Phát hiện đường tròn*: dùng để xử lý bằng phép biến đổi Hough đường tròn

3.3. Một số kết quả chương trình.



Hình 4.2: Kết quả thực hiện phát hiện đoạn thẳng



Hình 4.3: Kết quả thực hiện tìm hình tròn bằng phép biến đổi Hough

KẾT LUẬN

Kết quả đạt được:

- Tìm hiểu được các thư viện trong xử lý ảnh.
- Minh họa giải thích được Hough Transform.
- Phần Demo ứng dụng đã phát hiện diện các đường thẳng và hình tròn trong ảnh.

Bên cạnh những kết quả đạt được thì cũng còn nhiều hạn chế:

Trong quá trình tìm hiểu đề tài do khả năng có hạn nên không tránh khỏi những thiếu sót trong bản báo cáo, và thuật toán tìm kiếm. Nhóm em rất mong được sự góp ý và chia sẻ của các thầy cũng như các bạn để bài báo cáo của em có thể hoàn thiện hơn.

TÀI LIỆU THAM KHẢO

Tiếng Việt

- [1]. TS. Đỗ Năng Toàn, TS. Phạm Việt Bình (2007) - Giáo Trình Môn Học Xử Lý Ảnh, trường ĐH thái nguyên, khoa CNTT
- [2]. PGS. Nguyễn Quang Hoan (2006)- Giáo Trình Xử Lý Ảnh, học viện công nghệ bưu chính viễn thông
- [3]. Lương Mạnh Bá, Nguyễn Thanh Thủy (2003)- Nhập Môn Xử Lý Ảnh Số, Nhà xuất bản Khoa học và Kỹ thuật

Website

- [1]. OpenCV, Open Source Computer Vision, Hough Line Transform
https://docs.opencv.org/3.4.0/d9/db0/tutorial_hough_lines.html
- [2].]. OpenCV, Open Source Computer Vision, Hough Circle Transform
https://docs.opencv.org/3.4.0/d4/d70/tutorial_hough_circle.html