**Bài 23: Computer Vision - Lập trình AI bằng Python**

Thị giác máy tính liên quan đến việc mô hình hóa và tái tạo tầm nhìn của con người bằng phần mềm và phần cứng máy tính. Trong chương này, bạn sẽ tìm hiểu chi tiết về điều này.

**1. Thị giác máy tính :**

Thị giác máy tính là ngành học nghiên cứu cách tái tạo, ngắt và hiểu một cảnh 3D từ các hình ảnh 2D của nó, xét về các đặc tính của cấu trúc có trong cảnh đó.

**Hệ thống phân cấp thị giác máy tính**

Thị giác máy tính được chia thành ba loại cơ bản như sau

* Low-level vision : Nó bao gồm hình ảnh quy trình để trích xuất tính năng.
* Intermediate-level vision : Nó bao gồm nhận dạng đối tượng và giải thích cảnh 3D
* High-level vision : Nó bao gồm mô tả khái niệm về một cảnh như hoạt động, ý định và hành vi.

**2. Thị giác máy tính và xử lý ảnh :**

Xử lý ảnh nghiên cứu sự chuyển đổi ảnh sang ảnh. Đầu vào và đầu ra của quá trình xử lý hình ảnh đều là hình ảnh.

Thị giác máy tính là việc xây dựng các mô tả rõ ràng, có ý nghĩa về các đối tượng vật lý từ hình ảnh của chúng. Đầu ra của thị giác máy tính là mô tả hoặc diễn giải các cấu trúc trong cảnh 3D.

**Applications :**

Các ứng dụng chính của thị giác máy tính :

**Robotics :**

* Localization-determine robot location automatically
* Dẫn đường
* Tránh chướng ngại vật
* Lắp ráp (peg-in-hole, welding, painting)
* Manipulation (e.g. PUMA robot manipulator)
* Tương tác với người máy (HRI): Robot thông minh để tương tác và phục vụ con người

**Y tế :**

* Classification và detection (vd : phân loại tổn thương hoặc tế bào và phát hiện khối u)
* Phân đoạn 2D / 3D
* Tái tạo nội tạng người 3D (MRI hoặc siêu âm)
* Phẫu thuật robot có hướng dẫn thị giác

**Bảo mật :**

* Sinh trắc học (mống mắt, vân tay, nhận dạng khuôn mặt)
* Giám sát - phát hiện một số hoạt động hoặc hành vi đáng ngờ

**Vận chuyển**

* Xe tự hành
* An toàn(ví dụ, giám sát cảnh giác của người lái xe)

**Ứng dụng tự động hóa công nghiệp**

* Kiểm tra công nghiệp (phát hiện khuyết tật)
* Assembly
* Đọc mã vạch và nhãn gói
* Phân loại đối tượng
* Document understanding (e.g. OCR)

**3. Cài đặt thư viện hỗ trợ :**

Đối với Thị giác máy tính với Python, bạn có thể sử dụng một thư viện phổ biến có tên là OpenCV (Thị giác máy tính nguồn mở). Nó là một thư viện các chức năng lập trình chủ yếu nhắm vào thị giác máy tính thời gian thực. Nó được viết bằng C ++ và giao diện chính của nó là C ++. Bạn có thể cài đặt gói này với sự trợ giúp của lệnh sau:

**pip** **install** **opencv\_python-X**.X-cp36-cp36m-winX.whl

Ở đây X đại diện cho phiên bản Python được cài đặt trên máy của bạn cũng như win32 hoặc 64 bit mà bạn đang có.

Nếu bạn đang sử dụng môi trường anaconda, hãy sử dụng lệnh sau để cài đặt OpenCV :

conda install -c conda-forge opencv

**4. Đọc, Viết và Hiển thị Hình ảnh**

Hầu hết các ứng dụng CV cần lấy hình ảnh làm đầu vào và sản xuất hình ảnh làm đầu ra. Trong phần này, bạn sẽ học cách đọc và ghi tệp hình ảnh với sự trợ giúp của các chức năng do OpenCV cung cấp.

**Các chức năng OpenCV**

* **Hàm imread ()** - Đây là hàm để đọc một hình ảnh. OpenCV imread () hỗ trợ nhiều định dạng hình ảnh khác nhau như PNG, JPEG, JPG, TIFF, v.v.
* **Hàm imshow ()**- Đây là hàm để hiển thị hình ảnh trong cửa sổ. Cửa sổ tự động phù hợp với kích thước hình ảnh. OpenCV imshow () hỗ trợ nhiều định dạng hình ảnh khác nhau như PNG, JPEG, JPG, TIFF, v.v.
* **Hàm imwrite ()** - Đây là hàm để viết một hình ảnh. OpenCV imwrite () hỗ trợ nhiều định dạng hình ảnh khác nhau như PNG, JPEG, JPG, TIFF, v.v.

**Ví dụ :**

Ví dụ này sử dụng đoạn code python để đọc một hình ảnh ở một định dạng - hiển thị nó trong một cửa sổ và viết cùng một hình ảnh ở định dạng khác như sau :

**import** cv2

Đọc một hình ảnh cụ thể, sử dụng hàm imread ()

image = cv2.imread('image\_flower.jpg')

Để hiển thị hình ảnh, hãy sử dụng hàm imshow (). Tên của cửa sổ mà bạn có thể thấy hình ảnh sẽ là image\_flower.



Ta có thể viết cùng một hình ảnh sang định dạng khác, chẳng hạn .png bằng cách sử dụng hàm imwrite () -

**cv2**.imwrite('image\_flower.png',image)

Kết quả True có nghĩa là hình ảnh đã được ghi thành công dưới dạng tệp .png cũng trong cùng một thư mục.

True

Lưu ý - Hàm DestallWindows () chỉ đơn giản là hủy tất cả các cửa sổ chúng ta đã tạo.

**5. Chuyển đổi không gian màu :**

Trong OpenCV, hình ảnh không được lưu trữ bằng cách sử dụng màu RGB thông thường, thay vào đó chúng được lưu trữ theo thứ tự ngược lại, tức là theo thứ tự BGR. Do đó, mã màu mặc định trong khi đọc ảnh là BGR. Chức năng chuyển đổi màu cvtColor () để chuyển đổi hình ảnh từ mã màu này sang mã màu khác

**Ví dụ :**

Hãy xem xét ví dụ này để chuyển đổi hình ảnh từ BGR sang thang độ xám.

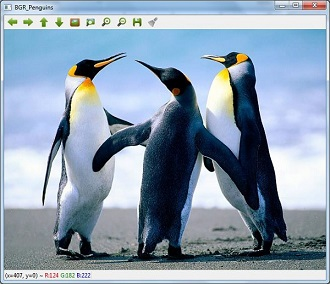
**import** cv2

Để đọc một hình ảnh cụ thể, hãy sử dụng hàm imread () -

image = cv2.imread('image\_flower.jpg')

nếu ta thấy hình ảnh này bằng cách sử dụng hàm imshow (), thì ta có thể thấy rằng hình ảnh này nằm trong BGR.

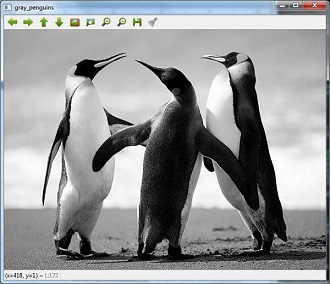
**cv2**.imshow('BGR\_Penguins',image)



Sử dụng hàm cvtColor () để chuyển đổi hình ảnh này sang thang độ xám.

**image** = cv2.cvtColor(image,cv2.COLOR\_BGR2GRAY)

cv2.imshow('gray\_penguins',image)



**6. Edge Detection**

Con người, sau khi xem một bản phác thảo thô, có thể dễ dàng nhận ra nhiều loại vật thể và tư thế của chúng. Đó là lý do tại sao các cạnh đóng một vai trò quan trọng trong cuộc sống của con người cũng như trong các ứng dụng của thị giác máy tính. OpenCV cung cấp hàm rất đơn giản và hữu ích gọi là Canny () để phát hiện các cạnh.

**Ví dụ :**

Ví dụ sau đây cho thấy sự xác định rõ ràng của các cạnh.

**import** cv2

**import** numpy as np

Để đọc một hình ảnh cụ thể, hãy sử dụng hàm imread ().

image = cv2.imread('Penguins.jpg')

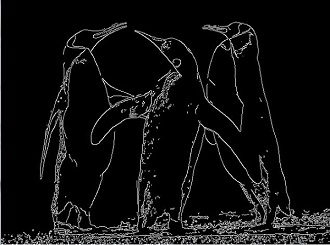
Sử dụng hàm Canny () để phát hiện các cạnh của hình ảnh đã được đọc.

**cv2**.imwrite(‘**edges\_Penguins**.jpg’,**cv2**.Canny(**image**,200,300))

Hiển thị hình ảnh có các cạnh, hãy sử dụng hàm imshow ().

**cv2**.imshow(‘**edges**’, **cv2**.imread(‘‘**edges\_Penguins**.jpg’))

Kết quả như sau :



**7. Face Detection**

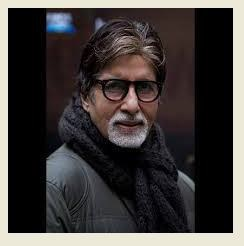
Nhận diện khuôn mặt là một trong những ứng dụng hấp dẫn của thị giác máy tính, nó làm cho nó trở nên thực tế hơn cũng như tương lai. OpenCV có một cơ sở tích hợp để thực hiện nhận diện khuôn mặt. Chúng tôi sẽ sử dụng bộ phân loại tầng Haar để nhận diện khuôn mặt.

**Dữ liệu tầng Haar**

Ta cần dữ liệu để sử dụng bộ phân loại tầng Haar. Bạn có thể tìm thấy dữ liệu này trong gói OpenCV . Sau khi cài đặt OpenCv, bạn có thể thấy tên thư mục haarcascades. Sẽ có các tệp .xml cho các ứng dụng khác nhau. Bây giờ, sao chép tất cả chúng để sử dụng khác nhau và dán vào một thư mục mới trong project hiện tại.

**Ví dụ :**

Sau đây là mã Python sử dụng Haar Cascade để phát hiện khuôn mặt của Amitabh Bachan được hiển thị trong hình ảnh sau



**import** cv2

**import** numpy as np

Sử dụng **HaarCascadeClassifier**để nhận diện khuôn mặt

face\_detection=

**cv2.CascadeClassifier('D:/ProgramData/cascadeclassifier/**

haarcascade\_frontalface\_default.xml')

Đọc một hình ảnh cụ thể, hãy sử dụng hàm imread ()

img = cv2.imread('AB.jpg')

chuyển nó thành thang độ xám vì nó sẽ chấp nhận hình ảnh màu xám :

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

Sử dụng **face\_detection.detectMultiScale ,**thực hiện nhận diện khuôn mặt thực tế

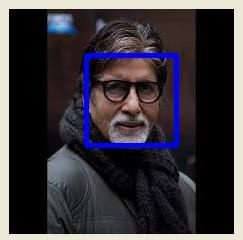
faces = face\_detection.detectMultiScale(gray, 1.3, 5)

vẽ một hình chữ nhật xung quanh toàn bộ khuôn mặt -

**for** (x,y,w,h) in faces:

img = cv2.rectangle(img,(x,y),(x+w, y+h),(255,0,0),3)

cv2.imwrite('Face\_AB.jpg',img)



**8. Eye Detection :**

Phát hiện mắt là một ứng dụng hấp dẫn khác của thị giác máy tính giúp nó trở nên thực tế hơn cũng như tương lai. OpenCV có một cơ sở tích hợp để thực hiện phát hiện mắt. Chúng tôi sẽ sử dụng bộ phân loại tầng Haar để phát hiện mắt.

**Ví dụ :**

Đoạn code sau sử dụng Haar Cascade để phát hiện khuôn mặt của Amitabh Bachan :



**import** cv2

**import** numpy as np

Sử dụng **HaarCascadeClassifier**

eye\_cascade = cv2.CascadeClassifier('D:/ProgramData/cascadeclassifier/haarcascade\_eye.xml')

Để đọc một hình ảnh cụ thể, hãy sử dụng hàm imread ()

img = cv2.imread('AB\_Eye.jpg')

chuyển nó thành thang độ xám

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

Giờ đây với sự trợ giúp của eye\_cascade.detectMultiScale, ta tiến hàng nhận diện khuôn mặt thực tế

eyes = eye\_cascade.detectMultiScale(gray, 1.03, 5)

vẽ một hình chữ nhật xung quanh toàn bộ khuôn mặt

**for** (ex,ey,ew,eh) in eyes:

img = cv2.rectangle(img,(ex,ey),(ex+ew, ey+eh),(0,255,0),2)

cv2.imwrite('Eye\_AB.jpg',img)

