**TÌM HIỂU THƯ VIỆN DLIB**

1. Định nghĩa

* Dlib là một thư viện mã nguồn mở được sử dụng để giải quyết các vấn đề liên quan đến computer vision, machine learning và các ứng dụng liên quan đến các lĩnh vực này. Thư viện này cung cấp các công cụ và thuật toán để xây dựng các hệ thống nhận diện khuôn mặt, phát hiện vật thể, theo dõi đối tượng, phân loại hình ảnh và nhiều ứng dụng khác. Dlib được viết bằng ngôn ngữ C++ và có sẵn cho các nền tảng Windows, MacOS và Linux.
* Dlib là một chương trình của thư viện OpenCV, hỗ trợ người dùng trong việc xác định khuôn mặt.
* Thư viện dlib được viết bằng ngôn ngữ C++ do Davis King tạo ra vào năm 2012. Dlib được sử dụng nhiều trong lĩnh vực Computer Vision, đặc biệt là trong phát hiện vật thể, nhận dạng khuôn mặt.

1. Các tính năng chính của Dlib:

* Phát hiện khuôn mặt: Dlib cung cấp thuật toán phát hiện khuôn mặt dựa trên HOG và SVM
* Trích xuất đặc trưng khuôn mặt: Dlib cung cấp bộ trích xuất đặc trưng khuôn mặt dựa trên CNNs, cho phép xác định các điểm landmark trên khuôn mặt
* Nhận diện khuôn mặt: Dlib cung cấp các thuật toán nhận diện khuôn mặt, bao gồm nhận diện và phân loại khuôn mặt dựa trên SVM và deep learning
* Theo dõi đối tượng: Dlib cung cấp các thuật toán theo dõi đối tượng trong video, bao gồm thuật toán Median Flow và thuật toán Kernelized Correlation Filters (KCF)
* Xây dựng hệ thống nhận diện vật thể: Dlib cung cấp các công cụ để xây dựng các hệ thống nhận diện vật thể, bao gồm thuật toán HOG và SVM
* Xử lý ảnh: Dlib cung cấp các công cụ để xử lý ảnh, bao gồm xử lý ma trận, tích chập, tăng mẫu và giảm mẫu.
* Tối ưu hóa: Dlib cung cấp các thuật toán tối ưu hóa, bao gồm Gradient Descent, Conjugate Gradient, Quasi-Newton Methods và Broyden-Fletcher-Goldfarb-Shanno (BFGS)
* Hỗ trợ nhiều ngôn ngữ lập trình: Dlib được viết bằng ngôn ngữ C++ và có sẵn cho các nền tảng Windows, MacOS và Linux. Ngoài ra, Dlib còn hỗ trợ Python, Java và C#
* Linh hoạt: Dlib cho phép người dùng tùy chỉnh và mở rộng các thuật toán và công cụ theo nhu cầu của mình.

1. Các hàm của dlib

Dlib cung cấp rất nhiều hàm để giải quyết các vấn đề liên quan đến computer vision, machine learning và các ứng dụng liên quan đến các lĩnh vực này.

* Hàm phát hiện khuôn mặt: dlib.get\_frontal\_face\_detector()
* Hàm tạo bộ trích xuất đặc trưng khuôn mặt: dlib.face\_recognition\_model\_v()
* Hàm tạo bộ trích xuất đặc trưng HOG (Histogram of Oriented Gradients): dlib.get\_frontal\_face\_detector()
* Hàm huấn luyện SVM (Support Vector Machine): dlib.train\_simple\_linear\_regression()
* Hàm tạo chuỗi hash SHA-256: dlib::sha256()
* Hàm tính toán khoảng cách Euclid: dlib::length()
* Hàm xử lý ma trận
* Hàm tối ưu hóa
* Hàm tương tác với file
* Hàm xử lý dữ liệu
* Hàm liên quan đến deep learning

1. Ưu, nhược điểm:
2. Ưu điểm:

* Tốc độ xử lý nhanh và hiệu quả
* Cung cấp nhiều thuật toán và công cụ hỗ trợ cho machine learning và computer vision
* Được viết bằng ngôn ngữ C++ => hoạt động mượt và ổn định
* Hỗ trợ tính năng xây dựng các hệ thống nhận diện khuôn mặt, phát hiện vật thể, theo dõi đối tượng, phân loại hình ảnh và nhiều ứng dụng khác.

1. Nhược điểm

* Khó tiếp cận đối với người mới bắt đầu vì cú pháp và hướng dẫn sử dụng không được rõ ràng.
* Không hỗ trợ nhiều cho deep learning so với các thư viện khác như TensorFlow hay PyTorch.
* Phạm vi ứng dụng hạn chế hơn so với các thư viện khác

1. Hàm phát hiện khuôn mặt của dlib

* Hàm phát hiện khuôn mặt của dlib là get\_frontal\_face\_detecor()
* Đây là một hàm được sử dụng để tạo ra một object detector (bộ phát hiện vật thể) cho việc phát hiện khuôn mặt.
* Bộ phát hiện này được xây dựng trên cơ sở thuật toán Viola-Jones và Histogram of Oriented Gradients.
* Để sử dụng hàm phát hiện khuôn mặt của Dlib cần import thư viện Dlib và khởi tạo bộ phát hiện:

Import dlib

detector = dlib.get\_frontal\_face\_detector()

* Áp dụng bộ phát hiện này để phát hiện khuôn mặt trong ảnh bằng cách sử dụng phương thức detect():

detections = detector(image, upsample\_num\_times=0)

+ image là ảnh đầu vào

+ upsample\_num\_times là số lần tăng mẫu, với giá trị mặc định là 0.

* Kết quả trả về là một danh sách các hình chữ nhật bao quanh các khuôn mặt được phát hiện trong ảnh.

1. Xác định facial landmark với dlib
2. Facial landmark là gì?

* Xác định được những điểm chính trong bức ảnh tạo nên hình dạng khuôn mặt người.
* Facial landmark là đầu vào cho nhiều bài toán khác như dự đoán tư thế đầu, tráo đổi khuôn mặt, phát hiện nháy mắt, xoay chỉnh lại khuôn mặt và điển hình là công nghệ nhận dạng khuôn mặt.
* Việc xác định facial landmark gồm 2 bước:

+ B1: Xác định được vị trí khuôn mặt trong bức ảnh

+ B2: Xác định được các điểm tạo nên cấu trúc của khuôn mặt

1. Cách sử dụng

* Để sử dụng file shape\_predictor\_68\_face\_landmarks.dat và thực hiện phát hiện khuôn mặt trong ảnh cần import các thư viện:

import dlib

import cv2

import numpy as np

* Sau đó, khởi tạo bộ phát hiện khuôn mặt và bộ trích xuất đặc trưng khuôn mặt:

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat")

+ detector là bộ phát hiện khuôn mặt

+ predictor là bộ trích xuất đặc trưng khuôn mặt được xây dựng trên file shape\_predictor\_68\_face\_landmarks.dat

* Ví dụ phát hiện khuôn mặt và các điểm landmark trên khuôn mặt trong ảnh image.jpg:

# Load ảnh

image = cv2.imread("image.jpg")

# Chuyển đổi sang grayscale

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

# Phát hiện khuôn mặt

faces = detector(gray)

# Lấy ra các điểm landmark trên từng khuôn mặt được phát hiện

for face in faces:

landmarks = predictor(gray, face)

landmarks = np.array([[p.x, p.y] for p in landmarks.parts()])

# Vẽ các điểm landmark lên ảnh

for (x, y) in landmarks:

cv2.circle(image, (x, y), 1, (0, 255, 0), -1)

# Hiển thị ảnh chứa các khuôn mặt và điểm landmark

cv2.imshow("Output", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Kết quả sẽ là một ảnh chứa các khuôn mặt được phát hiện và các điểm landmark trên từng khuôn mặt.

1. Sử dụng thư viện dlib để phát hiện khuôn mặt:

* B1: Cài đặt dlib trên máy tính (có thể thông qua pip).
* B2: Tải ảnh chứa khuôn mặt cần phát hiện lên.
* B3: Sử dụng hàm dlib.get\_frontal\_face\_detecor() để tạo ra một đối tượng phát hiện khuôn mặt
* B4: Chuyển đổi ảnh đầu vào sang dạng mảng numpy.
* B5: Sử dụng hàm phát hiện khuôn mặt của dlib để tìm kiếm các khuôn mặt trong ảnh đầu vào.
* B6: Vẽ hình chữ nhật xung quanh các khuôn mặt phát hiện được.

Ví dụ (code bằng Python):

import dlib

import cv2

# Tạo đối tượng phát hiện khuôn mặt

detector = dlib.get\_frontal\_face\_detector()

# Tải ảnh chứa khuôn mặt cần phát hiện lên

image = cv2.imread('path/to/image.jpg')

# Chuyển đổi ảnh đầu vào sang dạng mảng numpy

rgb\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

# Phát hiện khuôn mặt trong ảnh đầu vào

detections = detector(rgb\_image)

# Vẽ hình chữ nhật xung quanh các khuôn mặt phát hiện được

for face in detections:

x1 = face.left()

y1 = face.top()

x2 = face.right()

y2 = face.bottom()

cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)

# Hiển thị ảnh đầu ra

cv2.imshow('Output', image)

cv2.waitKey(0)

cv2.destroyAllWindows()