BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**Bài Báo Cáo Cuối Kì**

**MÔN HỌC: XỬ LÍ ẢNH**

**Giảng viên hướng dẫn: Nguyễn Hải Triều**

**Sinh viên thực hiện: 64132805 – Nguyễn Nhật Tú**

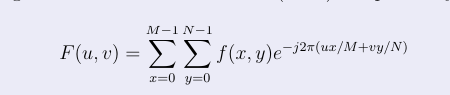
**Lớp: 64CNTT-2**

**Sinh Viên Trình Bày Lý Thuyết Xử Lí Ảnh Và Ứng Dụng Object Tracking In Video Trong Thực Tế**

**I.Lý Thuyết Xử Lí Ảnh :**

**Câu 1. Trình bày lý thuyết của Fast Fourier Transform (FFT):**

Từ công thức biến đổi Fourier rời rạc (DFT):



Ta nhận thấy độ phức tạp tính toán của biến đổi DFT/IDFT rất lớn: O(n)= n4.

Vậy nên chúng ta bắt buộc phải đơn giản tính toán cải thiện khả năng tính toán.

Do tính chất sóng, 1 hàm trong không gian 2 chiều có chu kì có thể biểu diễn thành tổng các hàm 1 chiều (sin/cosin). Vậy nên Công thức **2-D ở trên có thể được tách thành các 1-D DFT** như sau:

A math equations on a white background

Description automatically generated

ĐGH-2

ĐGH-1

Biến đổi 2-D DFT của hàm f(x,y) có thể đơn giản hoá biến đổi DFT/IDFT bằng cách tính biến đổi 1-D DFT các hàng của f(x,y) trước , sau đó biến đổi 1-D DFT dọc theo các cột của kết quả 1-D theo hàng. Ngoài ra chúng ta có thể tính IDFT sử dụng “DFT ALGORITHM”.

* Vậy việc tách thành 2 hàm rời rạc mới là F(x,v) và F(u,v) độ phức tạp của biến đổi DFT/IDFT là: O(n)= n2.

-Sử dụng biến đổi DFT/IDFT đơn giản hoá ở các phân tích trên và lọc ảnh trong miền tần số thông qua các bước sau:

**Bước 1:** Chuyển ảnh đầu vào f(x, y) có kích thước M × N thành ảnh fp(x, y) có kích thước P × Q. Với P = 2M và Q = 2N (vùng không chứa ảnh cho mức xám bằng 0). Bước này có ý nghĩa là mở rộng ảnh có kích thước gấp 4 lần ảnh đầu vào.

**Bước 2:** Tạo ảnh mới F(x, y) có các thành phần đối xứng qua tâm tại (P/2, Q/2) bằng cách nhân ảnh Fp (x, y) với (- 1)x+y

**Bước 3:** Tính DFT của ảnh Fp(x, y) bằng công thức đơn giản hoá **ĐGH-1**, **ĐGH-2** để thu được ảnh F(u, v).

**Bước 4**: Tạo bộ lọc H(u, v) có kích thước P × Q, các phần tử bộ lọc đối xứng với điểm trung tâm (P/2,Q/2).

**Bước 5**: Nhân từng cặp (elementwise multiplication) phần tử của ma trận F(u, v) với H(u, v) để thu được G(u, v). Cụ thể: G(i,k) = H(i,k)F(i, k) với i = 0, 1, 2, ..., P – 1 và k = 0, 1, 2, ..., Q-1.

**Bước 6**: Biến đổi ngược IDFT đối với G(u, v) ( theo ý tưởng công thức **ĐGH-1**, **ĐGH-2** ) . Tiếp tục nhân phần thực sau khi biến đổi ngược với (-1)x+y để tạo ảnh gp(x, y) có kích thước P x Q và tâm đối xứng (P/2, Q/2).

A black text on a white background

Description automatically generated

**Bước 7**: Trích vùng ảnh M × N từ ảnh gp(x, y) (từ góc phần tư trên bên trái) để được ảnh g(x, y). Ảnh g(x, y) là ảnh cuối cùng sau khi lọc.

* **Nâng Cao:**
* Các kỹ thuật dựa trên Fourier trở nên phổ biến nhờ sự phát triển của **Fast Fourier Transform (FFT)**.
* Cho phép biến đổi Fourier thực hiện trong thời gian có thể chấp nhận được.
* Giảm thời gian thực hiện biến đổi Fourier 100 - 600 lần với độ phức tạp MNlog2(MN).

**Câu 2. Thực hiện Code FFT IDEAL (HÀM LỌC THÔNG THẤP – IDEAL):**

A screen shot of a computer

Description automatically generated

**A screen shot of a computer

Description automatically generated**

**A computer screen shot of a program

Description automatically generated**

**A computer screen with text

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A computer screen shot of text

Description automatically generated**

**Kết quả:** A screenshot of a computer

Description automatically generated

**Áp dụng lọc Lowpass filter, Highpass filter cho cả ảnh xám và ảnh màu RGB trong miền tần số:**

**A screen shot of a computer program

Description automatically generated**

**A computer screen shot of text

Description automatically generated**

**A screen shot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screen shot of a computer code

Description automatically generated**

**Kết quả:**

A screenshot of a computer

Description automatically generated

**Các bước thực hiện:**

* **Bước 1**:Chuyển đổi Fourier 2D: **fft\_2d** và **ifft\_2d** được sử dụng để chuyển đổi ảnh sang miền tần số và chuyển đổi ngược lại.
* **Bước 2**:Tạo bộ lọc thông thấp và thông cao:

Bộ lọc thông thấp **loc\_thong\_thap\_khuon\_dang** giữ lại các thành phần tần số thấp của ảnh, giúp làm mờ ảnh và loại bỏ nhiễu tần số cao.

Bộ lọc thông cao **loc\_thong\_cao\_khuon\_dang** giữ lại các thành phần tần số cao của ảnh, giúp làm nổi bật các chi tiết và cạnh trong ảnh.

* **Bước 3**: Áp dụng bộ lọc: : **ap\_dung\_loc** Bộ lọc được áp dụng trực tiếp lên phổ tần số của ảnh đã được biến đổi. Sau đó, phổ tần số này được chuyển đổi ngược trở lại miền không gian để tạo ra ảnh đã được lọc.
* **Bước 4**:Xử lý ảnh xám và màu: **xu\_ly\_anh** xử lý ảnh xám hoặc màu bằng cách áp dụng cả bộ lọc thông thấp và thông cao, quá trình tương tự được áp dụng cho từng kênh màu riêng biệt.
* **Bước 5**:Hiển thị kết quả: Cuối cùng, sử dụng hàm imshow() của matlotlib để hiển thị ảnh gốc và ảnh đã qua xử lý để so sánh kết quả trước và sau khi áp dụng các bộ lọc.

**II.Object tracking in video:**

**1)object tracking in video (Theo dõi đối tượng “vật thể” trong video):**

- Theo dõi đối tượng, vật thể được định nghĩa là quá trình xác định vật thể thông qua màu sắc , hình dạng, cấu trúc của một vật thể trong một video đã chọn. Mục đích của việc theo dõi là giữ cho đối tượng được xác định và theo dõi qua từng thời gian, bất kể các thay đổi của ánh sáng, góc nhìn, tỷ lệ của vật thể, hoặc môi trường xung quanh vật thể đó. (vd: được sử dụng rộng rãi trong lĩnh vực an ninh, giám sát, theo dõi hành vi, phân tích giao thông như con người và xe cộ...)

A person taking a selfie

Description automatically generated

**2.Yêu cầu thực hiện:**

A screenshot of a computer screen

Description automatically generated

**Bước 1**:

Ta khai báo thư viện cần thiết để xử lí video và hình ảnh trong Python.

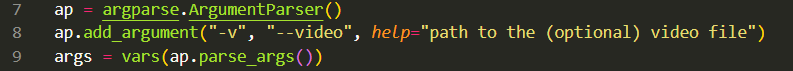
**1.import numpy as np:** là thư viện được sử dụng để xử lí các phép toán ma trận bằng cách import nó như np.

**2.import cv2:** thư viện xử lí hình ảnh và video.

**3.import time:** sử đụng để tạo ra độ trễ giữa các khung hình trong video.

**4..import argparse:** để phân tích các đối số truyền vào và lưu trữ để xử lí.

Tiếp theo:



**Tại dòng 7:** tạo một đối tượng từ argprase dùng để xử lý các đối số.

**Dòng 8:** thêm một đối số argument. Đối số “-v” “—video” sử dụng để chỉ định đường dẫn đến tệp video.

**Dòng 9:** Phân tích các đối số đã được người dùng cung cấp và lưu trữtrong biến args.

**Bước 2:**

-Đó là xác định các ngưỡng màu sắc để nhận diện vật thể và kiểm tra đường dẫn video đó có chính xác hay không:

A screen shot of a computer program

Description automatically generated

-Dòng 12 và 13: sử dụng **np.array** để tạo một danh sách các giá trị, và các thông số trong hình đại diện cho giá trị tối thiểu và và tối đa trong ngưỡng màu cần xác định (ở đây là màu xanh).

**Lưu ý:**Trong opencv, các pixel trong ảnh được lưu trữ theo thứ tự màu là (BGR) chứ không phải là RGB thường biết.

**Cách giải quyết:**

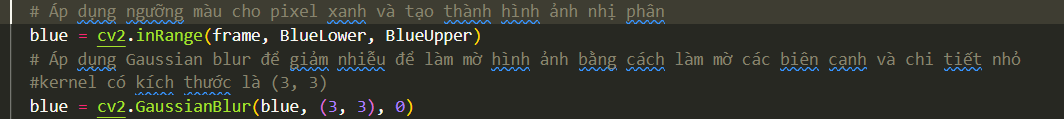
Để muốn thay đổi màu sắc nhận diện vật thể ,ta chỉ cần thay đổi thông số giá trị ngưỡng màu sắc là có thể xác định được. Vd ( np.array([0,0,100])

np.array([50,50,255]) trong trường hợp sẽ trả về là màu đỏ.)

“uint8” là một kiểu dữ liệu số nguyên không dấu 8 bit phù hợp cho việc lưu trữ các giá trị màu trong hình ảnh (8 bit cho mỗi kênh màu). **cv2.VideoCapture (./videos/mausac.mp4 "):** Mở tệp video từ đường dẫn cung cấp. Đường dẫn tệp video là tương đối so với thư mục làm việc hiện tại.

Nếu đúng đường dẫn thì file sẽ đọc khung hình trong video và ngược lại nếu sai thì tệp sẽ không tìm thấy.

**Bước 3: Áp dụng ngưỡng màu và làm mờ hình ảnh:**



-Ta sử dụng **hàm inRange** để kiểm tra pixel trong khung hình.

-Và bộ lọc **GaussianBlur** để làm mờ hình ảnh và giảm nhiễu (Kernel có kích thước là 3,3).

**Bước 4:** Tìm Contours và vẽ hình chữ nhật xung quanh vật thể:

A screen shot of a computer code

Description automatically generated

-Dòng 38: sử dụng hàm **“findContours”** để tìm các đường viền trong ảnh.

**Lưu ý:** opencv mới chỉ trả 2 giá trị contours và hierarchy : danh sách các đường viền tìm được và thông tin về cấu trúc phân cáp của đường viền.Thay vì là (\_,cnts,\_) thì bây giờ chỉ còn (cnts,\_).

-Dòng 39->42 : tìm đường viền lớn nhất và vẽ hình chữ nhật màu xanh xung quanh ( nếu có).

-Dòng 43: sử dụng hàm “drawContours để vẽ hình chữ nhật bao quanh contour lên khung hình gốc (frame).

**+ Đối số thứ hai:** là danh sách chứa các điểm đỉnh của hình chữ nhật, trong trường hợp này chỉ có một hình chữ nhật nên được đưa vào dưới dạng một danh sách con [rect].

**+ Đối số thứ ba:**  là chỉ số contour muốn vẽ. Nếu đặt giá trị này là -1, tất cả các contours trong danh sách được vẽ. Trong trường hợp này, sẽ vẽ tất cả các contours trong danh sách.

 **+ Đối số thứ tư (0, 255, 0):** đại diện cho màu của đường viền (màu xanh lá cây trong trường hợp này), # và đối số cuối cùng là độ dày của đường viền.

+**Đối số cuối cùng:** là độ dày của đường viền.

**Bước 5:** Hiển thị kết quả:

A screen shot of a computer code

Description automatically generated

-Dòng 50,51: hiển thị khung hình gốc với đường viền được vẽ và hiển thị hình ảnh nhị phân trên cửa sổ.

-Dòng 53: độ trễ ngắn giữa các khung hình.

**Bước cuối cùng:** xử lí thoát khỏi màn hình

A screen shot of a computer

Description automatically generated

Dòng 55,56: Nhấn q để thoát cửa sổ màn hình.

Camera.Release(): Giải phóng các tài nguyên camera.

Cv2.destroyAllwindows(): Đóng tất cả các cửa sổ hiện thị trên màn hình.

**Kết Quả:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screen shot of a computer

Description automatically generated