**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ ĐIỆN TỬ**



**BÁO CÁO THỰC HÀNH**

**Môn học**: Thực hành Xử Lý Ảnh

**GVHD:** Thầy Phan Vinh Hiếu

**Thành viên nhóm**:

Nguyễn Hạnh Uyên

Nguyễn Mai Đức Thắng

Bùi Trọng Nghĩa

*Hồ Chí Minh, ngày 4, tháng 4, năm 2024*

**NỘI DUNG**

**1 Code cutvideo.py**

import cv2

def FrameCapture(Path):

    vidObj = cv2.VideoCapture(Path)

    count = 0

    success = 1

    while success:

        success, image = vidObj.read()

        if count % 10 == 0:

            cv2.imwrite("dataset\\frame%d.jpg"%(count/15),image)

        count+=1

print("0")

#if \_\_name\_\_ == '\_\_name\_\_':

FrameCapture(r"C:\Users\User\Desktop\Xu ly anh\NEW\WIN\_20240321\_13\_41\_09\_Pro.mp4")

print("1")

**Giải thích code:**

Đoạn code trên sử dụng thư viện OpenCV (cv2) để chụp các khung hình từ một video và lưu chúng dưới dạng các tệp hình ảnh jpg. Dưới đây là từng bước được thực hiện trong hàm `FrameCapture`:

1. `import cv2`: Import thư viện OpenCV để làm việc với xử lý video và hình ảnh.

2. `def FrameCapture(Path)`: Định nghĩa một hàm có tên `FrameCapture`, nhận một đường dẫn của video đầu vào.

3. `vidObj = cv2.VideoCapture(Path)`: Mở video từ đường dẫn đã cho bằng cách sử dụng hàm `VideoCapture` từ thư viện OpenCV. Video này sẽ được đại diện bằng biến `vidObj`.

4. `count = 0`: Khởi tạo biến `count` để đếm số lượng khung hình đã chụp.

5. `success = 1`: Khởi tạo biến `success` với giá trị 1, đại diện cho việc đọc video thành công.

6. `while success:`: Bắt đầu một vòng lặp while, trong khi việc đọc video thành công.

7. `success, image = vidObj.read()`: Đọc một khung hình từ video và lưu nó vào biến `image`. Biến `success` sẽ chứa giá trị `True` nếu việc đọc khung hình thành công, và `False` nếu không còn khung hình nào để đọc.

8. `if count % 10 == 0:`: Kiểm tra nếu `count` là bội số của 10.

9. `cv2.imwrite("dataset\\frame%d.jpg"%(count/15),image)`: Lưu khung hình hiện tại dưới dạng một tệp ảnh jpg trong thư mục "dataset". Tên tệp được định dạng theo số lượng khung hình đã chụp, chia cho 15 (mỗi 10 khung hình lưu một khung hình, do đó sau 150 khung hình sẽ lưu một khung hình).

10. `count+=1`: Tăng biến đếm `count` lên 1 để theo dõi số lượng khung hình đã chụp.

11. `print("0")`: In ra màn hình chuỗi "0".

12. `FrameCapture(r"C:\Users\User\Desktop\Xu ly anh\NEW\WIN\_20240321\_13\_41\_09\_Pro.mp4")`: Gọi hàm `FrameCapture` với đường dẫn của video được truyền vào.

13. `print("1")`: In ra màn hình chuỗi "1".

Dòng `if \_\_name\_\_ == '\_\_main\_\_':` đã được bỏ comment (ở trên code), nhưng nó chưa được sử dụng trong code. Đoạn code sẽ chạy hàm `FrameCapture` khi được gọi và sau đó in ra "0" và "1".

**2 Code detect.py**

Code:

import cv2

from ultralytics import YOLO

model = YOLO(r'C:\Users\User\Desktop\Xu ly anh\NEW\runs\detect\train5\weights\best.pt')

vid = cv2.VideoCapture(0)

#vid = cv2.VideoCapture(r'C:\Users\User\Desktop\Xu ly anh\NEW\WIN\_20240321\_13\_41\_09\_Pro.mp4')

while (True):

    ret, frame = vid.read()

    frame = cv2.flip(frame, 1) #mirror

    results = model(frame)

    annotated\_image = results[0].plot() #

    cv2.imshow("YOLOv8 Inference", annotated\_image)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

vid.release()

cv2.destroyAllWindows()

**Giải thích code**:

Đoạn code trên sử dụng thư viện OpenCV và YOLO (You Only Look Once) từ gói ultralytics để nhận dạng đối tượng trong video từ webcam hoặc từ một tệp video đã được chỉ định.

Dưới đây là giải thích chi tiết từng dòng code:

1. `import cv2`: Import thư viện OpenCV để làm việc với xử lý video và hình ảnh.

2. `from ultralytics import YOLO`: Import lớp YOLO từ gói ultralytics để sử dụng mô hình nhận dạng đối tượng YOLO.

3. `model = YOLO(r'C:\Users\User\Desktop\Xu ly anh\NEW\runs\detect\train5\weights\best.pt')`: Khởi tạo một đối tượng YOLO với mô hình được định nghĩa trong tệp `best.pt`. Đường dẫn này trỏ đến tệp trọng số của mô hình đã được huấn luyện trước.

4. `vid = cv2.VideoCapture(0)`: Khởi tạo một đối tượng VideoCapture để đọc từ webcam (hoặc thiết bị có thể chụp video) được kết nối với máy tính. Tham số 0 ở đây đại diện cho webcam được sử dụng.

5. `while (True):`: Bắt đầu vòng lặp vô hạn để liên tục đọc và xử lý các khung hình từ video.

6. `ret, frame = vid.read()`: Đọc một khung hình từ video và lưu nó vào biến `frame`. Biến `ret` sẽ chứa `True` nếu việc đọc khung hình thành công, và `False` nếu không còn khung hình nào để đọc.

7. `frame = cv2.flip(frame, 1)`: Lật ngang khung hình (tạo hiệu ứng phản chiếu) bằng cách sử dụng hàm `flip` từ thư viện OpenCV.

8. `results = model(frame)`: Sử dụng mô hình YOLO để dự đoán các đối tượng trong khung hình `frame`. Kết quả sẽ chứa thông tin về các đối tượng được phát hiện.

9. `annotated\_image = results[0].plot()`: Vẽ bounding box và nhãn của các đối tượng được phát hiện lên khung hình gốc. Kết quả được lưu vào biến `annotated\_image`.

10. `cv2.imshow("YOLOv8 Inference", annotated\_image)`: Hiển thị khung hình đã được nhận dạng bằng YOLO trên cửa sổ mới có tên "YOLOv8 Inference".

11. `if cv2.waitKey(1) & 0xFF == ord('q'): break`: Chờ người dùng nhấn phím 'q' để thoát khỏi vòng lặp while. Hàm `waitKey` sẽ chờ một lượng thời gian nhỏ (1 miligiây trong trường hợp này) và trả về mã ASCII của phím được nhấn.

12. `vid.release()`: Giải phóng tài nguyên của đối tượng VideoCapture.

13. `cv2.destroyAllWindows()`: Đóng tất cả các cửa sổ hiển thị của OpenCV.

**3. Code my\_train.py**

Code:

# Train du lieu anh

from ultralytics import YOLO

# Load a model

#model = YOLO('yolov8n.yaml') # build a new model from YAML

model = YOLO(r'C:\Users\User\Desktop\Xu ly anh\NEW\runs\detect\train7\weights\last.pt') # load a pretrained model (recommended for training) goc:yolov8n.pt

if \_\_name\_\_ == '\_\_main\_\_': # can khi chay bang GPU

    model.train(data='dataset.yaml', epochs=10, imgsz=640, batch=10, optimizer='Adam', workers=1)

    metrics = model.val()

**Giải thích code:**

Đoạn code này sử dụng gói `ultralytics` để huấn luyện một mô hình YOLO trên dữ liệu ảnh.

Dưới đây là giải thích từng dòng code:

1. `from ultralytics import YOLO`: Import lớp YOLO từ gói ultralytics để sử dụng mô hình YOLO.

2. `model = YOLO(r'C:\Users\User\Desktop\Xu ly anh\NEW\runs\detect\train7\weights\last.pt')`: Tạo một đối tượng mô hình YOLO bằng cách tải một mô hình được huấn luyện trước từ tệp `last.pt`. Đường dẫn này trỏ đến tệp trọng số cuối cùng của mô hình sau quá trình huấn luyện.

3. `if \_\_name\_\_ == '\_\_main\_\_':`: Kiểm tra xem đoạn code có được chạy trực tiếp từ chính file script hay không. Điều này có nghĩa là nếu đoạn code này được import vào một file khác, các dòng code trong khối này sẽ không được thực thi.

4. `model.train(data='dataset.yaml', epochs=10, imgsz=640, batch=10, optimizer='Adam', workers=1)`: Bắt đầu quá trình huấn luyện mô hình. Các tham số cụ thể bao gồm:

- `data='dataset.yaml'`: Đường dẫn đến tệp cấu hình YAML mô tả dữ liệu huấn luyện.

- `epochs=10`: Số lượng epochs (vòng lặp huấn luyện).

- `imgsz=640`: Kích thước ảnh đầu vào mà mô hình sẽ sử dụng cho huấn luyện.

- `batch=10`: Kích thước batch (số lượng ảnh được sử dụng trong mỗi lần cập nhật trọng số).

- `optimizer='Adam'`: Thuật toán tối ưu hóa được sử dụng (trong trường hợp này là Adam).

- `workers=1`: Số lượng worker (tiến trình) được sử dụng trong quá trình huấn luyện.

5. `metrics = model.val()`: Đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm định và lưu trữ kết quả vào biến `metrics`. Điều này cho phép người dùng đánh giá hiệu suất của mô hình sau mỗi epoch trong quá trình huấn luyện.

Tóm lại, đoạn code này huấn luyện một mô hình YOLO trên dữ liệu ảnh được mô tả trong tệp YAML, sử dụng các siêu tham số đã được chỉ định và sau đó đánh giá hiệu suất của mô hình sau mỗi epoch.

**4. Code điều khiển đèn led**

**Code:**

char TransmitData, ReceiveData;

void main() {

ADCON1 |= 0x0F;

CMCON |=7;

PORTB =0x00; LATB = 0x00;

TRISB0\_bit = 1;

TRISB1\_bit = 1;

TRISB2\_bit = 1;

PORTE =0x00; LATE = 0x00;

TRISE1\_bit = 0;

TRISE2\_bit = 0;

TRISE0\_bit = 0;

LATE1\_bit = 1;

LATE2\_bit = 1;

LATE0\_bit = 1;

UART1\_Init(9600);

delay\_ms(100);

while(1)

{

if(UART1\_Data\_Ready() == 1)

{

ReceiveData = UART1\_Read();

if(ReceiveData == '0')

{

LATE0\_bit = 0;

}

else if(ReceiveData == '4')

{

LATE0\_bit = 1;

}

else if(ReceiveData == '1')

{

LATE1\_bit = 0;

}

else if (ReceiveData == '5')

{

LATE1\_bit = 1;

}

else if (ReceiveData == '2')

{

LATE2\_bit = 0;

}

else if (ReceiveData == '6')

{

LATE2\_bit = 1;

}

}

}

}

**Giải thích code:**

-char TransmitData, ReceiveData: Khai báo hai biến TransmitData và ReceiveData và kiểu dữ liệu

-ADCON1 |= 0x0F;CMCON |=7:Khai báo cấu hình PIC hoạt động và chuyển đổi các chân Analog thành chân Digital.

-PORTE =0x00; LATE = 0x00;TRISE1\_bit = 0;TRISE2\_bit = 0; TRISE0\_bit = 0;LATE1\_bit = 1;LATE2\_bit = 1; LATE0\_bit = 1: Khai cấu hình port E output và cấu hình chân, 3 chân set 0 mức tích cực thấp và trạng thái.

-UART1\_Init(9600);delay\_ms(100): Cấu hình module UART1 tốc độ hoạt động là 9600, cho delay 100s để vdk hoạt động ổn định.

- while(1): Khởi tạo vòng lặp vô hạn

- if(UART1\_Data\_Ready() == 1): Kiểm tra xem dữ liệu gửi vào đã đọc được hay chưa, nếu bằng 1 là sẵn sàng, bằng 0 là chưa.

-ReceiveData = UART1\_Read(); Chứa dữ liệu được gửi về sau đó truyền vô biến ReceiveData

-if(ReceiveData == '0'): Nếu dữ liệu nhận được là kí tự 0 thì bật đèn led chân RE0

-else if(ReceiveData == '4'): Nếu dữ liệu nhận được là kí tự 4 thì đèn led tắt chân RE0

-else if(ReceiveData == '1'): Nếu dữ liệu nhận được là kí tự 1 thì đèn led sáng chân RE1

-else if (ReceiveData == '5'): Nếu dữ liệu nhận được là kí tự 5 thì đèn led tắt chân RE1

-else if (ReceiveData == '2'): Nếu dữ liệu nhận được là kí tự 2 thì đèn led sáng chân RE2

-else if (ReceiveData == '6') Nếu dữ liệu nhận được là kí tự 6 thì đèn led tắt chân RE2

**MINH CHỨNG KẾT QUẢ THỰC HÀNH NHẬN DIỆN**

A computer screen shot of a person raising his hands

Description automatically generated

Có “trọng nghĩa” thì đèn 1 sáng

A computer screen shot of a person holding a device

Description automatically generated

Có “chai nước” thì đèn 2 sáng

A screenshot of a computer

Description automatically generated

Có “Đức Thắng” thì đèn 3 sáng

A screenshot of a computer

Description automatically generated

Có cả 3 đối tượng thì 3 đèn sáng