CS114.K21.KHTN

NHẬN DIỆN HƯỚNG NGÓN TAY BÀN TAY PHẢI ÁP DỤNG VÀO GAME RẮN SĂN MỒI

Sinh viên: Nguyễn Lâm Quỳnh

MSSV: 18523126

Giảng viên:Lê Đình Duy

Phạm Nguyễn Trường An

NỘI DUNG

1. Lý do chọn đề tài
2. Giới thiệu đề tài
3. Hướng giải quyết
4. Quy trình
5. Chạy chương trình
6. Kết luận
7. Tài liệu tham khảo

LÝ DO CHỌN ĐỀ TÀI

- Bài toán có ứng dụng thực tế cụ thể là trong các trò chơi điều khiển và có thể được tích hợp vào trong các thiết bị thông minh.
- Có khả năng áp dụng các thuật toán supervised learning.
- Chạy chương trình dễ dàng và thuận tiện.

GIỚI THIỆU ĐỀ TÀI

- Trò chơi rắn săn mồi là một trò chơi cổ điển với lối chơi đơn giản là di chuyển con rắn bằng cách nhấn các phím mũi tên trên, xuống, trái, phải sao cho con rắn đến được chấm thức ăn. Nhằm hướng đến cách tiếp cận phương thức chơi mới, tôi đã áp dụng Machine Learning vào việc di chuyển hướng đi của rắn bằng cách sử dụng cử chỉ ngón tay với ngón chỉ hướng là ngón cái bàn tay phải chỉ các hướng trái, phải, trên, xuống. Chương trình sẽ sử dụng webcam để chạy trực tiếp, code của game được thu thập trên internet.
- Input bài toán là hình ảnh ngón tay cái trỏ các hướng trái, phải, trên, xuống.
- Output là hướng của ngón tay và hướng con rắn di chuyển

HƯỚNG GIẢI QUYẾT

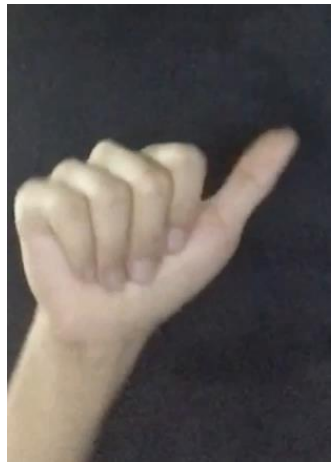
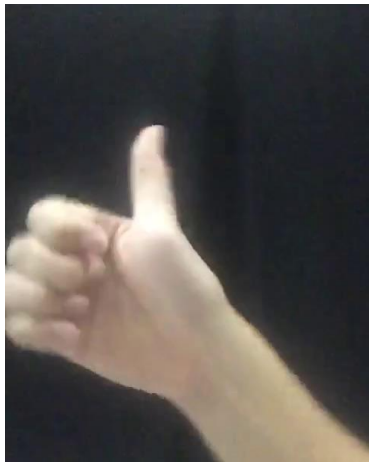
- Nhận xét đặc trưng của đối tượng: hình dáng bàn tay, hướng => Quy định ảnh xử lí là ảnh nhị phân
- Áp dụng các thuật toán supervised learing để xử lí bài toán
- Sau khi tạo ra model sẽ đưa vào code game rắn săn mồi để chạy chương trình. Ở trong code game sẽ thêm các hàm rút trích đặc trưng, hàm dự đoán và một hàm sử dụng webcam để nhận hình ảnh. Ảnh sẽ sử dụng các phương pháp xử lí để tách nền và chuyển về ảnh nhị phân. Sau đó sử dụng ảnh đó để rút trích đặc trưng và dự đoán hướng đi cho rắn.

QUY TRÌNH

Thu thập dữ liệu

Bộ dữ liệu được thu thập bằng hai cách:

- Thực tế: thu thập từ 4 người bằng cách quay video với phông màu đen, bàn tay nằm trọn trong khung hình, hướng chỉ tay là ngón cái bàn tay phải, video thu được sẽ được tách lấy các frame. Tổng cộng thu được 14800 ảnh.

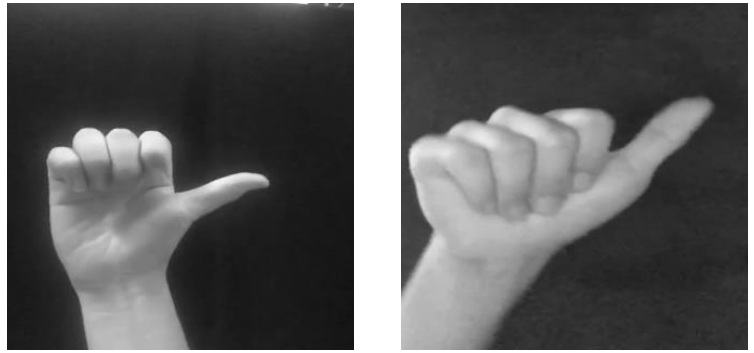

Thu thập dữ liệu

- Internet: thu thập trên dữ liệu trên kaggle thu được 4000 ảnh.

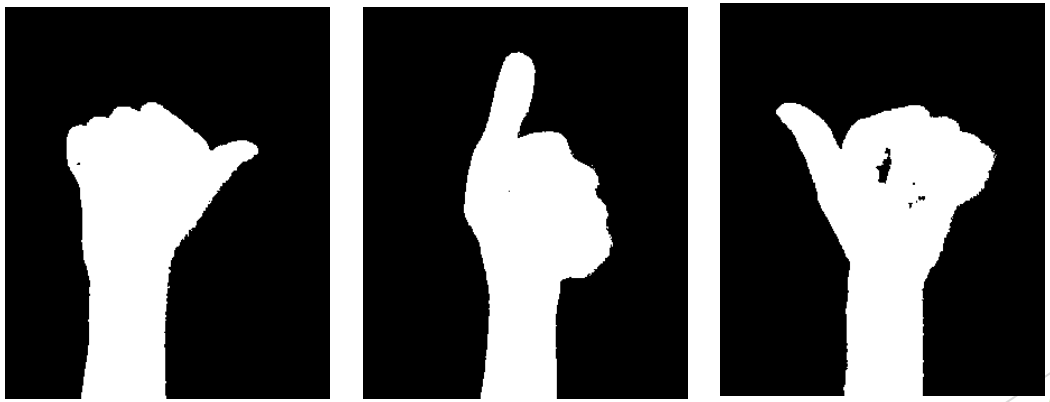
Tổng cộng thu được 18800 ảnh.

Tiền xử lí dữ liệu

- Thực hiện theo các bước:
- Chuyển về ảnh xám



- Chuyển về ảnh nhị phân



- Gắn nhãn cho ảnh với 1 trong 4 label: left, right, up, down.

Phân chia bộ dữ liệu

10Bộ dữ liệu được phân chia theo tỉ lệ train, test là 80/20

Rút trích đặc trưng

Sử dụng HOG để rút trích đặc trưng, trước đó sử dụng cv2 để tìm ra contours của hình chính là viền bàn tay. Sau khi tìm được nó, ta sẽ vẽ hình chữ nhật đứng bao xung quanh bàn tay, resize kích thước về 150x150 rồi áp dụng HOG vào hình chữ nhật đó.

Mô tả thuật toán máy học

- Các thuật toán máy học tôi dung để kiểm thử:
- LogisticRegression
- LinearDiscriminantAnalysis
- KNeighborsClassifier
- DecisionTreeClassifier
- RandomForestClassifier
- GaussianNB

Đánh giá thuật toán máy học

```
====================LogisticRegression==

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

```
====================KNeighborsClassifier=

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

Đánh giá thuật toán máy học

```
====================KNeighborsClassifier=

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

```
====================DecisionTreeClassifier==============

accuracy =  0.9936170212765958
Confusion matrix:
[[938   0   2   0]
 [  5 932   0   3]
 [  2   3 946   3]
 [  2   1   3 920]]
precision: [0.9904963  0.9957265  0.99474238 0.99352052]
recall: [0.99787234 0.99148936 0.99161426 0.99352052]
fscore: [0.99417064 0.99360341 0.99317585 0.99352052]
support: [940 940 954 926]
```

Đánh giá thuật toán máy học

```
====================RandomForestClassifier=

accuracy =  1.0
Confusion matrix:
[[940   0   0   0]
 [  0 940   0   0]
 [  0   0 954   0]
 [  0   0   0 926]]
precision: [1. 1. 1. 1.]
recall: [1. 1. 1. 1.]
fscore: [1. 1. 1. 1.]
support: [940 940 954 926]
```

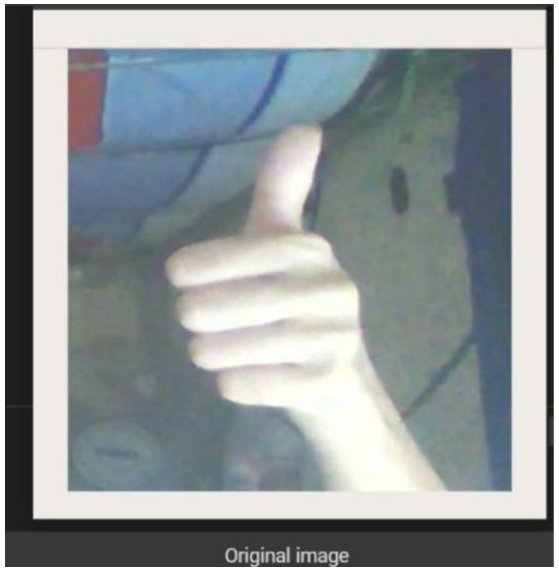
```
====================GaussianNB====================

accuracy =  0.9960106382978723
Confusion matrix:
[[940   0   0   0]
 [  3 934   0   3]
 [  1   2 946   5]
 [  0   1   0 925]]
precision: [0.99576271 0.99679829 1.         0.99142551]
recall   : [1.         0.99361702 0.99161426 0.99892009]
fscore   : [0.99787686 0.99520511 0.99578947 0.99515869]
support  : [940 940 954 926]
```

Kết quả cho thấy LinearDiscriminantAnalysis cho kết quả tốt nhất, chính là model được lựa chọn.

CHẠY CHƯƠNG TRÌNH

- Ngôn ngữ: python3
- Thư viện: scikit-image 0.14.2,numpy 1.15,…
- Thực hiện theo các bước để xử lí ảnh trước khi đưa vào trích xuất đặc trưng:
  - Lấy hình ảnh từ webcam



CHẠY CHƯƠNG TRÌNH

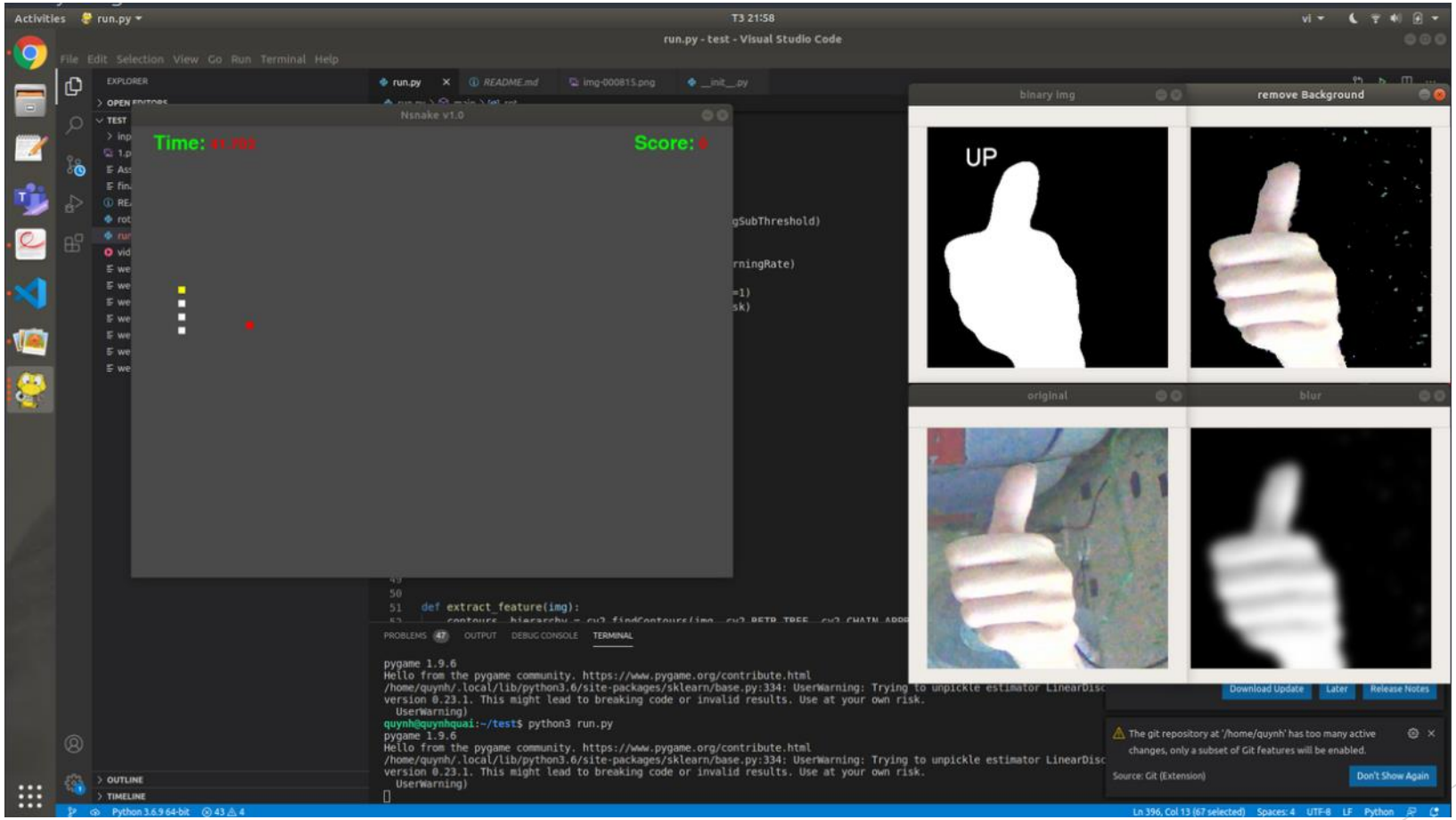
- Sử dụng BackgroundSubtractor để trừ nền

- Chuyển về ảnh grayscale và làm mờ

CHẠY CHƯƠNG TRÌNH

- Chuyển về ảnh nhị phân

CHẠY CHƯƠNG TRÌNH



KẾT LUẬN

- Việc chạy real-time mới chỉ giải quyết được khi background không chuyển động => Hướng giải quyết: Sử dụng YOLO để detect đối tượng
- Model hoạt động tốt trong hầu hết mọi trường hợp. Tuy nhiên trong lúc di chuyển từ hướng đi cũ đến hướng đi mới, khoảng cách giữa thời gian đó model có thể dự đoán sai.=> Sử dụng các phương pháp khác để rút trích đặc trưng như CNN,…

TÀI LIỆU THAM KHẢO

- https://www.tensorflow.org/
- https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html
- https://scikit-learn.org/stable/modules/model_evaluation.html#:~:text=2.-,Classification%20metrics,values%2C%20or%20binary%20decisions%20values.
- https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python
- Source code game snake: https://gist.github.com/someoneigna/5022021
- https://docs.opencv.org/3.4/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html


THANKS
FOR WATCHING