BÁO CÁO MÔN HỌC

CS231 & CS406

Đề tài: Phát hiện và làm đẹp khuôn mặt người Việt Nam

GVHD: TS. Mai Tiến Dũng

Đào Văn Tài – 19522148
Nguyễn Thành Trọng – 19522410
Ngô Gia Kiệt – 19521725

Nội dung

01

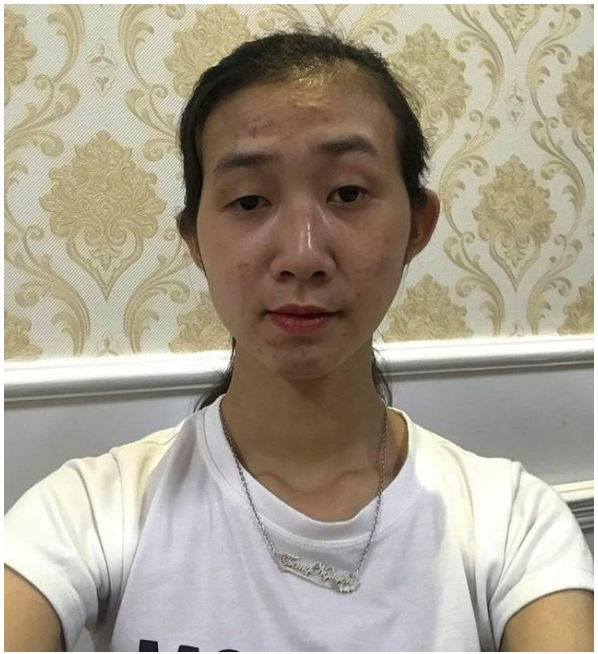
Bối cảnh đề tài

Bối cảnh

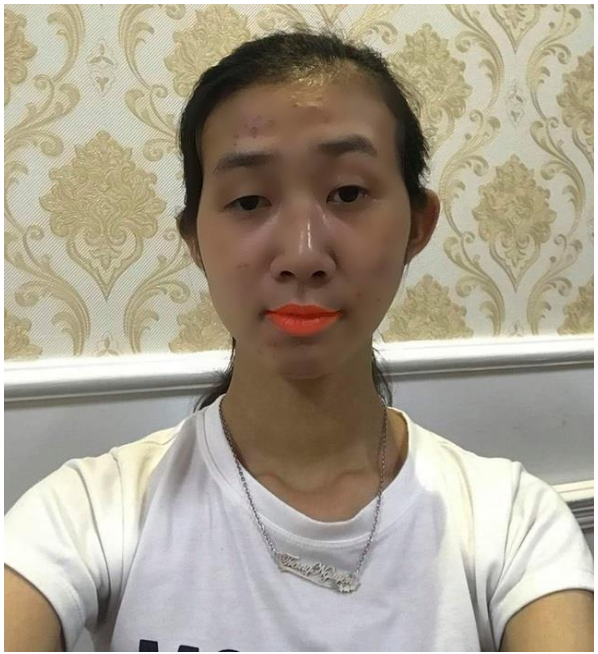
- **Trong bối cảnh dịch COVID-19, việc học, làm việc online không còn xa lạ đối với mọi người. Hầu hết học sinh, sinh viên rất ngại bật webcam dẫn đến việc tương tác giữa thầy cô và học sinh sinh viên gặp nhiều khó khăn.**

- **Vậy nên nhóm chúng em đã thực hiện đề tài thực hiện một lớp trang điểm để các bạn phần nào đó tự tin hơn trước webcam**

Input: Ảnh chân dung chứa 1 hoặc nhiều khuôn mặt ở khoảng cách gần

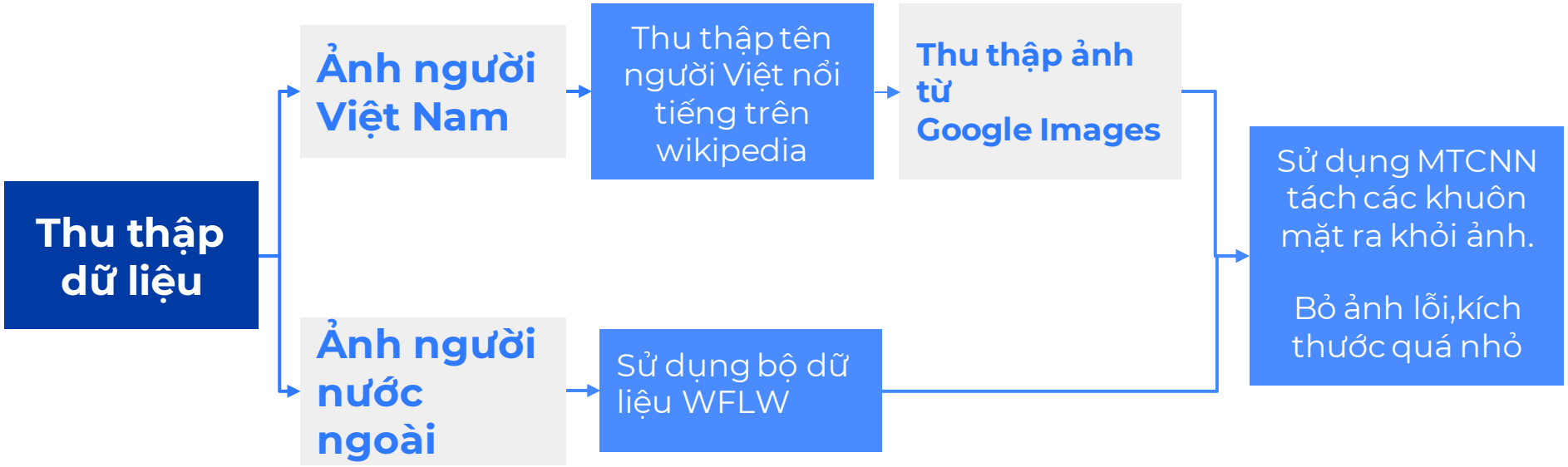


Output: Ảnh chân dung với các khuôn mặt người Việt Nam đã được trang điểm



02

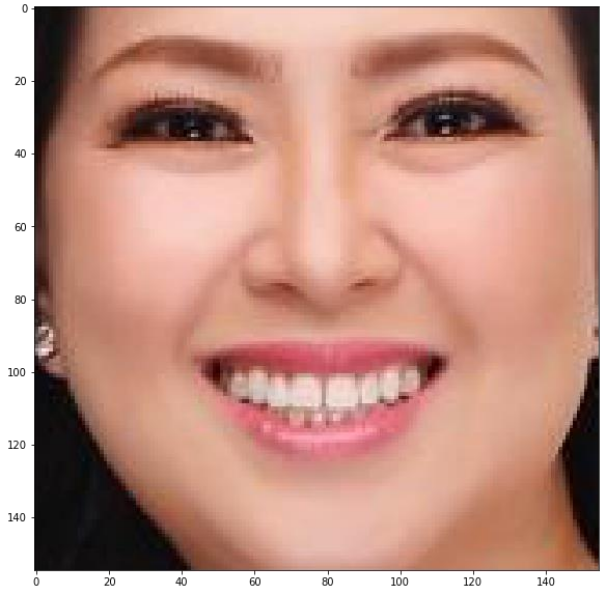
Bộ dữ liệu


Thu thập dữ liệu
Ảnh người Việt Nam
Thu thập tên người Việt nổi tiếng trên wikipedia
Thu thập ảnh từ Google Images
Ảnh người nước ngoài
Sử dụng bộ dữ liệu WFLW
Sử dụng MTCNN tách các khuôn mặt ra khỏi ảnh.
Bỏ ảnh lỗi,kích thước quá nhỏ

Dữ liệu

Phát hiện khuôn mặt

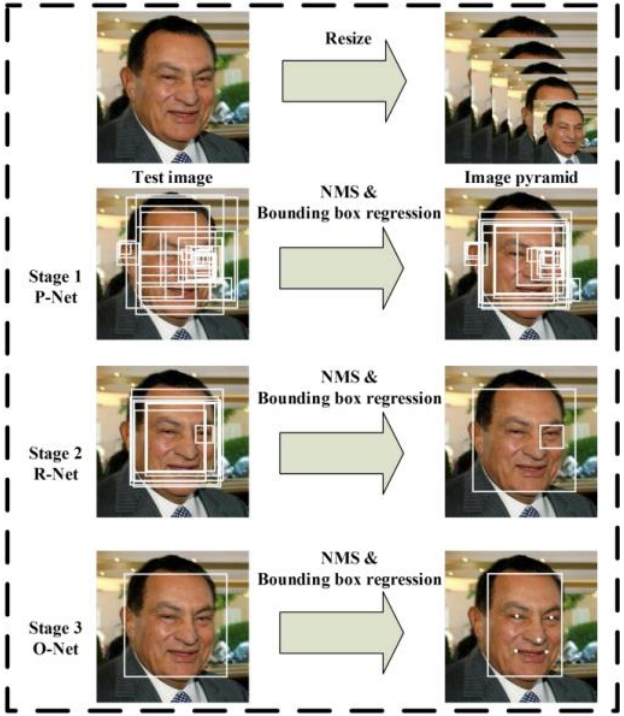
MTCNN



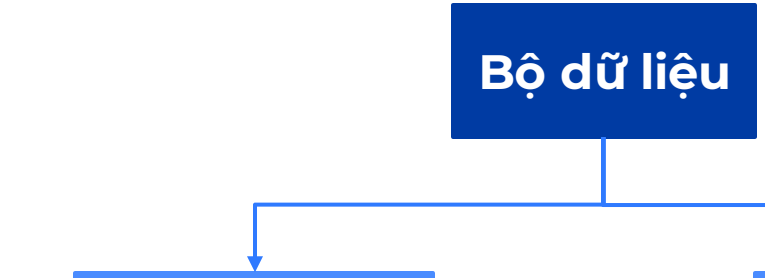
Dlib

MTCNN(Multi-task Cascaded Convolutional Networks)

- MTCNN là mạng nơ-ron dành cho việc phát hiện khuôn mặt có trong ảnh
- MTCNN bao gồm 3 mạng nơ-ron phức tạp dần theo thứ tự: P-Net, R-Net và O-Net



Dữ liệu


Bộ dữ liệu
Ảnh người Việt Nam
Ảnh người nước ngoài
10000 Ảnh
9589 Ảnh

03

PHÁT HIỆN KHUÔN MẶT VÀ PHÂN LOẠI

Input: Bức ảnh chân dung con người

Output: Bounding box chứa khuôn mặt người Việt Nam

Phát hiện và phân loại khuôn mặt

Bước 1

- Phát hiện khuôn mặt
  - MTCNN

Bước 2

- Trích xuất đặt trưng ảnh
  - Histogram
  - Sử dụng pre-trained model.

Bước 3

- Phân loại khuôn mặt
  - SVM
  - KNN
  - Custom

Pre-trained model

Sử dụng pre-trained model của mạng nơ-ron OpenFace để trích xuất đặc trưng của khuôn mặt thành những vector embedding 128 chiều.

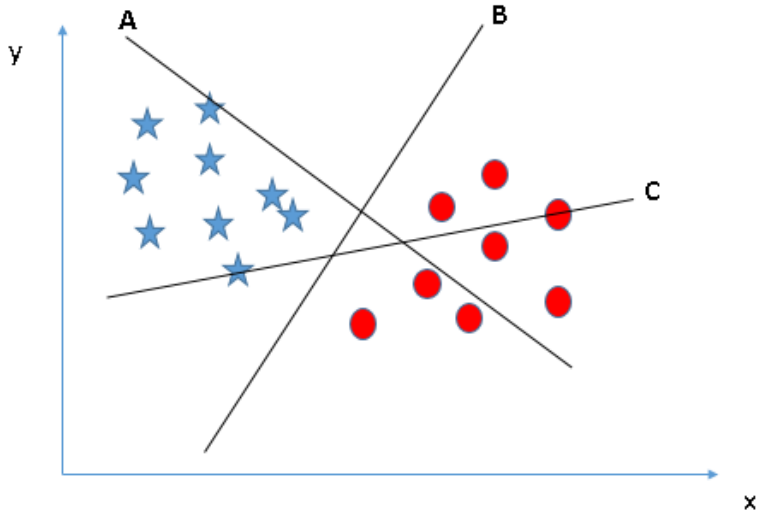
- nn4.v2
- nn4.small1.v1
- nn4.small2.v1

https://cmusatyalab.github.io/openface

Phân loại khuôn mặt

Support Vector Machine (SVM)

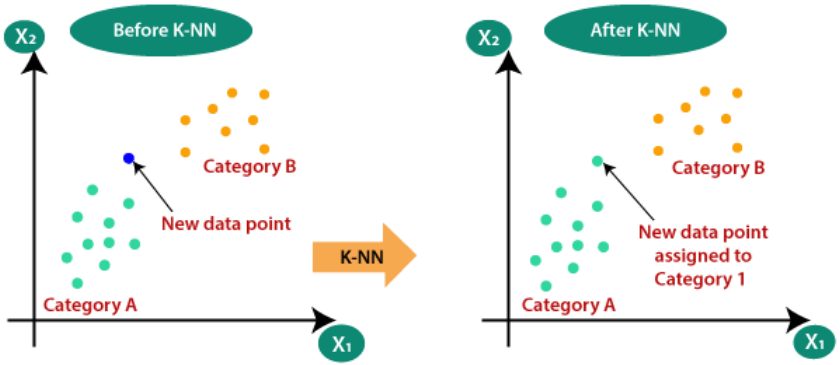
SVM là một thuật toán chủ yếu hỗ trợ giải quyết bài toán phân loại.



Phân loại khuôn mặt

K-nearest neighbors (KNN)

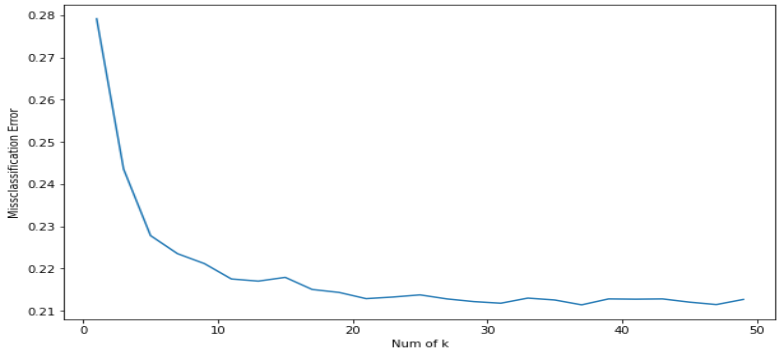
- Thuật toán K-NN lưu trữ tất cả dữ liệu có sẵn và phân loại một điểm dữ liệu mới dựa trên sự tương đồng.

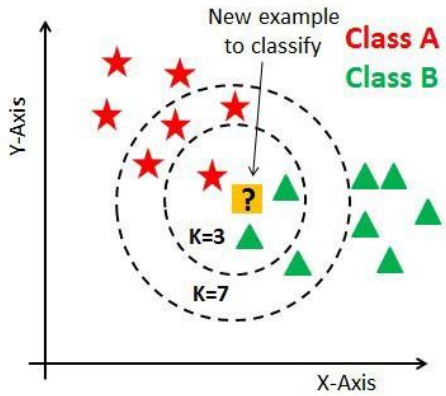


Phân loại khuôn mặt

K-nearest neighbors (KNN)

- Tìm số k neighbors tối ưu cho bài toán
- Biểu đồ các k neighbors:





- K tối ưu cho bài toán này là 37

Phân loại khuôn mặt

Mạng tự xây

```
def create_model():
  model = Sequential()
  model.add(Dense(10, input_dim=256, activation='relu'))
  model.add(Dense(5, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
```

Kết quả

Kết quả	SVM	KNN	CUSTOM
Histogram	60.64%	58.58%	61.74%
Pre-Train nn4.v2	60.67%	56.76%	60.44%
Pre-Train nn4.small1.v1	63.43%	58.63%	61.84%
Pre-Train nn4.small2.v1	64.96%	60.69%	64.45%

Tiền xử lí Blob

Blob images là để giảm nhiễu cho ảnh do chiếu sáng
Ta sử dụng hàm cv2.dnn.blobFromImage:
Chức năng:

+ Mean subtraction
+ Scaling

Mean subtraction

- Mean subtraction giúp cho ảnh chống lại các thay đổi về độ sáng trong ảnh

- Các biến sau là các trung bình của từng kênh Red, Green và Blue:
  + μ_R
  + μ_G
  + μ_B
- Ở đây các giá trị trung bình của các kênh R, G và B lần lượt là 104, 117 và 123 (các giá trị này thường xuất hiện trong các pre-trained model

- Tiến hành trừ các giá trị trung bình với mỗi kênh màu:
  + $R = R - \mu_R$
  + $G = G - \mu_G$
  + $B = B - \mu_B$

Mean subtraction

- R= 104
- G= 117
- B= 123

=

Scaling

- Scaling dùng để chuẩn hóa ảnh sau khi mean subtraction
- Tham số scaleFactor là độ lệch chuẩn của tập training hoặc được cài đặt thủ công, $\sigma = 1$ tức là mặc định không scaling
- Tham số σ tham gia vào qui trình chuẩn hóa:
  + $R = (R - \mu_R)/\sigma$
  + $G = (G - \mu_G)/\sigma$
  + $B = (B - \mu_B)/\sigma$

Kết quả

Kết quả	SVM	KNN	CUSTOM
Pre-Train nn4.small2.v1	81%	79%	82%

Một số trường hợp sai

0 | 1

1 | 0

Người nước ngoài có nét giống người Việt và ngược lại

Ảnh bị ám vàng

Khuôn mặt bị che bởi phụ kiện

04

LÀM ĐẸP TRÊN KHUÔN MẶT

Làm mịn da, trang điểm khuôn mặt

Bước 1

- Xác định vùng da mặt
  - Sobel edge detection
  - Lấy ngưỡng - HSV

Bước 2

- Làm mịn vùng da mặt

Bước 3

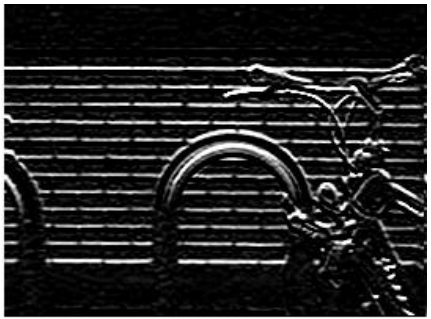
- Trang điểm khuôn mặt

Xác định vùng da bằng Sobel edge dectection

Sobel edge dectection là một thuật toán được dùng rộng rãi trong các bài toán phát hiện cạnh của vật thể. Các cạnh được đánh dấu theo sự thay đổi đột ngột cường độ của điểm ảnh.

Ảnh trên thang màu xám

Sobel theo trục x



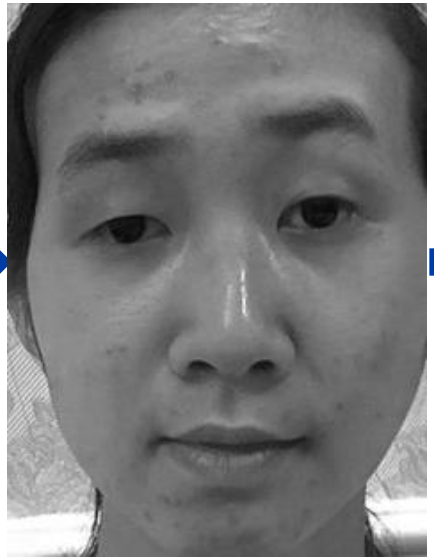
Sobel theo trục y

Sobel với cả 2 cạnh

Xác định vùng da bằng Sobel edge dectection

- Với Sobel, coi toàn bộ các vùng đồng màu đều là vùng da, bao gồm phần tóc đen và vùng nền phía sau.

- Threshold ảnh hưởng tới độ chi tiết của vùng da

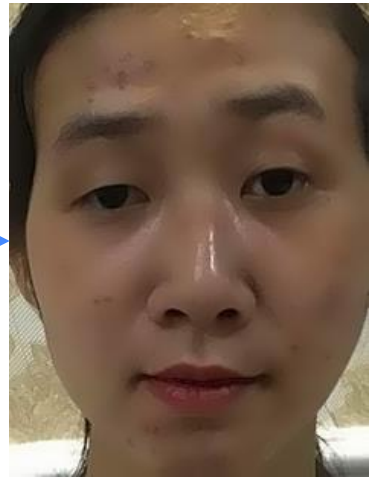


Threshold =20

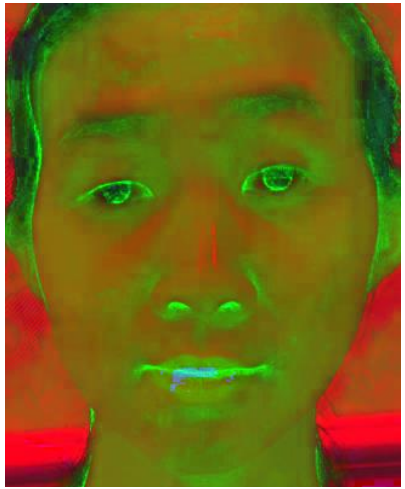
Threshold =40

Threshold =40

Threshold =20



Xác định vùng da bằng lấy ngưỡng - HSV



Ảnh trong
không gian HSV

Ảnh vùng da
được phát hiện

Chọn ngưỡng

(0.0, 60.0, 60.0)
(100.0, 255.0, 255.0)

(0.0, 75.0, 75.0)
(100.0, 255.0, 255.0)

(0.0, 90.0, 90.0)
(100.0, 255.0, 255.0)

Làm mịn khuôn mặt bilateralFilter()

cv2. bilateralFilter() dùng để làm mịn hình ảnh và giảm nhiễu, trong khi vẫn giữ được các cạnh.

(img, 15, 30, 30)

(img, 15, 70, 70)

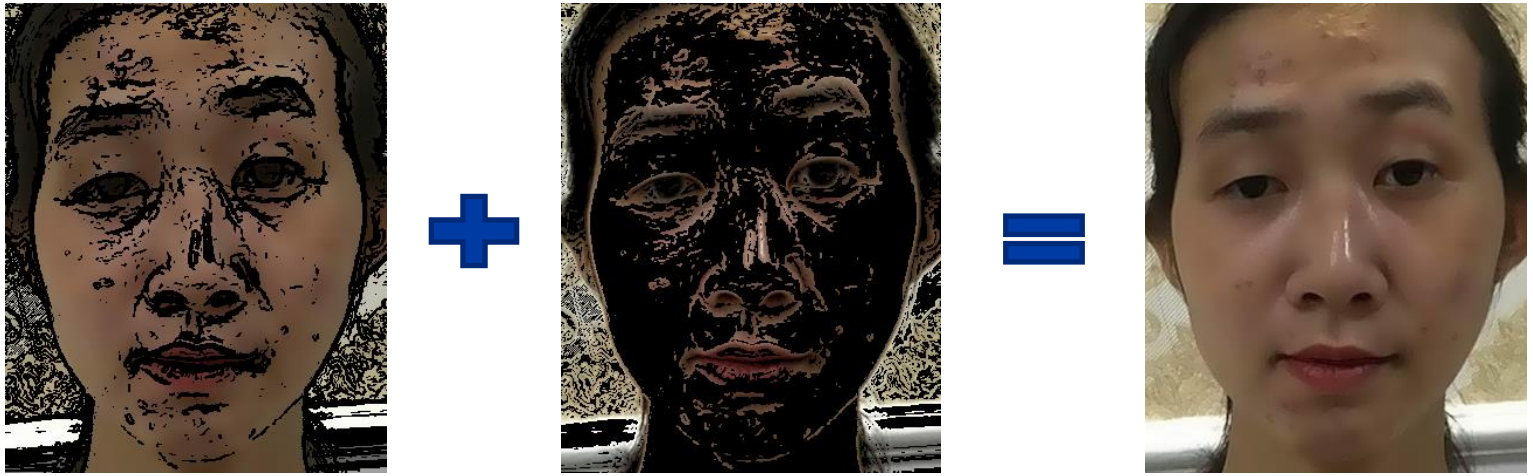
(img, 15, 100, 100)

Xác định thành phần ảnh

Bước 1

Bước 2

Xác định thành phần ảnh



Bước 3

Kết quả làm mịn da – chọn ngưỡng

bilateralFilter()

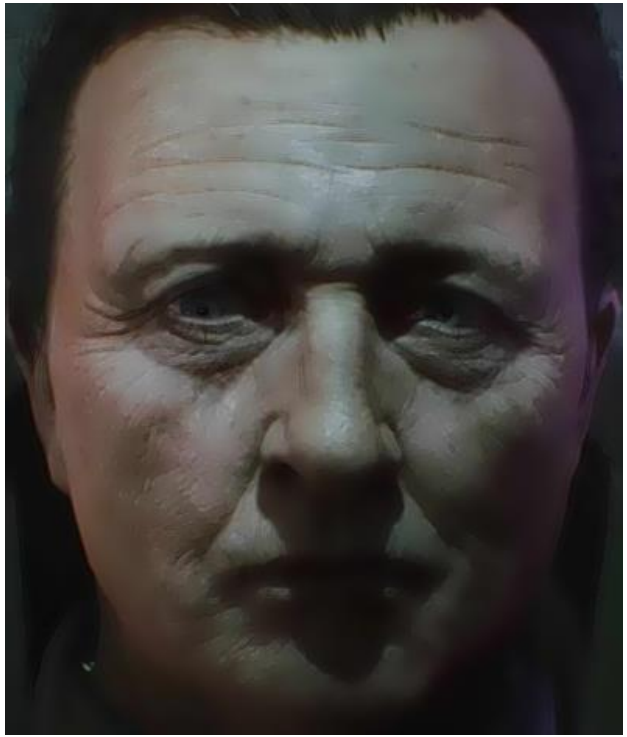
So sánh

Sobel

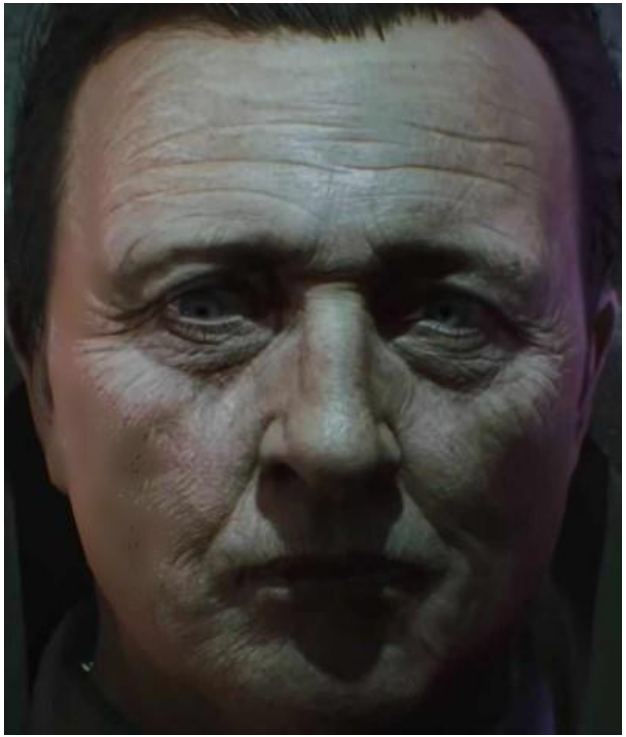
Chọn ngưỡng

So sánh

Sobel



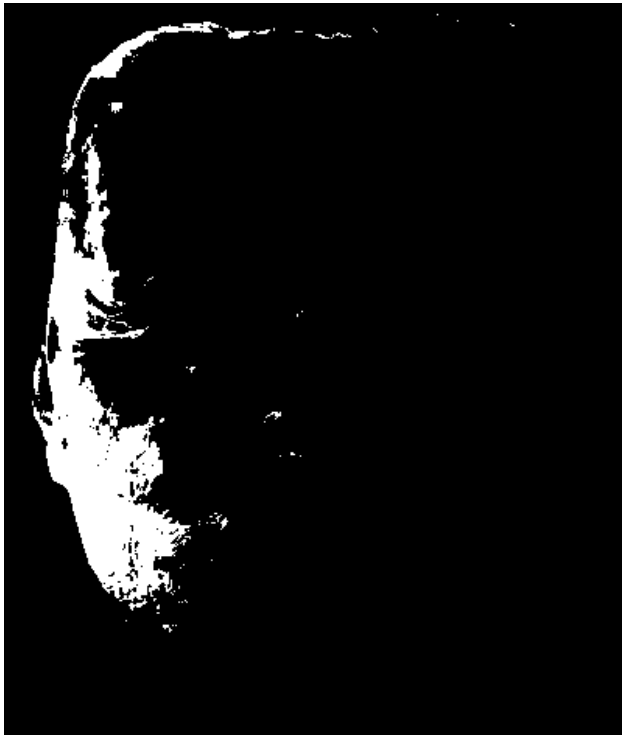
Chọn ngưỡng



So sánh tìm vùng da

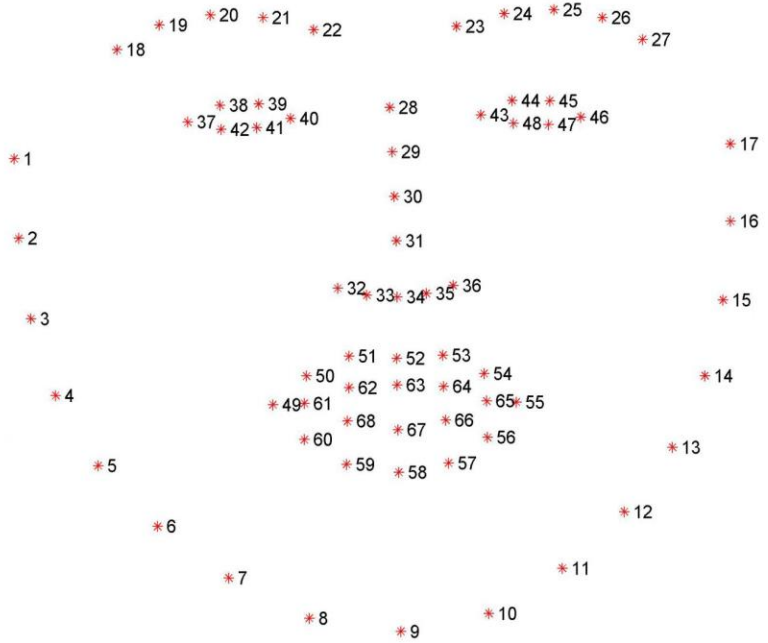
Sobel

Chọn ngưỡng



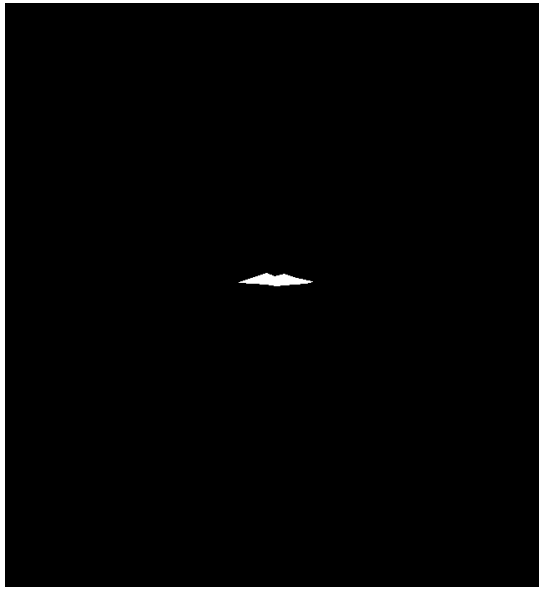
Trang điểm trên khuôn mặt

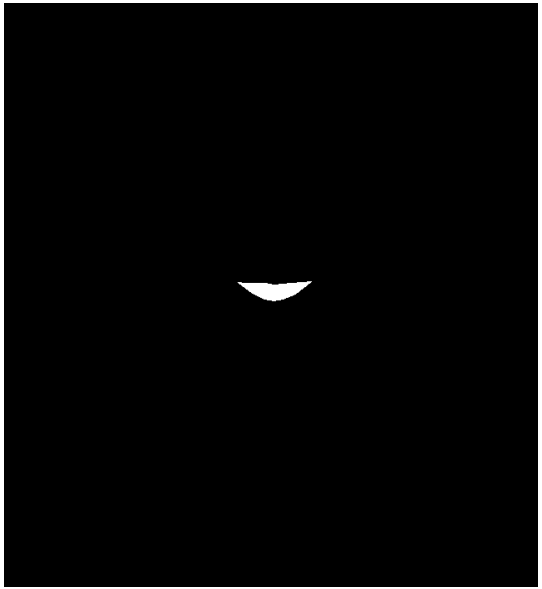
- Sử dụng các điểm trên khuôn mặt (facial landmarks) để xác định tọa độ của môi
- Vị trí của môi từ điểm thứ 49 → 68
- Phần môi trên bao gồm các điểm : 49→55, 65←61
- Phần môi dưới bao gồm các điểm: 61, 68←65, 55→ 60,49



Trang điểm trên khuôn mặt

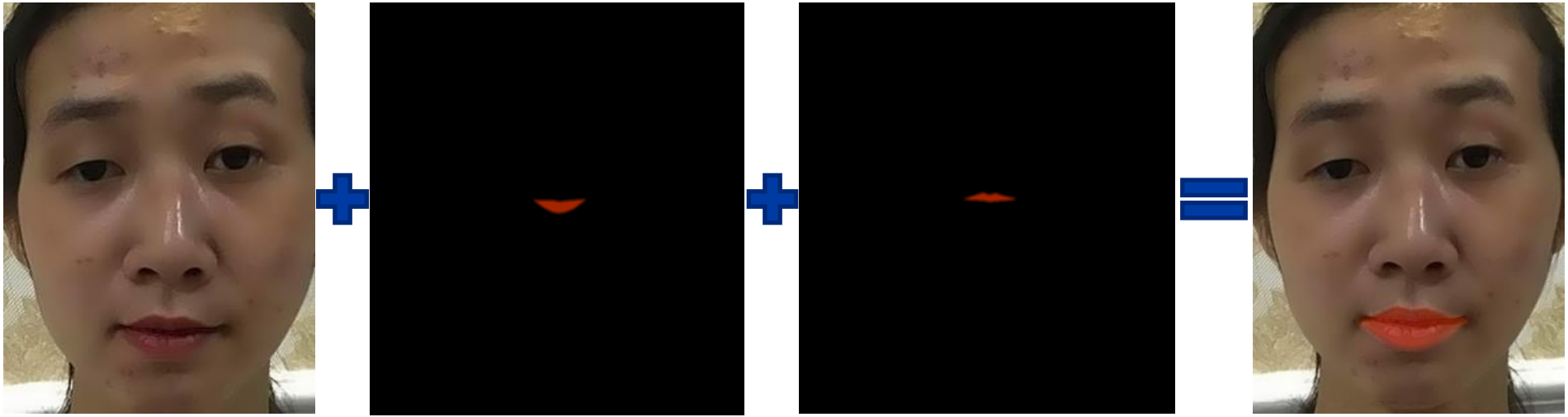
- Sử dụng pre-trained model phát hiện điểm trên khuôn mặt được tích hợp bên trong thư viện Dlib
- Tách phần môi trên và môi dưới thành 2 layer





Trang điểm trên khuôn mặt

- Thay đổi màu cho 2 layer môi và thêm chúng vào ảnh gốc



Nguồn tham khảo

- [1] - OpenCV - https://docs.opencv.org/tutorial_py_filtering.html
- [2] - 5starkarma - https://github.com/5starkarma
- [3] - Pyimagesearch - https://www.pyimagesearch.com/deep-learning-opencvs-blobfromimage
- [4] - Pham Dinh Khanh - https://phamdinhkhanh.github.io/faceNet.html
- [5] - OpenFace - https://cmusatyalab.github.io/openface
- [6]- Face Beautification and Color Enhancement with Scene Mode Detection - https://www.csie.ntu.edu.tw/~fuh/personal/FaceBeautificationandColorEnhancement.A2-1-0040.pdf

Thanks

Do you have any questions?