**Checkin bằng nhận diện khuôn mặ**t

1. Bài toán:

Bài toán xuất phát từ nhu cầu thực tế về việc sử dụng công nghệ nhận diện khuôn mặt để chấm công nhằm đẩy nhanh quá trình chấm công và tiết kiệm thời gian cho nhân viên khi ra vào công ty. Song song với việc quản lí nhân viên, hệ thống còn hướng tới nhận diện và quản lí khách ra vào công ty.

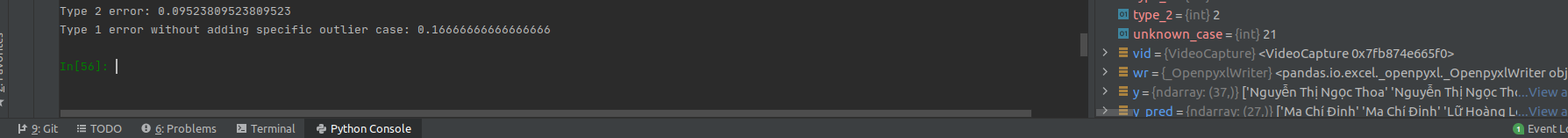
1. Phương pháp thực hiện:
   1. Phần cứng:

Các kết quả thu được đều sử dụng webcam với độ phân giải 720p.

* 1. Chi tiết:

Hiện tại, em đang làm theo hai hướng, một là quết khuôn mặt thành mô hình 2D rồi so sánh với cá mô hình đã biết để ra kết quả cuối, hai là tái tạo lại mô hình 3D từ khung hình để dựng được các vector mô tả khuôn mặt và so sánh với dữ liệu mô tả gốc để nhận diện. Mỗi cách làm đều có ưu, nhược điểm riêng, nhưng nhìn chung là đều đáp ứng về mặt độ chính xác.

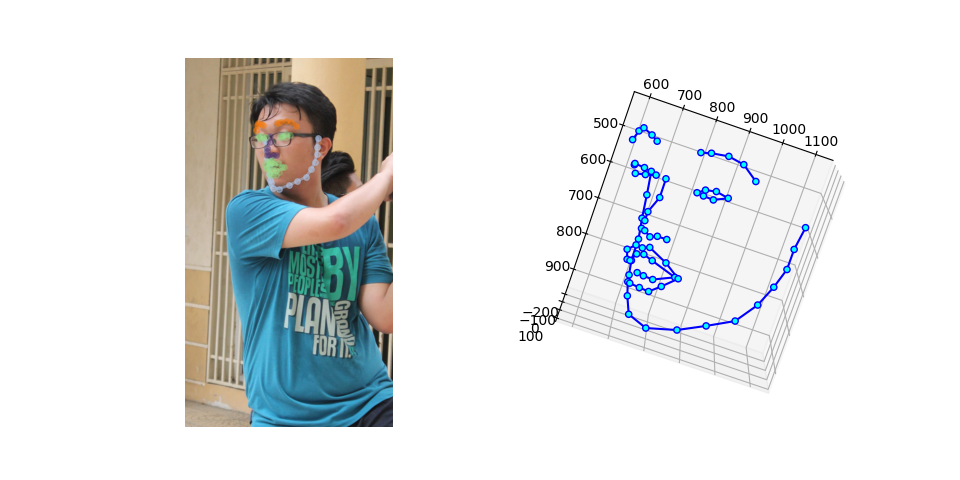
1. Thư viện hỗ trợ:
   1. Các thư viện dùng chung cho cả 2 cách:
      * Opencv: kiểm soát input/output của video, dữ liệu đàu vào
      * scikit-image: thây thế opencv trong một số dữ liệu đầu vào vì thứ tự lưu kênh màu của opencv (bgr)
      * scikit-learn: tạo và train model dự đoán (bài toán classification)
      * pandas: quản lí và preprocess dữ liệu gốc
   2. Riêng của mô hình 2D:
      * [face\_recognition](https://github.com/ageitgey/face_recognition)
   3. Riêng của mô hình 3D:
      * [face\_aglinment](https://github.com/1adrianb/face-alignment)
2. Quy trình đang thực hiện
   1. Với mô hình 2D
      * Quết toàn bộ tập dữ liệu gốc, lấy một vector đặc tả cho mỗi bức ảnh qua thư viện face\_recognition ở trên, lấy tên folder chứa là nhãn cho bài toán classification
      * Train model phân loại (classification) cho tập dữ liệu gốc ở trên
      * Lấy dữ liệu đầu vào từ video feed của webcam, xử lí một khung hình mỗi hai khung đê tiết kiểm tài nguyên và đẩy nhanh tốc độ xử lí
      * Coi từng khung hình dược chọn là một bức ảnh, cho khung hình đi qua hàm lấy đặc tả như với dữ liệu gốc.
      * Sử đụng mô hình đã train ở trên để dự đoán các khung hình đã được xử lí (vector chứa các đặc tả (128 đặc tả))
      * Recheck lại khoảng cách giữa các vector đặc tả của hình cần dự đoán với hình gốc của class được dữ đoán để loại bỏ các case có confidence thấp
      * In tên của các class đã nhận diện được ở khung hình không phải xử lí
   2. Với mô hình 3D
      * Xác định một số điểm có thể phân biệt như độ dài xương quai hàm, góc đọ cằm, độ rộng hộp sọ, góc lệch sống mũi, đọ dày môi để lập list cac đặc trưng
      * Với bộ dư liệu gốc, dựng mô hình 3D, tính toán các giá trị cửa đặc trung
      * Train model phân loại (classification) cho tập dữ liệu gốc ở trên
      * Coi từng khung hình dược chọn là một bức ảnh, cho khung hình đi qua hàm lấy đặc tả như với dữ liệu gốc.
      * Sử đụng mô hình đã train ở trên để dự đoán các khung hình đã được xử lí . Cho liên tiếp 10 khung hình thực hiện bước này. lấy trung bình dự đoán của 10 khung hình
      * Nếu confidence vượt ngưỡng chắc chắn, trả tên class, nếu thấp hơn ngưỡng activate thì trả không biết, nếu nằm giữa thì so sánh với các class có khả năng tương đối cao còn lại để trả 1 tên với confidence trung bình hay một tập hợp các tên.
      * In ra tên class.
3. Kết quả:
   1. Với demo 2D:
      * Chính xác: 85% với video feedback, trong đó lỗi loại 2 tương đối thấp, khoảng hơn 1% tổng case
      * Tốc độ: tương đối nhanh, gần hệ thống thời gian thực
      * Thời gian tạo bộ dữ liệu chuẩn từ dữ liệu gốc: tương đối nhanh chưa tới 1 giây cho ảnh
   2. Với demo 3D:
      * Chính xác: 82% với video, trong đó lỗi loại 2 chiếm tỉ lệ khoảng 1 % tổng case
      * Tốc độ: chậm, thời gian xử lí 1 khung hình vào khoảng 4-5 giây khi không xử lí thông qua gpu
      * Thời gian tạo bộ dữ liệu chuẩn từ dữ liệu gốc: chậm hơn rất nhiều so với cách trên

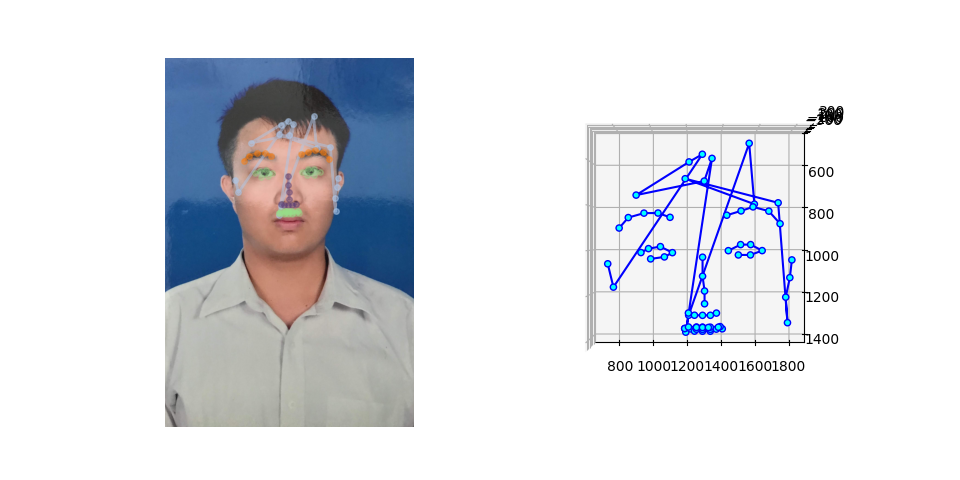


1. Ưu, nhược điểm của hai phương án

|  |  |  |
| --- | --- | --- |
|  | 2D | 3D |
| Ưu điểm | * Nhanh * Dễ dùng * Dễ update dữ liệu chuẩn * Không phụ thược quá nhiều vào điều kiện ánh sáng, chất lượng video đầu vào | * Dễ dùng * Hạn chế được lỗi loại 2 * Nhìn chung là cho kết quả chính xác ơn với tập dữ liệu đủ lớn * Phỏng doán được cả khuôn mặt không chính diện |
| Nhược điểm | * Phụ thuộc vào một số thư viện khó cài trên môi trường Windows. ví dụ như dlib * Qua thử nghiệm, có hiện tượng giật khung hình khi xử lí. Hiện tượng này xảy ra ít hơn trên Unix * Với các góc lệch , mô hình giảm độ chính xác rõ rệt | * Chậm * Việc dựng model càn nhiều tài nguyên * Việc update/ thêm từng record một cho bộ dữ liệu sẽ khó khả thi * Phụ thuộc nhiều vào chất lượng camera và điều kiện ánh sáng |

1. Hạn chế của demo hiện tại
   * Với mô hình 3D, hiện tại chưa hỗ trợ checkin nhiều người trên một frame
   * Với mô hình 3D, cái đặc trưng được xác định bằng tay, để thực hiện việc xác định đặc trưng cho dữ liệu lớn sẽ rất mất thời gian
   * Với mô hình 2D, thư viện không chính thức hỗ trợ mội trường Windows dẫn tới việc cài đặt trên môi trường này phức tạp hơn
   * Với cả hai model, dữ liệu hiện nay đang được lấy ra từ camera không chuyên dụng





1. Đề xuất cách làm cho tập dữ liệu đủ lớn

Phương pháp này sẽ sử dụng để xây dựng một eigenface cho các class. Mỗi eigenface là một "khung" của mặt với hình dáng quan trọng và bỏ qua các chi tiết cụ thể như màu sắc.

* 1. Số lượng ảnh của mỗi đối tượng:
     + Tối thiểu mỗi đối tượng sẽ có 40-50 tấm ảnh
     + Tổng số lượng ảnh gốc tối thiểu phải hơn cạnh dài nhất của ảnh có kích thước lớn nhất
  2. Yêu cầu về góc độ, ánh sáng, độ tương phản:
     + Góc chính diện hoặc lệch tối đa 5 độ, nhìn thấy rõ toàn bộ mặt
     + Độ tương phản càng cao càng tối
     + Không nhất thiết phải là ảnh màu
  3. Kết quả demo với tập dữ liệu lfw\_people:
     + Code có thể xem qua ở: [eigenface notebook demo](https://colab.research.google.com/drive/118rFkbftdYHHvaJfto7mQa7cZbhYOTEY?usp=sharing)
     + accuracy 0.79
     + marco avg 0.77
     + weighted avg 0.79
  4. Ưu, nhược điểm của phương pháp này
     + Yêu cầu dữ liệu rất lớn
     + Phụ thuộc vào chất lượng ảnh
     + Qua trình chọn tham số cho PCA phức tạp

