**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**TRỊNH HOÀNG MINH**

**PHAN VĂN TÂN**

**KHÓA LUẬN TỐT NGHIỆP**

**NHẬN DIỆN CẢM XÚC KHUÔN MẶT THEO THỜI GIAN THỰC BẰNG WEBCAM**

**CỬ NHÂN NGÀNH KHOA HỌC MÁY TÍNH**

**TP. HỒ CHÍ MINH, 2016**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**TRỊNH HOÀNG MINH - 12520638**

**PHAN VĂN TÂN - 12520971**

**KHÓA LUẬN TỐT NGHIỆP**

**NHẬN DIỆN CẢM XÚC KHUÔN MẶT THEO THỜI GIAN THỰC BẰNG WEBCAM**

**KỸ SƯ/ CỬ NHÂN NGÀNH <TÊN NGÀNH>**

**GIẢNG VIÊN HƯỚNG DẪN**

**TS. NGUYỄN THANH BÌNH**

**TP. HỒ CHÍ MINH, 2016**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.

**LỜI CẢM ƠN**

Nhóm chúng em xin gửi lời cảm ơn sâu sắc đến thầy - TS. Nguyễn Thanh Bình bởi sự động viên và hướng dẫn tận tình của thầy. Mặc dù vẫn còn nhiều thiếu sót trong phong cách làm việc lẫn kiến thức chuyên môn nhưng thầy vẫn tận tình chỉ bảo, nhóm vô cùng biết ơn thầy. Nhờ thầy nên nhóm chúng em mới có thể hoàn thành luận văn này.

Nhóm cũng xin gửi lời cảm ơn đến các thầy cô trong khoa Khoa Học Máy Tính của trường Đại học Công Nghệ Thông Tin TP.HCM đã tận tình dạy dỗ, chỉ bảo kiến thức quý báu giúp em hoàn thành khóa học và làm nền tảng cho nghiên cứu của em.

Vì thời gian làm luận văn có hạn và trình độ còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của các thầy cũng như là của quý độc giả để luận văn này hoàn thiện hơn nữa.

TPHCM, ngày 30 tháng 06 năm 2016

Sinh viên thực hiện

Trịnh Hoàng Minh

Phan Văn Tân

MỤC LỤC

[Chương 1. TỔNG QUAN 2](#_Toc455137816)

[1.1. Đặt vấn đề 2](#_Toc455137817)

[1.2. Các nghiên cứu liên quan 2](#_Toc455137818)

[1.3. Mục tiêu đề tài, phạm vi bài toán 3](#_Toc455137819)

[1.4. Bố cục luận văn 3](#_Toc455137820)

[Chương 2. CƠ SỞ LÝ THUYẾT 5](#_Toc455137821)

[2.1. Thuật toán SVM (Support Vector Machines) 5](#_Toc455137822)

[2.1.1. Tổng quan 5](#_Toc455137823)

[2.1.2. SVM tuyến tính (Linear SVM) 5](#_Toc455137824)

[2.1.2.1. Hard-margin 5](#_Toc455137825)

[2.1.2.2. Soft-margin 7](#_Toc455137826)

[2.1.3. Kernel method 8](#_Toc455137827)

[2.1.3.1. Định nghĩa 8](#_Toc455137828)

[2.1.3.2. Điều kiện Mercer 9](#_Toc455137829)

[2.1.3.3. Radial basis function (RBF) 9](#_Toc455137830)

[2.2. Multi-class SVM 10](#_Toc455137831)

[2.2.1. One-versus-all 10](#_Toc455137832)

[2.2.2. One-versus-one 10](#_Toc455137833)

[2.3. Cải tiến model 11](#_Toc455137834)

[2.3.1. Tối ưu hóa các tham số 11](#_Toc455137835)

[2.3.2. Giảm chiều thuộc tính 11](#_Toc455137836)

[Chương 3. [inf] 13](#_Toc455137837)

[3.1. Giới thiệu về ứng dụng 13](#_Toc455137838)

[3.1.1. Tính năng nhận diện cảm xúc 13](#_Toc455137839)

[3.1.2. Tính năng ghi hình 13](#_Toc455137840)

[3.2. Thư viện sử dụng 13](#_Toc455137841)

[3.2.1. Thư viện dlib 13](#_Toc455137842)

[3.2.2. Thư viện libSVM 13](#_Toc455137843)

[3.3. Thu thập dữ liệu 14](#_Toc455137844)

[3.4. Tiền xử lý dữ liệu 14](#_Toc455137845)

[3.4.1. Tiền xử lý 14](#_Toc455137846)

[3.4.2. Trích xuất thuộc tính 15](#_Toc455137847)

[3.5. Xây dựng SVM model 16](#_Toc455137848)

[3.6. Kết quả thử nghiệm 16](#_Toc455137849)

[Chương 4. KẾT LUẬN 17](#_Toc455137850)

**DANH MỤC HÌNH VẼ**

**DANH MỤC BẢNG**

DANH MỤC TỪ VIẾT TẮT

LỜI NÓI ĐẦU

Cùng với sự phát triển không ngừng của Trí tuệ nhân tạo, con người đang hướng đến việc xây dựng những robot không chỉ có khả năng phân tích và dự đoán ưu việt hơn con người mà còn cần có khả năng giao tiếp tốt giữa các robot với nhau và với cả con người. Một trong những điểm thiết yếu để robot có thể giao tiếp với con người một cách thoải mái đó chính là việc biểu lộ cảm xúc khuôn mặt sao cho phù hợp với đối tượng giao tiếp. Hiện tại cũng có nhiều bài báo nghiên cứu về vấn đề này nhưng vẫn chưa giải quyết vấn đề một cách thỏa đáng.

Trong luận văn này, chúng tôi sẽ đưa ra một hướng tiếp cận mới nhằm giải quyết bài toán này: đó là thông qua các facial landmark ta sẽ thiết lập mô hình huấn luyện để nhận diện cảm xúc, từ đó tiến hành xây dựng ứng dụng nhận diện với mục tiêu là có thể thực thi thời gian thực.

# TỔNG QUAN

## Đặt vấn đề

Trong khoa học Trí tuệ nhân tạo đặc biệt là robot, việc tăng cường khả năng giao tiếp với con người của robot là hệ quả tất yếu. Một robot với khả năng giao tiếp giống người thật sẽ giúp cho công việc giữa người và máy có thể thuận tiện và suôn sẻ hơn. Để đạt được mục đích đó, ta cần phải tạo điều kiện giúp cho robot phân tích được những biểu lộ cảm xúc của con người thông qua giọng nói, sóng não, nét biểu lộ trên khuôn mặt, v.v... Một trong những hướng tiếp cận thuận tiện nhất đó chính là thông qua những thể hiện trên khuôn mặt. Để đạt được mục tiêu nói trên, ta phải đảm bảo việc nhận diện cảm xúc phải có được độ chính xác cao và tốc độ nhận diện chấp nhận được. Nhóm chúng em đề xuất ra 1 hướng tiếp cận mới nhằm giải quyết vấn đề đó.

## Các nghiên cứu liên quan

Theo như nghiên cứu của Pantic & Rothkrantz [0] thì nhóm ông cho rằng để giải quyết bài toán nhận diện cảm xúc khuôn mặt ta cần phải trải qua 3 quá trình: nhận diện khuôn mặt, trích xuất dữ liệu từ khuôn mặt thu được và cuối cùng là phân loại dạng cảm xúc.

Những nghiên cứu từ trước đến nay đều mặc định sẵn rằng trong 1 ảnh hoặc chuỗi ảnh đang được xử lý thì ta có được 1 số thông tin về vị trí của frontal face. Nhóm nghiên cứu của Viola & Jones [1] đã sử dụng thuật toán Adaboost để thực hiện tìm kiếm vét cạn thông qua các sub-window với các kích cỡ khác nhau của ảnh gốc nhằm xác định được vị trí chính xác của khuôn mặt. Một điểm chưa mạnh của nghiên cứu trên là tốc độ xử lý khá chậm (15FPS). V.Kazemi và J.Sullivan [2] đã đưa ra được 1 hướng tiếp cận tốt hơn, đó chính là sử dụng cây hồi quy để xác định được vị trí chính xác của khuôn mặt kèm với các landmark với tốc độ xử lý cao.

Về vấn đề trích xuất dữ liệu, nhóm của Cohn [3] đầu tiên khoanh vùng các điểm feature trong frame ảnh đầu tiên của chuỗi ảnh bằng tay, sau đó sử dụng hierarchical optical flow để track chuyển động của các cửa sổ nhỏ xung quanh các điểm đó xuyên suốt các frame. Những vector chuyển vị tính được cho mỗi landmark giữa frame khởi đầu và đỉnh điểm thể hiện thông tin thu được về biểu cảm khuôn mặt.

Công đoạn cuối cùng của quá trình nhận diện cảm xúc khuôn mặt đó là việc phân loại cảm xúc thông qua một cách sắp xếp cụ thể. Đa số các công trình nghiên cứu hiện nay đều chia cảm xúc khuôn mặt thành 6 loại chính: giận dữ, ghê tởm, sợ hãi, vui vẻ, buồn rầu và ngạc nhiên theo như nhóm nghiên cứu của P. Ekman [4]. Nhóm của ông đã thiết lập hệ thống Facial Action Coding System (FACS) cho những biểu hiện trên khuôn mặt thông qua 1 bộ 44 cử chỉ khuôn mặt được gọi là Action Units (AUs).

## Mục tiêu đề tài, phạm vi bài toán

Nhóm sẽ sử dụng thuật toán nhận diện facial landmark của V.Kazemi và J.Sullivan [2] để tối ưu hóa thời gian nhận diện các facial landmark. Sau khi tiền xử lý landmark thu được nhóm sẽ tính toán 1 set gồm 105 feature kèm với các label. Trong nghiên cứu này thì nhóm chỉ nhận dạng frontal face với 3 dạng cảm xúc cơ bản: tích cực (positive), tiêu cực (negative) và bình thường (neutral). Sau khi thử nghiệm qua các model máy học thì nhóm quyết định sử dụng thuật toán SVM để phân loại vì tính hiệu quả và tốc độ xử lý của nó phù hợp với mục tiêu của đề tài.

## Bố cục luận văn

**Chương 1:** Trình bày cách thu thập dữ liệu. Bên cạnh việc thu thập dữ liệu cho ứng dụng chính, nhóm cũng sẽ trình bày những thuật giải tiền xử lý dữ liệu bao gồm: tiền xử lý ảnh, tính toán vị trí các facial landmark.

**Chương 2:** Trình bày cách trích xuất feature từ vị trí các facial landmark thu được, từ đó gán nhãn cho các feature vector tạo thành bộ dữ liệu cho việc training.

**Chương 3:** Giới thiệu sơ lược về cơ sở lý thuyết sử dụng trong ứng dụng *EMO* : Support Vector Machines (SVM). Đây sẽ là nền tảng kiến thức cho việc xây dựng các mô hình học để dự đoán cảm xúc thể hiện trên khuôn mặt của người dùng.

**Chương 4:** Trình bày quá trình xây dựng ứng dụng và kết quả thực nghiệm

**Chương 5:** Trình bày những ưu khuyết điểm, hướng phát triển và kết luận của luận văn

# CƠ SỞ LÝ THUYẾT

## Thuật toán SVM (Support Vector Machines)

### Tổng quan

Support Vector Machines (SVM) là một kĩ thuật học có giám sát được dùng để phân lớp hoặc phân tích hồi quy. Bắt nguồn từ lý thuyết học thống kê (Statistical Learning Theory) được phát triển bởi nhóm nghiên cứu của V. Vapnik [5], SVM được xây dựng dựa trên nguyên tắc giảm thiểu rủi ro (Structural Risk Minimization).

Với kĩ thuật SVM không gian dữ liệu đầu vào sẽ được ánh xạ vào không gian đặc trưng, và từ đó ta sẽ xác định được siêu phẳng phân chia tối ưu.

### SVM tuyến tính (Linear SVM)

Ta có tập S gồm *e* các mẫu học

với một vector đầu vào *n* chiều thuộc 2 lớp với nhãn ứng với *yi = 1* hoặc *yi = -1*.

Đối với bài toán phân chia dữ liệu tuyến tính, ta cần tìm siêu phẳng maximum- margin có thể tách riêng nhóm điểm có *yi = 1* với *yi = -1*, tức là tìm siêu phẳng sao cho khoảng cách giữa nó và vector gần nhất từ cả 2 nhóm là lớn nhất.

#### Hard-margin

Với điều kiện data để học có thể phân chia tuyến tính, ta chọn 2 siêu phẳng song song phân chia 2 lớp sao cho khoảng cách giữa chúng là lớn nhất. Miền tạo bởi 2 siêu phẳng đó ta gọi là margin, siêu phẳng maximum-margin là siêu phẳng nằm giữa chúng.

Ta có thể mô tả 2 siêu phẳng ấy bằng phương trình:

và

Trong hình học giải tích, khoảng cách giữa 2 siêu phẳng trên là . Do đó để cực đại hóa khoảng cách giữa 2 siêu phẳng trên ta cần phần giảm thiểu . Đồng thời để các điểm không rơi vào trong margin, ta thêm ràng buộc rằng với mọi *i* ta có

nếu *yi = 1*

hoặc

nếu *yi = -1*

Những ràng buộc này đảm bảo các điểm đều nằm đúng vào đúng vùng class của nó. Ta có thể viết lại như sau:

với mọi (1)

Tổng hợp lại ta có bài toán tối ưu như sau:

“Tìm cực tiểu sao cho thỏa với mọi ”

Vector pháp tuyến xác định chiều của siêu phẳng, còn giá trị ngưỡng *b* xác định khoảng cách giữa siêu phẳng và gốc tọa độ. Từ 2 kết quả trên ta thu được bộ phân loại . Thêm một điểm nữa ta cần lưu ý, maximum-margin được xác định bởi những nằm kề nó. Chúng được gọi là các support vectors.



**Hình 2.1** Siêu phẳng maximum-margin và các margin của 1 bộ SVM được học từ các mẫu được gán nhãn thành 2 lớp. Những mẫu nằm trong margin được gọi là support vectors

#### Soft-margin

Trong trường hợp dữ liệu không thể được phân chia tuyến tính, chúng ta sử dụng hàm *hinge loss*

Nếu thỏa điều kiện (1) thì kết quả trả về của hàm này bằng 0. Ngược lại thì giá trị của nó sẽ tỉ lệ thuận với khoảng cách từ margin. Từ đó ta cần phải giảm thiểu

với λ thể hiện tradeoff kích thước của margin nhằm đảm bảo thỏa điều kiện (1). Với λ thích hợp, soft-margin SVM cũng giống như hard-margin SVM nếu dữ liệu đầu vào có thể phân loại tuyến tính, ngược lại nó vẫn có thể học được 1 cách phân loại khả thi khác.

Hay nói khác hơn là soft-margin cho phép có một vài điểm data được gán sai nhãn.

### Kernel method

Được sử dụng rộng rãi trong lĩnh vực máy học, các kĩ thuật kernel methods có thể mở rộng các thuật toán học, trong bài viết này là SVM có thể phân loại được trường hợp dữ liệu không thể phân loại tuyến tính được. Ở hình 2.2 nêu lên cho ta 1 ví dụ về việc mọi siêu phẳng đều chứa cả 2 class. Với việc sử dụng 1 hàm vô cùng phức tạp để phân loại nó là khả thi, nhưng thay vào đó ta chỉ cần dùng 1 hàm ánh xạ Ф biến không gian đầu vào *X* thành 1 không gian ℍ với số chiều lớn hơn và có thể phân loại tuyến tính được.

 (a) (b)

**Hình 2.2** Trường hợp không thể phân chia tuyến tính với 2 class ⚫ và ◼. Với hình (a) ta không tìm được siêu phẳng nào có thể tách riêng 2 class được. Thay vào đó ta sử dụng 1 ánh xạ không tuyến tính ở (b)

Trong thực tế, số chiều của ℍ có thể vô cùng lớn. Cụ thể là trong bài toán phân loại tài liệu, với bộ từ điển chỉ với 100,000 từ, số chiều của không gian feature ℍ đạt đến 1015, điều đó khiến cho việc tính toán vô cùng tốn kém. Để giải quyết vấn đề trên ta dùng *kernel methods* vốn dựa trên *kernels* hoặc *hàm kernels*.

#### Định nghĩa

Hàm được gọi là 1 kernel của *X*

Ta định nghĩa 1 hàm kernel *K* sao cho với 2 điểm bất kì bằng với inner product của vector Ф(*x*) và Ф(*y*):

cho ánh xạ đến 1 không gian Hilbert được gọi là không gian feature. Điểm mạnh của kernel *K* là độ hiệu quả của nó: ta có thể tính được *K* với hiệu quả cao hơn so với việc tính toán Ф và inner product trong ℍ. Bên cạnh đó ta không cần định nghĩa tường minh hoặc tính toán 1 hàm ánh xạ Ф. Ta có thể chọn kernel *K* tùy ý miễn tồn tại hàm Ф, nghĩa là *K* phải thỏa mãn điều kiện Mercer.

#### Điều kiện Mercer

Với là 1 tập compact và hàm là 1 hàm liên tục và đối xứng. Ta có [K admits a uniformly convergent expansion of the form]

với *an > 0* khi và chỉ khi với bất kì hàm bình phương khả tích thỏa:

#### Radial basis function (RBF)

Với mọi hàng số , một *Gaussian kernal* hoặc *Radial basis function (RBF)* là kernel *K* định nghĩa trong tập ℝN bởi:

Gaussians kernels nằm trong số những kernel được sử dụng rộng rãi nhất trong các ứng dụng. Trong quá trình huấn luyện model học ta cần quan tâm đến 2 thông số quan trọng nhất chính là và C. Thông số gamma cho ta biết ảnh hưởng của 1 training sample với xung quanh nó, hiểu đơn giản nó là nghịch đảo của bán kính của các sample được lựa chọn làm support vector của model. Gamma tradeoff giữa tốc độ xử lý và độ hiệu quả, với giá trị càng cao thì tốc độ xử lý càng nhanh nhưng chưa chắc hiệu quả và ngược lại. Còn với thông số C tradeoff giữa việc phân loại nhầm giữa các sample và độ phức tạp của hàm kernel. C thấp sẽ cho kernel với độ phức tạp thấp trong khi C cao dễ dẫn đến hiện tượng overfit.

## Multi-class SVM

Trong trường hợp chúng ta phải phân loại nhiều lớp thì ta giải quyết vấn đề trên bằng cách vẫn sử dụng kĩ thuật SVM nguyên thủy chia không gian dữ liệu thành 2 phần và quá trình này lặp đi lặp lại nhiều lần. Ta sẽ đề cập đến 2 kĩ thuật thường sử dụng nhất đó là *one-versus-one (OVO)* và *one-versus-all (OVA)*

### One-versus-all

Với tập mẫu training , bằng kĩ thuật *one-versus-all (OVA)* hoặc *one-versus-the-rest* ta chuyển đổi bộ phân lớp đa lớp thành phân lớp nhị phân. Kĩ thuật này bao gồm việc học *k* bộ phân lớp nhị phân sao cho phân biệt 1 lớp khỏi tất cả các lớp khác. Với bất kì thu được sau khi train 1 bộ phân lớp nhị phân trên toàn bộ tập mẫu S sau khi gán nhãn 1 cho class *l* và -1 cho tất cả các class còn lại. Với , giả sử *hl* được suyra từ dấu của hàm đánh giá nghĩa là , ta có hàm giả thiết đa lớp được định nghĩa bởi kĩ thuật OVA như sau:

### One-versus-one

Kĩ thuật này sử dụng dữ liệu training để học 1 cách độc lập, với mỗi cặp class phân biệt ta có 1 bộ phân lớp nhị phân tách riêng class *l* và *l’*. Với bất kì , ta thu được bằng cách train 1 bộ phân lớp nhị phân trên 1 tập mẫu con chỉ chứa các điểm được gán nhãn *l* hoặc *l’*, với giá trị trả về là +1 cho class *l’* và -1 cho class *l*. Ta cần train bộ phân lớp và tổng hợp lại để định nghĩa 1 hàm giả thuyết phân lớp multi-class *h*:

Ta có thể hiểu với 1 điểm , nếu ta mô tả kết quả dự đoán như kết quả của các trận đấu trong 1 giải đấu giữa 2 tuyển thủ *l* và *l’*, với cho ta biết *l´* đang hơn điển *l*, class được dự đoán bởi *h* là class có số trận thắng nhiều nhất trong giải đấu.

|  |  |  |
| --- | --- | --- |
|  | Training | Testing |
| OVA |  |  |
| OVO |  |  |

**Bảng 2.1** Bảng so sánh độ phức tạp giữa 2 kĩ thuật OVA và OVO ở phần training và testing. Bảng này được xây dựng dựa trên giả thiết có 1 mẫu training kích cỡ m với m/k điểm cho mỗi class. Giả sử thời gian tiêu tốn cho 1 thuật toán phân lớp nhị phân trên tập mẫu size n là O(nα). Ta tính được thời gian training với kĩ thuật OVO là . ct thể hiện thời gian tiêu tốn cho việc testing của 1 bộ phân lớp

## Cải tiến model

### Tối ưu hóa các tham số

Như đã đề cập ở trên, vấn đề mới nảy sinh chính là tìm cặp (gamma,C) sao cho model thu được là tốt nhất. Ta sử dụng kĩ thuật k-fold cross-validation.

Với k-fold cross-validation, ta chia tập training thành k tập con cùng kích cỡ. Từng tập con sẽ lần lượt được dùng để test trong khi các tập còn lại được dùng để train. Vì thế mỗi phần của training data đều được dùng để dự đoán 1 lần nên độ chính xác của cross-validation cũng chính là độ chính xác của lượng data được dự đoán chính xác. Việc sử dụng cross-validation sẽ giúp ta hạn chế được vấn đề overfit của model. Nghiên cứu của Chih-Wei Hsu, Chih-Chung Chang, và Chih-Jen [6] đã gợi ý bằng cách dùng dãy các số cấp số nhân tăng tiến để tìm ra thông số tốt nhất (VD: )

### Giảm chiều thuộc tính

Bên cạnh việc tăng độ chính xác của thuật toán học ta xem xét đến việc tối ưu tốc độ xử lý của ứng dụng nhằm tăng hiệu quả. Một trong những phương pháp chính đó là giảm đi số lượng thuộc tính ít ảnh hưởng đến việc phân loại và dự đoán của ứng dụng.

# [inf]

## Giới thiệu về ứng dụng

*EMO* là một ứng dụng chạy trên hệ điều hành Ubuntu. Được lập trình trên nền C++, ứng dụng *EMO* đảm bảo việc xử lý webcam real-time diễn ra với độ trễ thấp. Các tính năng của ứng dụng *EMO* bao gồm nhận diện cảm xúc và ghi hình.

### Tính năng nhận diện cảm xúc

Đây là tính năng chính của ứng dụng. Nhóm giới hạn lại ứng dụng chỉ nhận diện cảm xúc của 1 frontal face nhận diện được, kết quả nhận diện sẽ được overlay bên góc trái của cửa sổ thể hiện.

### Tính năng ghi hình

Nhóm thêm tính năng này nhằm mục đích thu thập dữ liệu cho việc huấn luyện mô hình học. Dữ liệu được thu trực tiếp từ webcam của laptop và ghi lại các frame ảnh dạng JPEG.

## Thư viện sử dụng

### Thư viện dlib

Là một thư viện mở được release kèm Boost Software License, dlib là 1 thư viện C++ cross-platform miễn phí được sử dụng rộng rãi. Từ năm 2002 đến nay, dlib được phát triển trong nhiều lĩnh vực như mạng, thread, GUI, máy học, v.v... Lí do chính nhóm chọn thư viện dlib là vì dlib đã implement được thuật toán nhận diện facial landmark với hiệu quả cao, rất cần thiết cho việc xử lý real-time thông qua webcam. Bên cạnh đó dlib cũng có wrapper cho ngôn ngữ Python, 1 ngôn ngữ rất dễ dùng và tiện lợi.

### Thư viện libSVM

Thư viện libSVM cũng là 1 thư viện mở rất phổ biến được phát triển trên nền C++ và port sang các ngôn ngữ khác. Vì khả năng cross-platform của libSVM nên nhóm chọn sử dụng thư viện này để hỗ trợ cho việc training model thuận tiện hơn. Quá trình huấn luyện sẽ được tiến hành bằng Python.

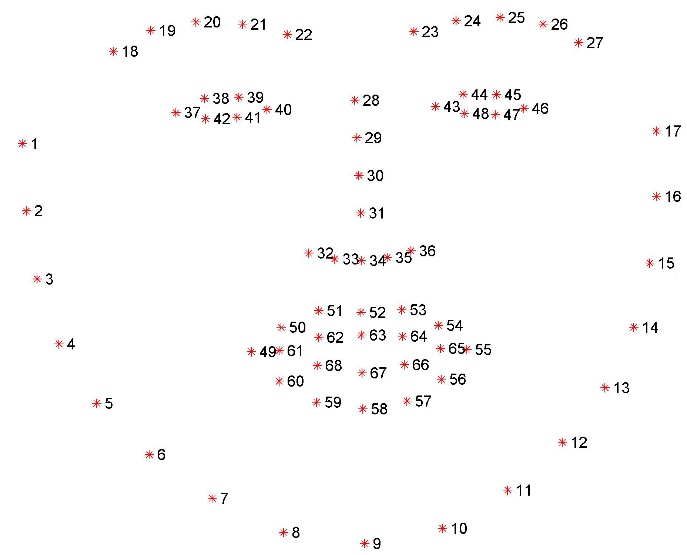
## Thu thập dữ liệu

Dữ liệu nhóm thu được là các ảnh portrait tìm được từ Google với đặc điểm chính đều là frontal face và thể hiện 3 dạng cảm xúc chính mà nhóm nghiên cứu: positive, neutral và negative. Sau khi lọc các ảnh có thể xác định được facial landmark, nhóm thu được tổng cộng khoảng 600 ảnh chia đều cho 3 dạng cảm xúc dùng để xây dựng mô hình huấn luyện.

## Tiền xử lý dữ liệu

### Tiền xử lý

Nhóm sử dụng thuật toán xác định facial landmark của thư viện dlib để thu được vị trí của 68 điểm facial landmark có dạng như sau:



**Hình 3.1** Vị trí của 68 điểm landmark trên khuôn mặt

Trong trường hợp khuôn mặt bị nghiêng, ta tiến hành xoay ảnh quanh tâm ảnh với 1 góc tạo bởi đường thẳng nối giữa 2 điểm 32 và 26 với trục hoành. Sau đó ta cắt ảnh theo bounding box của facial landmark và resize lại với độ rộng 100px.

ooxWord://word/media/image10.png

**Hình 3.2** Quá trình tiền xử lý ảnh

Với tập ảnh thu được, ta tiến hành gán nhãn và trích xuất feature từ các ảnh để xây dựng training dataset. Nhóm sử dụng script của Python để xử lý quá trình này.

ooxWord://word/media/image11.png

**Hình 3.3** Quá trình tiền xử lý dữ liệu

### Trích xuất thuộc tính

Ở đây chúng em dựa trên paper [Novel Method…] để tính toán các thuộc tính có sẵn và sau đó phát triển thêm các thuộc tính khác nữa nhằm mục đích so sánh độ chính xác của thuật toán học.

Bên cạnh 26 thuộc tính có sẵn trong paper, nhóm tụi em phát triển thêm 79 thuộc tính khác cũng dựa trên các 68 landmark thu được bên trên. Chi tiết 79 thuộc tính gồm:

-3 thuộc tính độ mở của miệng và 2 mắt

-18 thuộc tính về khoảng cách và góc của lần lượt các điểm thuộc ½ mặt bên phải đến điểm mép bờ môi phải

-18 thuộc tính về khoảng cách và góc của lần lượt các điểm thuộc ½ mặt bên trái đến điểm mép bờ môi trái

-40 thuộc tính về khoảng cách và góc của tất cả các điểm trên môi đến giao điểm đã xác định trước.

## Xây dựng SVM model

Nhóm đã viết 1 Python script để tiến hành train model bằng libSVM thử nghiệm nhiều thông số (gamma, C) khác nhau dựa theo nghiên cứu của Chih-Wei Hsu, Chih-Chung Chang, và Chih-Jen [6] và tiến hành 5-fold cross-validation. Kết quả thu được là 1 SVM model có độ chính xác [acc] với thông số (gamma, C) là [gamma,C].

## Kết quả thử nghiệm

# KẾT LUẬN

TÀI LIỆU THAM KHẢO

Theo chuẩn IEEE