

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG KHOA CÔNG  
NGHỆ THÔNG TIN 1**

-----



**BÁO CÁO BÀI TẬP LỚN**

**MÔN: HỆ CƠ SỞ DỮ LIỆU ĐA PHƯƠNG TIỆN**

**Đề tài: *Lưu trữ và tìm kiếm video có sự xuất hiện khuôn mặt người  
trong ảnh cho trước***

**Giảng viên : Nguyễn Đình Hóa**

**Nhóm môn học : D17-071**

**Nhóm bài tập : 12**

**Danh sách thành viên nhóm:**

**Nguyễn Thị Hà - B17DCCN192**

**Nguyễn Thùy Linh - B17DCCN378**

**Hà Thị Kim Phụng - B17DCCN492**

**Hà Nội - 2021**

# Mục lục

I. Mô tả hệ thống.....	3
II. Các kỹ thuật trong xử lý video .....	3
1. Kỹ thuật tách shot video, lấy được frame đại diện: .....	3
1.1 Phát hiện cảnh quay có sự thay đổi đột ngột. ....	3
1.2 Phát hiện ranh giới cảnh quay với sự thay đổi dần dần .....	5
1.3 Kỹ thuật tìm frame đại diện cho shot video .....	6
2. Kỹ thuật phát hiện mặt người trong ảnh sử dụng thuật toán Viola-Jones (Haar Cascade) .....	8
3. Kỹ thuật trích rút đặc trưng trong ảnh: .....	10
3.1. Màu sắc .....	10
3.2. Tra cứu ảnh dựa trên kết cấu .....	12
3.3. Truy xuất hình ảnh dựa trên hình dạng.....	14
3.4. Tra cứu ảnh dựa trên đặc trưng bất biến.....	16
III. Hệ thống chương trình: .....	23
1. Đối tượng và cơ sở dữ liệu: .....	23
2. Hoạt động của chương trình: .....	25
2.1. Upload và lưu trữ dữ liệu: .....	25
2.2. Tách shot chọn frame đại diện:.....	26
2.3. Phát hiện mặt người sử dụng Haar Cascade: .....	27
2.4. Sử dụng LBPH để rút trích đặc trưng ảnh: .....	28
2.5. Tìm kiếm video từ ảnh đầu vào .....	30
3. Demo hệ thống.....	31

## I. Mô tả hệ thống

Hệ thống được xây dựng là một hệ thống lưu trữ video, cho phép nhận dạng và tìm kiếm video. Đầu vào là một ảnh, đầu ra là một vài file video có nội dung giống nhất hoặc chứa nội dung của ảnh đầu vào. Nội dung được xét đến là khuôn mặt người.

## II. Các kỹ thuật trong xử lý video

Kỹ thuật xử lý video nhóm tìm hiểu chủ yếu về tìm kiếm video (dựa trên siêu dữ liệu, dựa trên văn bản, dựa trên âm thanh, dựa trên nội dung), tập chung vào tìm kiếm dựa trên nội dung.

### 1. Kỹ thuật tách shot video, lấy được frame đại diện:

**Kỹ thuật tách shot video:** hay còn gọi là phân đoạn cảnh quay video (lia- đoạn cơ sở là gồm các frame liên tục có nhiều đặc chung như mô tả cùng một cảnh, báo hiệu một thao tác máy quay bấm máy, chứa sự kiện mô tả hoạt động của một đối tượng, người dùng có thể lựa chọn như thực thể để chỉ mục, nói chung là chứa cùng nội dung thông tin, mô tả hoạt động liên tục, bị giới hạn bởi 2 chuyển cảnh) :

+ Đối với các băng video có sự biến đổi đột ngột.

+ Đối với biến đổi dần dần.

**Kỹ thuật tìm frame đại diện cho shot video:** sau khi tác lia, cần chọn ra các frame đại diện cho video đó.

#### 1.1 Phát hiện cảnh quay có sự thay đổi đột ngột.

Để phân đoạn được cần phải đo lường sự khác biệt giữa các frame, sau đó lựa chọn ngưỡng T thích hợp để xác định sự khác biệt thế nào là có thể được phân ra.

#### Các đo lường sự khác biệt giữa các khung hình:

- **Cách 1:** (đơn giản nhất ) tính tổng của sự khác biệt pixel-to-pixel giữa các khung hình lân cận. Nếu tổng > ngưỡng T đặt trước, ranh giới ảnh sẽ tồn tại giữa hai khung hình này. Hiệu quả đối với hard cut. Ít hiệu quả khi có đối tượng chuyển động giữa các Frames

$$DP(x, y) = \begin{cases} 1 & , \text{Nếu } |f_1(x, y) - f_2(x, y)| \\ 0 & , \text{ngược lại} \end{cases}$$

$$D(f_1, f_2) = \frac{1}{X \times Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} DP(x, y)$$

+ Trường hợp nếu ảnh màu:

$$D(f_1, f_2) = \sum_{x=0}^X \sum_{y=0}^Y \sum_{i \in \{R, G, B\}} w_i |f_{1i}(x, y) - f_{2i}(x, y)|$$

+ Nhược điểm: không hiệu quả và nhiều phát hiện shot nhầm. Điều này là do hai khung hình trong một lần chụp có thể có sự chênh lệch pixel-to-pixel lớn do chuyển động của đối tượng từ khung này sang khung khác. => khắc phục bằng cách tính độ lệch bởi công thức sau:

$$D(f_1, f_2) = \frac{1}{X \times Y} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \frac{|f_1(x, y) - f_2(x, y)|}{f_2(x, y)}$$

Hampapur gọi ảnh thu được từ độ chênh lệch hiệu chỉnh là ảnh chromatic

- **Cách 2:** Khắc phục cách 1, đo khoảng cách biểu đồ màu (color histogram) giữa các khung lân cận. (thực chất nguyên tắc phương pháp này là chuyển động của đối tượng gây ra sự khác biệt nhỏ về biểu đồ nên áp dụng được):

+ Gọi  $H_i(j)$  biểu thị histogram cho frame  $i$ , trong đó  $j$  là một trong các mức xám có thể có của  $G$ . Khi đó, sự khác biệt giữa khung thứ  $i$  và khung kế là:

$$SD_i = \sum_j |H_i(j) - H_{i+1}(j)|$$

Nếu  $SD_i >$  ngưỡng  $T$  nào đó (được xác định trước), thì là xuất hiện frame mới.

+ Với video màu, kỹ thuật trên sẽ phải tính đến các thành phần màu sắc: đơn giản + hiệu quả thì sẽ so sánh histogram dựa trên mã màu bắt nguồn từ các thành phần  $R$ ,  $G$  và  $B$ .

Trong trường hợp này,  $j$  trong phương trình trên là mã màu chứ không phải mức xám nữa. Để giảm tính toán, sẽ chọn hai hoặc ba bit quan trọng nhất của mỗi thành phần màu để tạo mã màu. Ví dụ: nếu ba bit cho mỗi thành phần được sử dụng, biểu đồ có tổng số 512 ngăn. ( $8 \times 8 \times 8$  vì mỗi ô 8 bit, mà một pixel có 3 thành phần màu)????

- **Cách 3:** là sửa của phương pháp 2. Khoảng cách giữa 2 frame sẽ là:

$$SD_i = \sum_j \frac{(H_i(j) - H_{i+1}(j))^2}{H_{i+1}(j)}$$

Cách này tính tổng sự khác biệt trên cái histogram tiếp theo

**Lựa chọn các giá trị ngưỡng  $T$  sao cho thích hợp:** Đây là điểm then chốt để việc xác định hiệu suất phân đoạn có tốt không.

+ **Cách 1:** Thường, ngưỡng được chọn là giá trị trung bình của chênh lệch khung hình cộng với giá trị dung sai nhỏ

+ **Cách 2:** T lại có thể xác định dựa trên mô hình thống kê cho sự khác biệt giữa khung hình.

Xác định T thế nào thì xác định nhưng nó có thể chấp nhận các biến thể trong các frame riêng lẻ nhưng phát hiện các ranh giới thực tế.

## ***1.2 Phát hiện ranh giới cảnh quay với sự thay đổi dần dần***

- Thực tế, các kỹ thuật cơ bản trên không thể phát hiện ranh giới của cảnh quay khi sự thay đổi giữa các frame diễn ra dần dần (fade-in, fade-out, dissolve, and wipe : hình dung nó giống kiểu hiệu ứng chuyển slide power point). Mặt khác, kỹ thuật trên không xét đến sự phân bố màu sắc theo không gian nên nó sẽ không nhận ra ranh giới giữa hai frame của hai cảnh khác nhau khi dùng color histogram tương ứng. Cần nhiều kỹ thuật khác.

- Kỹ thuật của Zhang et al. đã so sánh sử dụng hai ngưỡng:

+ ngưỡng  $T_b$ : được sử dụng để phát hiện chuyển frame đột ngột

+ ngưỡng  $T_s$  ( $< T_b$ ) để phát hiện ra frame tiềm năng (có nguy cơ) là chuyển đổi dần dần. Nếu sự khác biệt frame-to-frame tích lũy của các frame tiềm năng (cái này tính tổng frame-to-frame differences từ lúc bắt đầu shot đến cái frame này- thầy cho ghi rồi) là lớn hơn  $T_b$ , => lúc này là chuyển sang shot mới

(phương pháp này gọi là so sánh cặp tính toán chênh lệch tích lũy giữa các khung hình trong chuyển cảnh dần dần)

- Nhận xét: khó để xác định chính xác quá trình chuyển đổi dần dần. Boreczky và Rowe đã thực hiện một so sánh: tỉ lệ thành công của việc phát hiện chính xác quá trình chuyển đổi dần dần là dưới 16%.

***Với các kỹ thuật trên có thể áp dụng với histogram cục bộ, hoặc toàn cục với toàn bộ ảnh:***

### **Cục bộ:**

Ý tưởng là, ta sẽ chia khung hình thành  $b$  khối, đánh số từ 1 đến  $b$ . So sánh biểu đồ của các khối tương ứng rồi tính tổng chênh lệch để có kết quả trừ ảnh cuối cùng.

$$D(f_1, f_2) = \sum_{k=1}^b DP(f_1, f_2, k) \quad , \text{ với}$$

$$DP(f_1, f_2) = \sum_{j=0}^G |H_1(j, k) - H_2(j, k)|$$

Trong đó:  $H(j,k)$  là giá trị biểu đồ tại màu (mức xám)  $j$  ứng với khối thứ  $k$ . Nagasaka và Tanaka đã cài đặt thử nghiệm phương pháp thống kê mức xám, so sánh cặp điểm ảnh và phương pháp biểu đồ. Kết quả tốt nhất thu được khi thực hiện chia khung hình thành 16 khối cùng kích thước sử dụng thuật toán với biểu đồ màu cho các khối này và loại bỏ sai lệch lớn nhất để giảm tác động của nhiễu và di chuyển camera và đối tượng.

Hướng tiếp cận khác trong kỹ thuật trừ ảnh dựa vào biểu đồ cục bộ được Swanberg đưa ra. Sự chênh lệch  $DP(f_1, f_2, k)$  giữa các khối được tính bằng cách so sánh biểu đồ màu RGB sử dụng công thức sau:

$$DP(f_1, f_2, k) = \sum_{c \in \{R, G, B\}} \sum_{j=0}^G \frac{(H_1^c(j, k) - H_2^c(j, k))^2}{H_2^c(j, k)}$$

### 1.3 Kỹ thuật tìm frame đại diện cho shot video

Không có sự so sánh kỹ lưỡng giữa các phương pháp và vì vậy khó có thể nói phương pháp nào là tốt nhất. Nói chung, việc lựa chọn phương pháp chọn frame  $r$  là tùy thuộc vào ứng dụng. Dưới đây trình bày 3 phương pháp.

#### a) Phương pháp chọn frame đại diện 1:

- Shot tương đối tĩnh: có thể chọn một làm frame đại diện
- Shot động: là shot có các Pan máy quay hay có các đối tượng chuyển động trong các frame của shot
- \* Việc chọn frame đại diện được thực hiện bởi 2 nhiệm vụ:
  - Nhiệm vụ 1: Xác định tổng số frames chính đại diện cho shot
  - Nhiệm vụ 2: Xác định các frames đại diện trong shot

**Nhiệm vụ 1: Với Xác định tổng số frames đại diện cho shot :** là xác định tổng số frame  $r$  sử dụng cho mỗi shot. Một số phương pháp hay được sử dụng như sau đây:

Một frame $r$ /shot	một frame $r$ /second	một frame $r$ / shot con

<b>Phương pháp 1</b> sử dụng một frame $r$ / shot. Hạn chế của phương pháp này là nó không xem xét độ dài và thay đổi nội dung của shots., nếu shot dài thì điều này không ổn chút nào, shot ngắn thì được.	<b>Phương pháp thứ 2:</b> gán nhiều frame $r$ cho shot tùy theo độ dài của chúng. Nếu độ dài của shot bằng hay nhỏ hơn 1 giây thì chỉ một frame $r$ được gán cho shot. Nếu độ dài của shot dài hơn 1 sec thì chỉ một frame $r$ được gán cho mỗi giây của video. Phương pháp này quan tâm đến độ dài shot nhưng bỏ qua nội dung shot.	<b>Phương pháp thứ 3:</b> chia shot thành shot con (subshot) hay cảnh (scene) và gán một frame $r$ cho mỗi shot con. Các shot con được tách trên cơ sở nội dung video thay đổi. Nội dung được xác định trên cơ sở các vector chuyển động, dòng quang học (optical flow), hoặc độ lệch giữa frame-to-frame,..
---	--	--

**Nhiệm vụ 2: Xác định các frames đại diện trong shot:** Phương pháp chung nhất lựa chọn frame  $r$  cho mỗi segment như sau:

<p><b>Phương pháp 1:</b> Frame thứ nhất của mỗi đoạn được sử dụng làm frame <math>r</math>. Lựa chọn này trên cơ sở quan sát thấy rằng nhà quay phim lựa chọn đoạn tiêu biểu với vài frame sau đó di chuyển theo (track) hay phóng/thu (zoom). Do vậy, frame đầu tiên của đoạn thường “nắm bắt” toàn bộ nội dung của đoạn.</p>	<p><b>Phương pháp 2:</b> Frame trung bình được xác định sao cho mỗi pixel trong frame này là trung bình của các giá trị pixel tại cùng điểm lưới trong toàn bộ frame của đoạn. Sau đó frame trong đoạn mà nó gần (tương tự) nhất với frame trung bình sẽ được chọn làm frame đại diện của đoạn.</p>	<p><b>Phương pháp 3:</b> Tính trung bình các biểu đồ màu của mọi frames trong đoạn. Frame nào mà biểu đồ màu của nó gần nhất biểu đồ giá trị trung bình sẽ được chọn làm frame đại diện.</p>	<p><b>Phương pháp 4:</b> Được sử dụng chủ yếu dành cho các đoạn có được di chuyển (panning) máy quay. Mỗi ảnh hay frame trong đoạn được phân thành nền và đối tượng tiền cảnh. Một nền lớn được hình thành từ nền của tất cả frame, và các đối tượng cận cảnh chính của mọi frame được đặt lên trên nền vừa tạo ra.</p>
--	---	--	---

#### b. Phương pháp chọn frame đại diện thứ 2:

- Khung hình đầu tiên của mỗi shot sẽ tự động được sử dụng làm khung hình  $r$ . Sau đó, mỗi khung hình tiếp theo trong ảnh được so sánh với khung hình  $r$  trước đó. Nếu khoảng cách lớn hơn ngưỡng  $T$  được tính trước, khung đó được đánh dấu là frame  $r$

mới. Vấn đề với cách tiếp cận này là số khung hình  $r$  là không thể đoán trước. Tìm ra 1 lượng frame  $r$  không lờ hoặc quá ít frame  $r$ .

=> Để giải quyết vấn đề này, sẽ tạo ra 1 giới hạn trên của số frame  $r$  và số frame  $r$  chọn sẽ tỷ lệ với lượng nội dung của mỗi shot. Lượng nội dung mỗi shot được đo dựa trên tổng số frame-to-frame differences trong mỗi shot.

### c. Phương pháp chọn frame đại diện thứ 3.

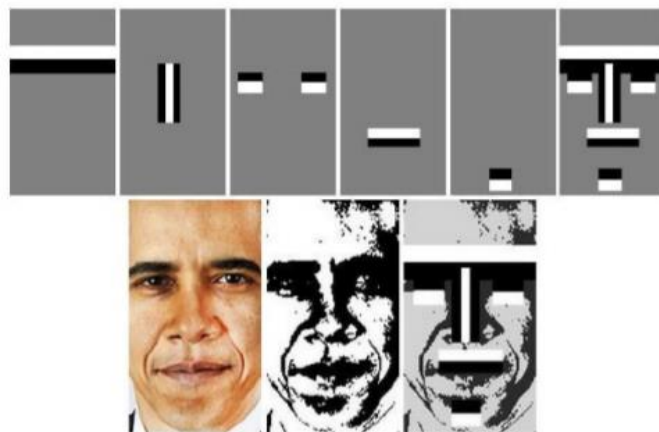
Tất cả các khung hình được tách ra thành hai phần đối tượng và nền. Khung hình có tỉ số giữa đối tượng và nền lớn nhất sẽ được chọn là khung hình chính của đoạn, bởi vì nó được cho rằng khung hình đó chứa thông tin nhiều nhất về đoạn đó. Một số cách tiếp cận khác cố gắng nhóm khung hình chính tương tự nhau (trong mỗi đoạn cơ sở hoặc toàn bộ video) vào thành các nhóm.

## 2. Kỹ thuật phát hiện mặt người trong ảnh sử dụng thuật toán Viola-Jones (Haar Cascade)

Vì nhóm có thêm một yêu cầu của bài toán là phát hiện video chưa mặt người từ ảnh cho trước nên cần phát hiện mặt người thì mới trích rút đặc trưng, nên phần này sẽ được trình bày thêm ở đây.

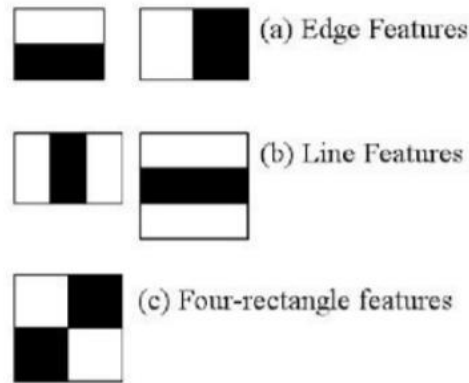
Ý tưởng của thuật toán là đi xây dựng một tập đầy đủ các đặc trưng thể hiện độ tương phản về mức xám giữa những vùng ảnh trong một cửa sổ. Khi đó, quá trình training cho mô hình sẽ chọn ra những đặc trưng có khả năng mô tả đối tượng tốt nhất (gọi là weak classifiers) và kết hợp chúng lại để tạo thành mô hình có khả năng nhận diện (strong classifier).

Mặt người cũng có những vùng có sự tương phản đặc trưng. Ví dụ như lông mày thường tối hơn trán và sống mũi thường sáng hơn hai vùng da bên sống mũi:



Các đặc trưng được sử dụng trong thuật toán là các đặc trưng kiểu Haar, có dạng các hình chữ nhật đen và trắng có kích thước bằng nhau.





Giá trị của một đặc trưng khi đặt vào bộ lọc harr kia sẽ là tổng giá trị mức xám của các điểm ảnh dưới các HCN đen trừ đi tổng giá trị mức xám dưới HCN trắng:

$$f(x) = \left| \sum_{i=1}^{w_1} \sum_{j=1}^{h_1} x(i,j) - \sum_{t=2}^{w_2} \sum_{k=1}^{h_2} x(t,k) \right|$$

Trong quá trình học, thuật toán sẽ chọn ra những đặc trưng tốt nhất để đưa vào mô hình. Việc huấn luyện và lựa chọn này được dựa trên thuật toán ADABOOST: Thuật toán này bắt đầu bằng việc khởi tạo trọng số của các bức ảnh training bằng nhau. Sau đó, thuật toán sẽ lặp T lần, mỗi lần chọn ra 1 feature gây ra ít lỗi nhất khi classify tập dữ liệu training để đưa vào strong classifier. Lỗi mà feature gây ra khi classify ảnh tỉ lệ thuận với trọng số của bức ảnh đó. Cuối cùng, trọng số của các bức ảnh sẽ được cập nhật lại sao cho những bức ảnh bị classify sai sẽ có trọng số lớn hơn và ngược lại. Strong classifier sẽ có dạng là tổ hợp tuyến tính của các weak classifier. Từ đây, để xác định xem trong ảnh có mặt người không, ta chỉ cần trượt cửa sổ qua ảnh và tính giá trị của hàm strong classifier với vùng ảnh dưới cửa sổ. Nếu giá trị đầu ra đủ lớn thì cửa sổ chứa mặt người. Mã giả như sau:

For  $t = 1, \dots, T$ :

1. Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that  $w_t$  is a probability distribution.

2. For each feature,  $j$ , train a classifier  $h_j$  which is restricted to using a single feature. The error is evaluated with respect to  $w_t$ ,  $e_j = \sum_i w_i |h_j(x_i) - y_i|$ .
3. Choose the classifier,  $h_t$ , with the lowest error  $e_t$ .
4. Update the weights:

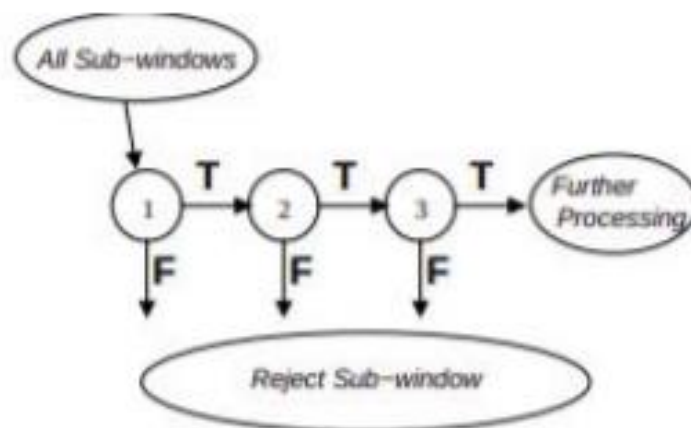
$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where  $e_i = 0$  if example  $x_i$  is classified correctly,  $e_i = 1$  otherwise, and  $\beta_t = \frac{e_t}{1-e_t}$ .

Hàm strong classifier có dạng:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Trong thực tế, hàm strong classifier có thể sẽ rất phức tạp về mặt tính toán. Vì vậy thay vì phải tính lại toàn bộ strong classifier cho từng cửa sổ, ta có thể tìm cách chia hàm classifier mạnh thành nhiều stage. Mỗi stage chỉ chứa vài weak classifier và các stage càng về sau thì chứa càng nhiều weak classifier. Nếu một cửa sổ bị đánh là không chứa mặt người ở stage nào đó thì những stage sau sẽ không cần phải tính toán nữa -> Giảm được đáng kể thời gian tính toán. Cấu trúc classifier này được gọi là cascade classifier. Minh họa:



### 3. Kỹ thuật trích rút đặc trưng trong ảnh:

Từ ảnh cho trước cần trích rút được đặc trưng của ảnh, rồi mới thể thể đem ra so sánh giữa các ảnh. Một số cách trích rút đặc trưng sẽ trình bày sau đây.

#### 3.1. Màu sắc

##### 3.1.1. Lấy đặc trưng dựa trên màu sắc cơ bản

Mỗi hình ảnh trong cơ sở dữ liệu được biểu diễn bằng cách sử dụng ba bầu chọn chính hoặc các kênh của không gian màu đã chọn. Thường, không gian màu sử dụng là (RGB). Mỗi kênh màu được sắp xếp tùy ý thành m khoảng. Vậy tổng số tổ hợp màu rời rạc là  $m^3$ .

Đặt  $n = m^3$

Sau đó tính toán biểu đồ màu  $H(M)$  là một vector  $(h_1, h_2, \dots, h_j, \dots, h_n)$ , trong đó phần tử  $h_j$  đại diện cho số pixel trong ảnh  $M$  rơi vào vị trí thứ  $j$  trong vector, hay gọi là  $bin_j$ . Biểu đồ này sẽ là đặc trưng của ảnh.

Về vấn đề cách so sánh: phép đo khoảng đơn giản nhất giữa hình ảnh  $I$  và  $H$  được xác định là:

$$d(I, H) = \sum_{l=1}^n |i_l - h_l|$$

Trong số :  $i_l$  là số pixel rơi vào bin  $l$  của ảnh  $I$ ,  $h_l$  là số pixel rơi vào bin  $l$  của ảnh  $H$

Ngoài ra có thể sử dụng khoảng cách Euclide.

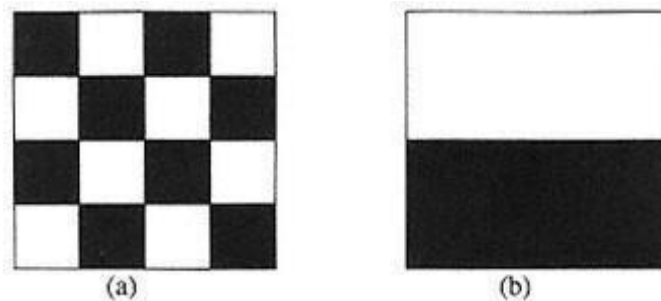
Cuối cùng hình ảnh có khoảng cách biểu đồ nhỏ hơn ngưỡng xác định trước được truy xuất từ cơ sở dữ liệu và hiển thị cho người dùng. Ngoài ra, hình ảnh đầu tiên có khoảng cách nhỏ nhất cũng có thể được lấy ra.

### 3.1.2. Biểu đồ màu cục bộ (Local Color Histogram) để tận dụng sự tương đồng giữa các màu

Hạn chế đầu tiên của kỹ thuật trên là sự giống nhau giữa các màu (và thùng) khác nhau bị bỏ qua và nó bỏ qua các mối quan hệ không gian giữa các pixel.

Ví dụ 1: có hai bin đại diện cho dải màu 1 - 10 và 11 - 20 thì với màu 10 rơi vào bin 1, màu 11 và màu 20 rơi vào bin 2. Như vậy có nghĩa là chúng ta đang nói màu 11 giống với màu 20 và khác với màu 10. Điều này là không chính xác.

Ví dụ 2: hai hình ảnh dưới đây có cùng một histogram màu nếu được tính theo kỹ thuật ở phần a trên, dẫn đến kết luận sai rằng hai hình ảnh này giống nhau.



Một ví dụ hiển thị hai hình ảnh khác nhau có cùng một biểu đồ cơ bản.

Khắc phục hạn chế này, đề xuất rằng mỗi hình ảnh được phân đoạn thành một số vùng cố định và biểu đồ được tính toán cho từng vùng.

Ví dụ nếu mỗi kênh màu được chia thành 16 khoảng, chúng ta có tổng cộng 4.096 tổ hợp. Ảnh được phân đoạn thành  $k$  vùng. Chiều dài của vector đặc trưng histogram là  $16 \times 16 \times 16 \times k$ .

Trong quá trình truy xuất, biểu đồ của các vùng tương ứng được so sánh.

$$D(Q, I) = \sum_{k=1}^M \sqrt{\sum_{i=1}^N (H_0^k[i] - H_1^k[i])^2}$$

+ Ở đây M là số vùng được phân đoạn trong ảnh, N là số màu trong biểu đồ màu và  $H[i]$  là giá trị của màu i trong biểu đồ màu đại diện cho vùng k của ảnh.

### 3.1.3. Độ đo tương đồng về màu sắc

Gọi  $h(I)$  và  $h(M)$  tương ứng là 2 lượt đồ màu của hai ảnh I và ảnh M. Khi đó các loại độ đo màu được định nghĩa là 1 trong các cách sau:

- Khoảng cách Euclidean: khoảng cách Euclidean thông thường giữa các K bin:

$$Intersection(h(I), h(M)) = \sum_{j=1}^K \sqrt{(h(I) - h(M))^2}$$

- Hoặc:

$$Intersection(h(I), h(M)) = \sum_{j=1}^K |h(I) - h(M)|$$

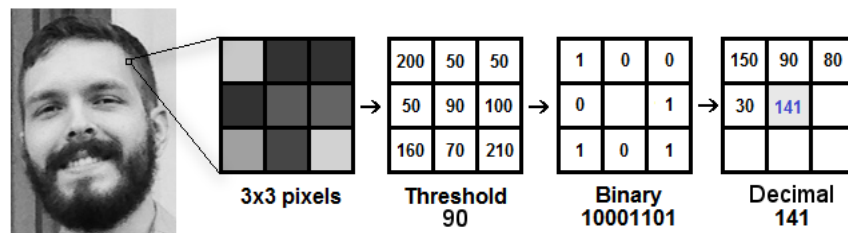
## 3.2. Tra cứu ảnh dựa trên kết cấu

### 3.2.1. Trích xuất kết cấu dựa vào thuật toán Local Binary Patterns

Có rất nhiều loại mô tả kết cấu khác nhau được sử dụng để trích xuất các đặc điểm của hình ảnh. Local Binary Pattern, còn được gọi là LBP, là một thước đo mô tả kết cấu bất biến đơn giản và thang xám để phân loại. Trong LBP, mã nhị phân được tạo ở mỗi pixel bằng cách đặt các pixel lân cận của nó thành 0 hoặc 1 dựa trên giá trị của pixel trung tâm.

**Thuật toán như sau:**

- Ban đầu, tại mỗi điểm của hình ảnh, xét 8 điểm xung quanh điểm đang xét, lấy giá trị của điểm đang xét là ngưỡng giá trị, các điểm xung quanh có giá trị lớn hơn hoặc bằng ngưỡng sẽ được đánh dấu là 1 và nếu nhỏ hơn ngưỡng sẽ được đánh dấu là 0
- Sau khi lập ngưỡng, thu thập tất cả các giá trị ngưỡng từ vùng lân cận theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ. từ đó thu thập được dãy nhị phân gồm 8 chữ số. Chuyển mã nhị phân thành thập phân. Thay thế giá trị pixel trung tâm bằng số thập phân kết quả và thực hiện quy trình tương tự cho tất cả các giá trị pixel có trong hình ảnh.

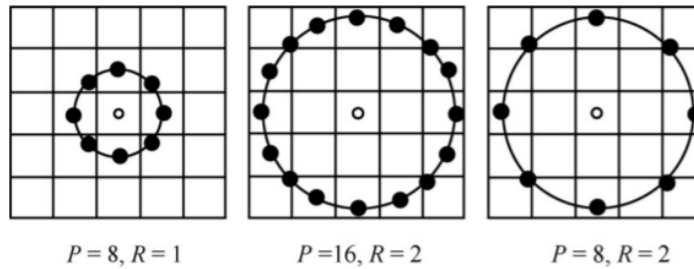


- VD:

**Cải tiến:** Phương pháp này sẽ khiến các đặc trưng của LBP không quá ổn định và biểu diễn được các đặc trưng quá lớn. Và vì vậy Ojala đưa ra một phương pháp cải tiến: các điểm lân cận không được lấy mẫu theo 8 điểm xung quanh mà sẽ là một tập hợp các điểm thuộc đường tròn với điểm đang xét là tâm. Gọi (P, R) là một vùng lân cận gồm P điểm trên một đường tròn có bán kính R. T là đặc điểm kết cấu của vùng lân cận, T được kí hiệu là:

$$T = t(g_c, g_0, \dots, g_{p-1})$$

- Trong đó, là giá trị trên ảnh xám của điểm trung tâm và các điểm trên đường tròn bán kính R. Ví dụ  $g_c$  và  $(g_0, g_1, \dots, g_{P-1})$  dụ sau:

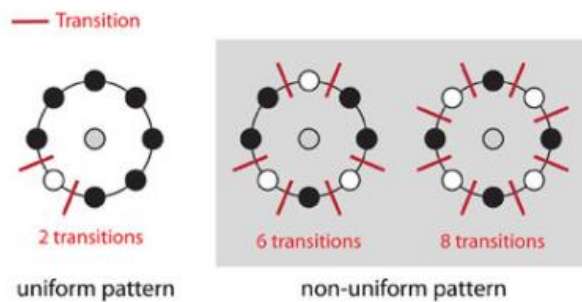


Từ đó kết cấu cục bộ xung quanh điểm đang xét:

$$LBP_{P,R} = \sum_{i=0}^{P-1} s(g_i - g_c) 2^i$$

Công thức cải tiến này đã thêm nhiều bán kính vào để tính tổng, ngoài R=1 như ở phần đầu.

Tuy nhiên nếu số điểm lân cận xét nhiều, từng đặc trưng kết cấu cục bộ của từng điểm tăng, cản trở cho việc phân loại và khai phá, thì Ojala đã thêm một khái niệm một dãy nhị phân nếu có ít hơn hoặc bằng hai thay đổi thì gọi là “Uniform patterns”. Ví dụ minh họa:



Do đó, giá trị của biểu diễn mẫu sẽ được giảm đi đáng kể. Ví dụ: với 8 điểm lân cận, giá trị biểu diễn mẫu được giảm xuống từ 256 xuống 58, công thức tính Uniform patterns:

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c), & \text{if } U(LBP_{P,R}) \leq 2 \\ P + 1, & \text{otherwise} \end{cases}$$

Ngoài ra, để đạt được bất biến khi ảnh bị xoay, một hàm bất biến xoay của LBP được định nghĩa như sau:

$$LBP_{P,R}^{ri} = \min(ROR(LBP_{P,R}^{ri}, i) \mid i = 0, 1, \dots, P - 1)$$

+ Trong đó, ROR là hàm quay. Hàm này sẽ thay đổi chuỗi nhị phân thu được từ các điểm mẫu lần lượt. Sau khi tính toán hết giá trị LBP cho mỗi chuỗi, giá trị nhỏ nhất sẽ được chọn để biểu diễn mẫu kết cấu đó. Hình dưới mô tả một hình ảnh qua xử lý bằng LBP.

Trong hầu hết các ứng dụng, histogram LBP được khai thác như các đặc điểm kết cấu tạo ra không gian đặc trưng về chiều, tác dụng của việc này để giảm kích thước của vector đặc trưng trích rút ra mà vẫn có hiệu quả. Ví dụ như từ ảnh 300x300 sau khi qua phép LBP trên sẽ cho kích thước vector đặc trưng là 90000, nếu dùng histogram thì kích thước vector đặc trưng là 256 nếu như áp dụng LBP chưa có cải tiến.

Tóm lại là ảnh đi qua các phép LBP (sẽ tính LBP cho từng điểm) được biểu diễn thành các vector histogram.

### 3.2.2. Độ đo tương đồng cho kết cấu ảnh

Để đo độ tương đồng theo kết cấu giữa các ảnh, người ta thường sử dụng độ đo Euclidean giữa các đặc trưng của ảnh truy vấn với đặc trưng của ảnh trong cơ sở dữ liệu.

## 3.3. Truy xuất hình ảnh dựa trên hình dạng

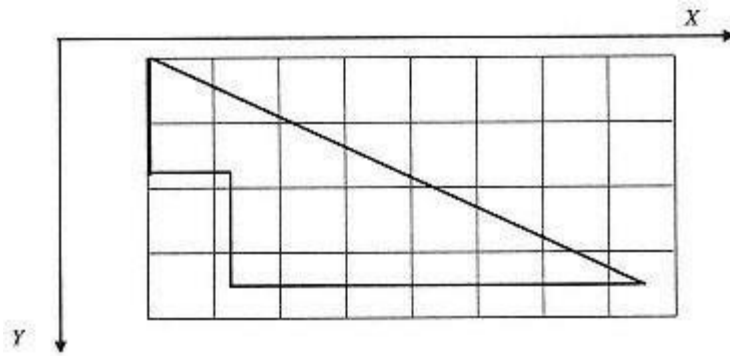
- Trục chính: đoạn thẳng nối hai điểm trên biên cách xa nhau nhất.
- Trục nhỏ: đường thẳng vuông góc với trục chính và có độ dài sao cho một hình chữ nhật có các cạnh song song với các trục chính và trục phụ chỉ bao quanh giới hạn-có thể được hình thành bằng cách sử dụng độ dài của trục chính và trục phụ.
- Hình chữ nhật cơ bản: hình chữ nhật trên được tạo thành với trục chính và trục phụ là hai cạnh của nó được gọi là hình chữ nhật cơ bản.
- Độ lệch tâm: tỷ số giữa trục chính và trục phụ được gọi là độ lệch tâm của đường biên.

### 3.3.1. Đo lường sự tương đồng và biểu diễn hình dạng dựa trên khu vực

Cho một hình dạng, chúng ta phủ một không gian lưới lên trên nó gồm các ô vuông có kích thước cố định, vừa đủ lớn để bao phủ hoàn toàn hình dạng.

Ta chỉ định 1 cho ô có ít nhất 15% pixel được bao phủ bởi hình dạng và 0 cho mỗi ô khác. Sau đó, từ trái sang phải và từ trên xuống dưới ta thu được một chuỗi nhị phân cho hình dạng.

Ví dụ, hình dạng dưới đây được biểu diễn bằng chuỗi nhị phân 11100000  
11111000 01111110 01111111.

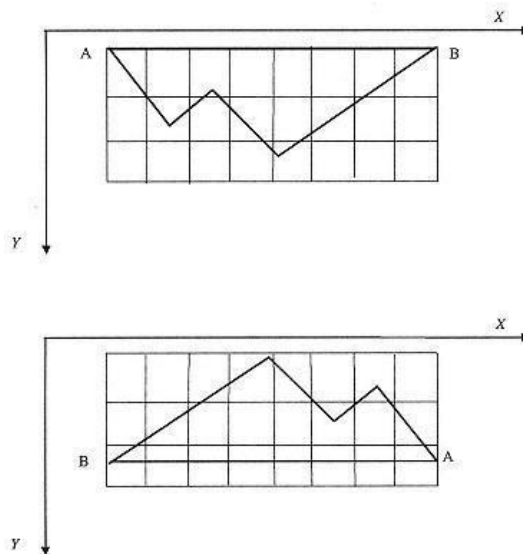


Thấy rằng kích thước ô càng nhỏ thì việc biểu diễn hình dạng càng chính xác vì vậy việc tính toán và lưu trữ sẽ cần nhiều hơn. Kích thước ô phù hợp khoảng 10x10 đến 20x20 pixel.

Tuy nhiên, khi cách tiếp cận này không bất biến đối với tỷ lệ và phép quay. Do đó, chuỗi nhị phân được chuẩn hóa cho tỷ lệ và phép quay.

### ***Chuẩn hóa xoay***

Mục đích của chuẩn hóa xoay là đặt các hình dạng theo một hướng chung duy nhất. Khi xoay hình để trục chính của nó song song với trục x, sẽ có hai khả năng cho vị trí hình dạng: một trong những điểm xa nhất có thể ở bên trái hoặc bên phải. Điều này là do quay  $180^\circ$ . Minh họa:



Hai hướng có thể có với trục chính dọc theo hướng x.

Vì vậy ta dùng hai chuỗi nhị phân khác để biểu diễn hai cách xoay này. Để tiết kiệm dung lượng lưu trữ, ta chỉ lấy và lưu trữ một trong các chuỗi nhị phân kia. Hai hướng được tính toán trong thời gian truy xuất bằng cách biểu diễn hình dạng truy vấn bằng cách sử dụng hai chuỗi nhị phân được so sánh với mỗi chỉ mục hình dạng được lưu trữ trong cơ sở dữ liệu.

### ***Chuẩn hóa quy mô***

Chia tỷ lệ tất cả các hình dạng theo tỷ lệ để các trục chính của chúng có cùng độ dài cố định. Một số tác giả giới thiệu độ dài cố định là 192 pixel.

### ***Các hoạt động hình dạng khác***

Ngoài việc xoay  $180^\circ$  của các hình dạng, hai thao tác khác dẫn đến các hình dạng tương tự về mặt tri giác là lật ngang và lật dọc.

Ta sẽ tạo bốn chuỗi nhị phân cho mỗi hình dạng truy vấn trong quá trình truy xuất và lưu thì vẫn lưu một chuỗi.

### ***Xử lý nhiều trục chính***

Khi đó một ảnh sẽ phải có nhiều cặp số nhị phân bằng với số trục chính. Khoảng cách giữa hai hình là khoảng cách nhỏ nhất giữa mỗi cặp số nhị phân của hai hình này.

#### ***3.3.2. Đo lường độ tương đồng***

Có 3 cách thường để tính độ tương đồng:

- Cách 1: Nếu hai hình dạng đã được chuẩn hóa (xoay, tỉ lệ) so sánh theo bit của hai hình dạng này và khoảng cách giữa chúng bằng số vị trí có các giá trị khác nhau.

+ Ví dụ: nếu hình dạng A và B có dãy nhị phân lần lượt là 11111111  
11100000 và 11111111 11111100, thì khoảng cách giữa A và B là 3.

- Cách 2: Nếu hai hình dạng đã được chuẩn hóa có độ dài trục nhỏ rất khác nhau thì không cần tính độ giống nhau của chúng vì có thể giả định là hai hình này rất khác nhau.

+ Ví dụ: hình A và B lần lượt có độ dài của trục nhỏ là 1 và 4 ô thì có thể giả định rằng hai hình này hoàn toàn khác nhau và không có giá trị nào trong việc truy xuất hình dạng. Ngưỡng chênh lệch giữa các trục nhỏ phụ thuộc vào các ứng dụng và kích thước ô. Thông thường, nếu độ dài của các trục nhỏ của hai hình khác nhau hơn 3 ô, thì hai hình này được coi là khá khác nhau.

- Cách 3: Nếu hai hình dạng đã được chuẩn hóa có trục nhỏ hơi khác nhau, thì thêm các số 0 vào cuối chỉ mục của hình có trục nhỏ ngắn hơn, để chỉ mục mở rộng có cùng độ dài với chỉ số của hình còn lại. Khoảng cách giữa hai hình này được tính như cách 1.

#### ***3.4. Tra cứu ảnh dựa trên đặc trưng bất biến***

- Phương pháp tra cứu này có tên là Scale-Invariant Feature Transform (SIFT) và đặc trưng trích rút được gọi là đặc trưng SIFT (David Lowe đề xuất)
- Phương pháp này trích rút các **đặc trưng cục bộ bất biến của ảnh**.
- **Đặc trưng cục bộ bất biến của ảnh** là các đặc trưng mà bất biến với việc thay đổi tỉ lệ ảnh, quay ảnh, thay đổi cường độ chiếu sáng của ảnh, tức là phụ thuộc rất ít vào các phép biến đổi cơ bản như xoay, phóng to, thu nhỏ, tăng giảm cường độ sáng, vì vậy được gọi là các đặc trưng mang tính cục bộ.
- **Ưu:** Với ảnh tự nhiên việc đối sánh gặp nhiều khó khăn do các thay đổi của ảnh như độ chiếu sáng, co dãn, chồng lấp, thay đổi góc nhìn... gây ảnh



hưởng nhiều đến độ chính xác, phương pháp này có thể khắc phục một số khó khăn đó.

- **Nhược:** Tốc độ thuật toán sẽ bị ảnh hưởng vì khi đối sánh này cần chi phí đối sánh rất lớn đối với cơ sở dữ liệu ảnh có số lượng lớn và số lượng các đặc trưng ở mỗi ảnh lớn.

#### 3.4.1. Các trích chọn đặc trưng SIFT

- Đặc trưng cục bộ bất biến của ảnh(SIFT) là các điểm đặc trưng (nhiều điểm), gọi là các keypoint. Mỗi keypoint này sẽ được mô tả dưới dạng các vector(nhiều chiều) và keypoint mô tả đó là gốc của vector đó.

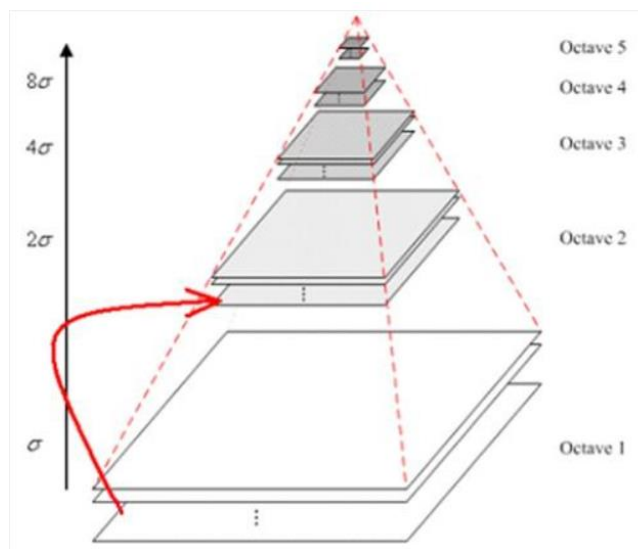
- Cách thực hiện trích đặc trưng như sau:

- + **Phát hiện các điểm cực trị (Scale-Space extrema detection):** sử dụng đạo hàm của hàm Gaussian (DoG - Difference of Gaussians) để tìm ra các điểm có khả năng làm điểm đặc trưng tiềm năng, bất biến về tỉ lệ và
- + **Định vị các điểm đặc trưng (keypoint localization):** Từ những điểm tìm ra ở bước trước sẽ lọc và lấy ra tập các điểm đặc trưng tốt nhất(keypoints).
- + **Xác định hướng cho các điểm đặc trưng (Orientation assignment):** Mỗi keypoint sẽ được gán cho một hoặc nhiều hướng dựa trên hướng gradient của ảnh.
- + **Mô tả các điểm đặc trưng (Keypoint descriptor):** Các điểm hấp dẫn sau khi được xác định hướng sẽ được mô tả dưới dạng các vector đặc trưng nhiều chiều.

Chi tiết các bước là các phần sau:

##### a) Phát hiện các điểm cực trị

- Bước đầu tiên sẽ tìm các điểm tiềm năng có thể trở thành điểm đặc trưng. Như đã nêu ở trên là các điểm lấy ra ở bước này sẽ ít phụ thuộc vào sự thu phóng ảnh và xoay ảnh.
- Bằng phương pháp **lọc kim tự tháp** dựa vào việc thay đổi tham số bộ lọc Gaussian ở các tỉ lệ khác nhau, minh họa:



- Cụ thể cách tính như sau:

+ Hàm  $L(x,y,\sigma)$  (Hàm không gian tỷ lệ của ảnh  $I$ ) được tạo ra bằng cách nhân chập ảnh gốc  $I(x,y)$  với một hàm Gaussian  $G(x,y,\sigma)$  có tham số về độ đo  $\sigma$  thay đổi.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad [2.1]$$

Với: +  $G(x, y, \sigma)$  : biến tỷ lệ Gaussian (variable scale Gaussian) :

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

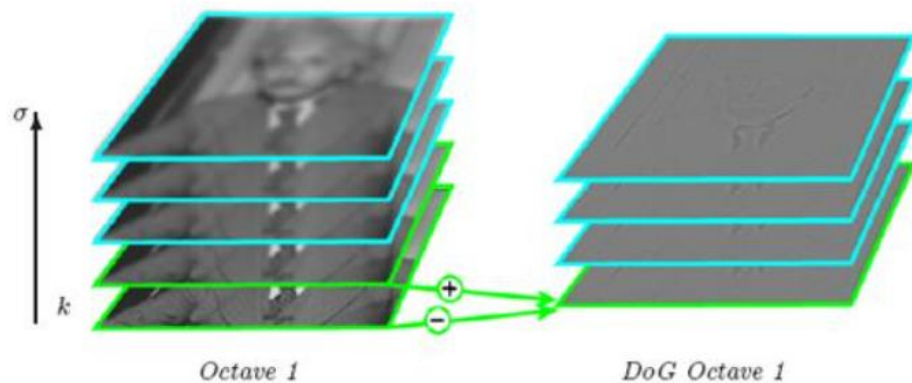
+  $I(x, y)$  : Ảnh đầu vào

+  $*$  là phép nhân chập giữa  $x$  và  $y$

Hàm tìm cực trị cục bộ của đạo hàm của hàm Gaussian viết tắt là DoG (Difference-of-Gaussian), tính toán từ sự sai khác giữa 2 độ đo không gian cạnh nhau của một ảnh với tham số đo lệch nhau một hằng số  $k$ :

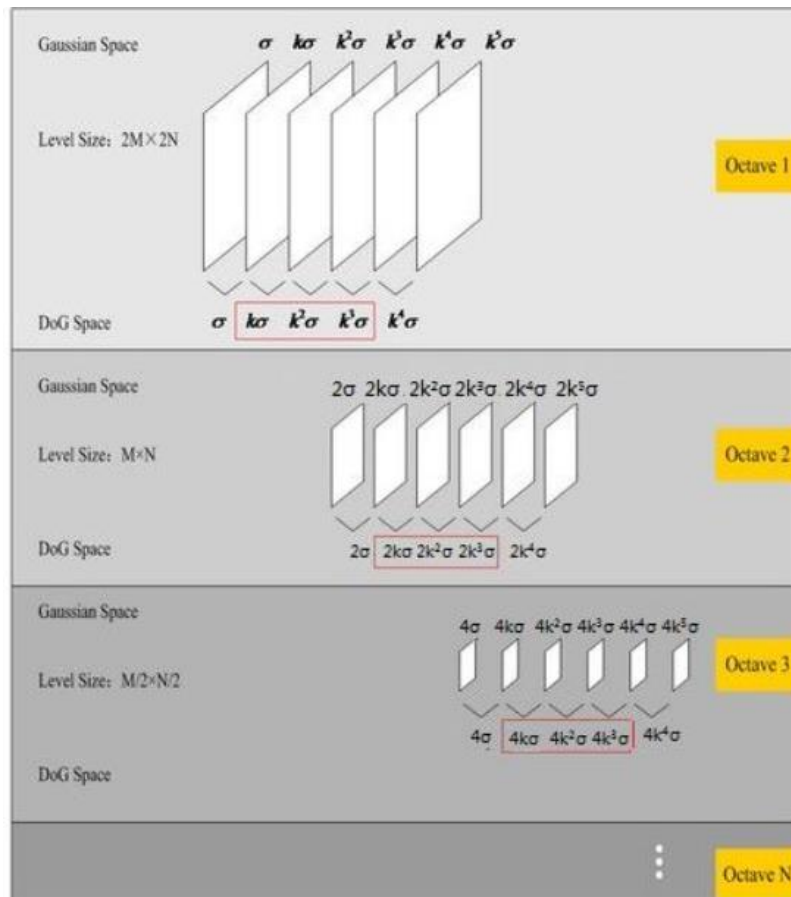
$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

Hai hình ảnh liên tiếp trong một nhóm octave được ghép với nhau để thực hiện, kết quả là một xấp xỉ bất biến tỷ lệ của Laplacian of Gaussian

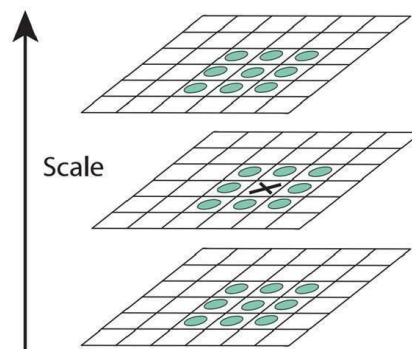


Biểu đồ mô phỏng việc tính toán các ảnh DoG từ các ảnh mờ

Các ảnh cuộn được nhóm thành nhóm octave (mỗi octave tương ứng với giá trị gấp đôi của  $\sigma$ ). Giá trị của  $k$  được chọn với số lượng ảnh mờ cho mỗi nhóm octave là cố định và đảm bảo cho số lượng các ảnh DoG cho mỗi nhóm octave là như nhau:



Tiếp theo, mỗi điểm ảnh trong DoG được so sánh với 8 điểm ảnh láng giềng của nó ở cùng tỉ lệ đó và 9 láng giềng kề ở các tỉ lệ ngay trước và sau nó. Nếu điểm ảnh đó đạt giá trị cực tiểu hoặc cực đại thì sẽ được chọn làm các điểm hấp dẫn ứng viên.



Hình 2.3 : Quá trình tìm điểm cực trị trong các hàm sai khác DoG

(X là điểm hiện tại, các vòng tròn màu xanh là các láng giềng của nó)

Thông thường, một vị trí không cực đại hoặc không cực tiểu sẽ không phải đi qua tất cả 26 kiểm tra. Một vài kiểm tra ban đầu thường là đủ để loại bỏ.

## b) Định vị điểm hấp dẫn :

- Bước này xem xét mỗi điểm hấp dẫn bước trước khi được chọn sẽ được đánh giá xem có bị xóa bỏ hay không, thỏa mãn một trong hai tiêu chí:
- Các điểm có độ tương phản thấp
- Một số điểm dọc theo các cạnh không giữ được tính ổn định khi ảnh bị nhiễu cũng bị loại bỏ

**Bước thực hiện gồm 3 công đoạn :**

**Bước 1: Phép nội suy lân cận cho vị trí đúng của điểm tiềm năng:**

- Ở bước trước, các điểm tìm được là các keypoint tiềm năng, ở bước này sẽ xác định vị trí subpixel một cách có toán học.
- Phép nội suy lân cận sử dụng mở rộng Taylor cho hàm Difference-of-Gaussian  $D(x,y,\sigma)$ :

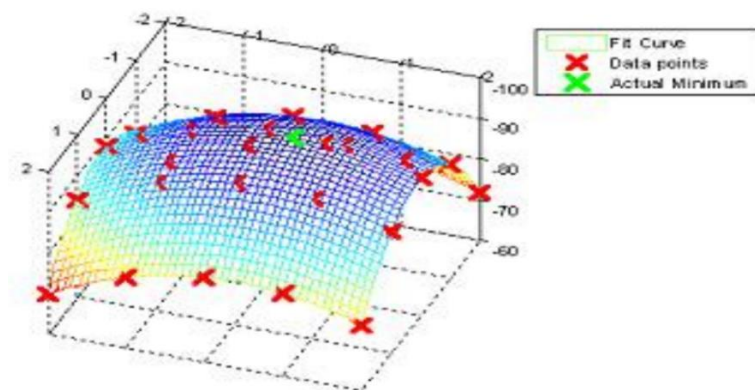
$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x} \quad (1)$$

$\mathbf{X} = (x, y, \sigma)^T$  : là độ dịch so với các điểm lân cận của điểm lấy mẫu.

- Có thể dễ dàng tìm được các điểm cực trị của phương trình này, bằng cách lấy đạo hàm hàm này đối với  $\mathbf{x}$  và cho bằng 0, và gọi vị trí điểm cực trị là  $\hat{\mathbf{x}}$ .

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}. \quad (2)$$

Nếu  $\hat{X} > 0.5$  theo một chiều nào đó thì nó có chỉ số cực trị không gần với các điểm tiềm năng khác, nó sẽ bị thay đổi và phép nội suy sẽ thay thế vai trò của nó bằng điểm khác gần nó. Thực hiện tiếp tục với các điểm lấy mẫu khác.



**Mô phỏng sử dụng công thức mở rộng của Taylor cho hàm DoG**

## Bước 2: Loại trừ các điểm có tính tương phản kém :

- Những điểm có thỏa mãn ( $< 0.5$ ) được thêm vào tập hợp mẫu tốt nhất, tiếp tục phân tích tiếp.
- Dùng  $D(\hat{X})$  để loại những điểm cực trị không ổn định (độ tương phản thấp).
- Thay (2) vào (1) ta được:

$$D(\hat{X}) = D + \frac{1}{2} \frac{\partial D^T}{\partial X} \hat{X}$$

Nếu  $|D(\hat{x})|$  nhỏ hơn 0.03 sẽ bị loại bỏ (giả sử là pixel hình ảnh đã trong phạm vi  $[0;1]$  , 0.03 là do tác giả thực nghiệm thí nghiệm có được kết quả như vậy)

## Bước 3: Loại bỏ các điểm dư thừa theo biên :

- Sử dụng hàm DoG sẽ cho tác động mạnh đến biên khi vị trí của biên, khó xác định nó và vì vậy các điểm tiềm năng trên biên có thể sẽ không bắt biên và bị nhiễu, và bước này, ta sẽ loại trừ các điểm tiềm năng khó định vị (tức là vị trí dễ thay đổi khi có nhiễu do nằm ở biên).
- Sau khi áp dụng hàm DoG sẽ làm đường biên ảnh không rõ ràng và độ cong chính sẽ có giá trị lớn hơn nhiều so với độ cong dọc theo biên vì vậy cần loại bỏ bớt các điểm đặc biệt dọc theo cùng một biên. Giải pháp cho việc này là sử dụng giá trị của ma trận Hessian cấp 2 tại mỗi điểm tiềm năng như sau:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} [2.9]$$

- Tính toán giá trị alpha, beta:

$$\begin{aligned} \text{Tr}(\mathbf{H}) &= D_{xx} + D_{yy} = \alpha + \beta, \\ \text{Det}(\mathbf{H}) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \end{aligned}$$

- Tính  $r = \alpha/\beta$
- Các điểm bị loại bỏ là  $r > 10$ , theo bài báo của tác giả.
- Sau bước này, chúng ta có những điểm hấp dẫn có độ ổn định.

## c) Xác định hướng cho các điểm hấp dẫn

- Bằng cách chỉ định một hướng phù hợp cho mỗi điểm hấp dẫn dựa trên các thuộc tính hình ảnh cục bộ, dựa vào hướng của điểm hấp dẫn này ta có thể biết được điểm hấp dẫn bất biến với phép quay ảnh.
- **Với tác giả sau khi thử nghiệm**, với một số phương pháp tiếp cận để xác định hướng, phương pháp đã được tìm thấy cho kết quả ổn định nhất là:
  - + Tại mỗi điểm hấp dẫn người ta tính toán biểu đồ hướng Gradient trong vùng láng giềng của điểm hấp dẫn. Độ lớn và hướng của các điểm hấp dẫn được xác định theo công thức:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

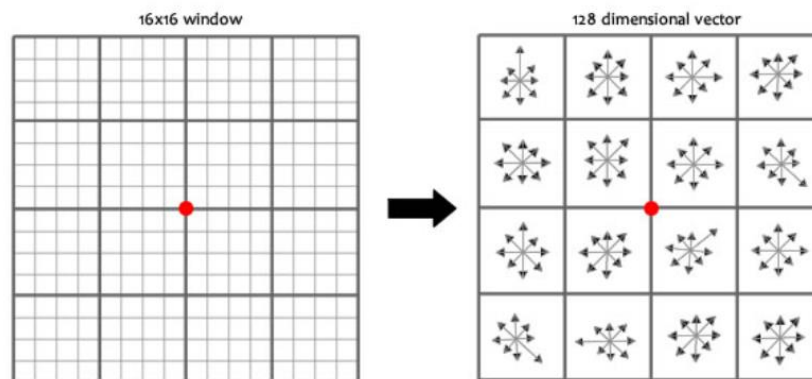
Trong đó :

$m(x, y)$  : Độ lớn của vector định hướng

$\theta(x, y)$  : Hướng của vector định hướng (biểu diễn qua góc  $\theta$ )

#### d) Mô tả các điểm hấp dẫn

- Bước này: sẽ tính toán một bộ mô tả cho một vùng ảnh cục bộ mà có tính đặc trưng cao (bất biến với các thay đổi khác nhau về độ sáng, thu - phóng ảnh, xoay).
- Một cửa sổ 16x16 xung quanh keypoint. Cửa sổ 16x16 này được chia thành mười sáu cửa sổ 4x4

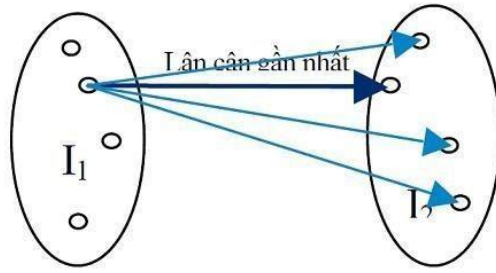


- Trong mỗi cửa sổ 4x4, độ lớn và hướng của gradient được tính toán. Những định hướng này được đưa vào một histogram 8 bin .
- Điểm hấp dẫn sau khi được xác định hướng sẽ được biểu diễn dưới dạng các vector 4x4x8=128 chiều, các vector này có đặc điểm : chung gốc và độ dài mỗi vector tương ứng độ lớn gradient m của nó.

#### 3.4.2. Đối sánh đặc trưng SIFT

Để đối sánh và **nhận dạng** hai ảnh thì ta tìm tập keypoint giống nhau trong hai ảnh, dựa vào hướng và tỉ lệ để có thể biết đối tượng trong ảnh gốc đã xoay, thu phóng bao nhiêu so với ảnh đem đối sánh. thuật toán này dựa vào điểm bất biến cục bộ của ảnh, chúng được trích xuất ra, được định hướng và mô tả sao cho hai keypoint ở hai vùng khác nhau thì khác nhau.

Việc đối sánh hai tập hợp điểm đặc trưng quy về bài toán tìm láng giềng gần nhất của mỗi điểm đặc trưng ví dụ sau:



Một phương pháp được đề xuất bởi D. Mount cho phép tìm kiếm nhanh các điểm lân cận được sử dụng, ANN là viết tắt của Approximative Neighbour. Nó cho phép tổ chức dữ liệu dưới dạng kd-tree, việc tìm kiếm láng giềng gần nhất mang tính xấp xỉ trên kd-tree. Cụ thể là hai điểm trong không gian đặc trưng được coi là giống nhau nếu khoảng cách Euclidean giữa hai điểm là nhỏ nhất và tỉ số giữa khoảng cách gần nhất với khoảng cách gần nhì phải nhỏ hơn 1 ngưỡng cho trước.

Giả sử cặp keypoint có bộ mô tả lần lượt là:

$$A = (a_1, a_2, a_3, \dots, a_{128}) \text{ và } B = (b_1, b_2, b_3, \dots, b_{128})$$

Thì khoảng cách Euclidean giữa A và B được tính bằng công thức:

$$D(A, B) = \sum (a_i - b_i)^2 \quad [2.12]$$

- **Một số độ đo tương đồng cho ảnh sử dụng đặc trưng trong SIFT**

- Độ đo Cosin :

$$d(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|} \quad [2.13]$$

- - Khoảng cách góc :

$$d(x, y) = \cos^{-1}(x \cdot y) \quad [2.14]$$

- - Độ đo Euclidean :

$$d(x, y) = \sqrt{\sum_{i=1}^n |x_i - y_i|^2} \quad [2.15]$$

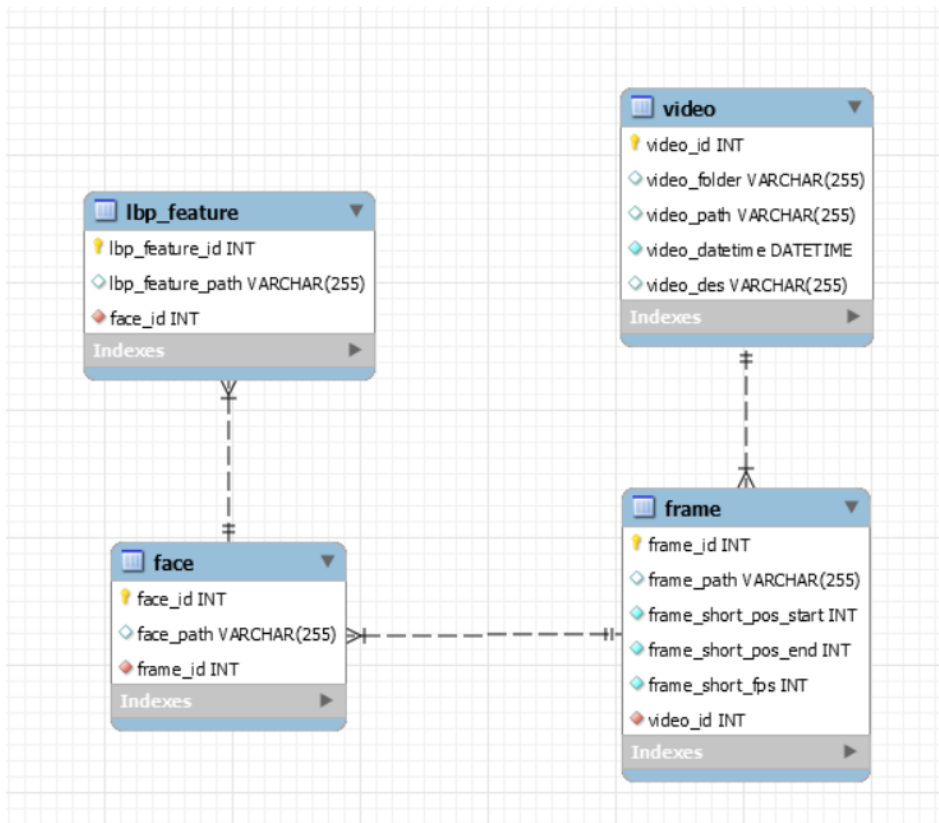
- - Độ đo Jensen-Shannon divergence :

$$d_{JSD}(H, H') = \sum_{m=1}^M H_m \log \frac{2H_m}{H_m + H'_m} + H'_m \log \frac{2H'_m}{H'_m + H_m} \quad [2.16]$$

Với H, H' là 2 biểu đồ biểu diễn các vector đặc trưng SIFT

### III. Hệ thống chương trình:

#### 1. Đối tượng và cơ sở dữ liệu:



*Thiết kế cơ sở dữ liệu*

Hệ thống gồm 4 bảng là video, frame, face, lbp\_feature:

- Bảng video:
  - video\_id: khóa chính
  - video\_folder: tên folder của video
  - video\_path: đường dẫn vật lý đến video được lưu trong máy
  - video\_datetime: ngày và giờ upload vào database
  - video\_des: mô tả video ( có mặt người hay không: có mặt người giá trị bằng 1, không có mặt người giá trị bằng 0)
- Bảng frame:
  - frame\_id: khóa chính.
  - frame\_path: đường dẫn vật lý.tới file frame .
  - frame\_short\_pos\_start: thứ tự của frame bắt đầu của shot .
  - frame\_short\_pos\_end: thứ tự của frame kết thúc của shot.
  - frame\_short\_fps: số frame trên giây của từng video.
  - video\_id: khóa ngoại chỉ tới bảng video, mô tả frame đó được tách ra từ video nào.
- Bảng face:
  - face\_id: khóa chính.



face\_path: đường dẫn vật lý đến file image trong máy.

frame\_id: khóa ngoại chỉ tới bảng frame, mô tả face đó được tách ra từ frame nào.

- Bảng lbp\_feature:

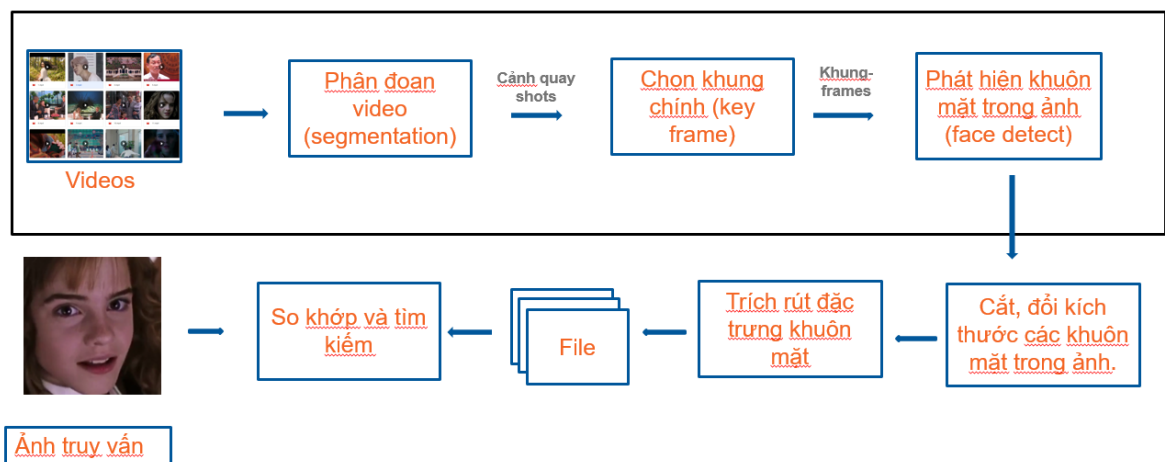
lbp\_feature\_id: khóa chính.

lbp\_feature\_path: đường dẫn vật lý đến file txt trong máy.

face\_id: khóa ngoại chỉ tới bảng face, biết được file đặc trưng này của image nào.

## 2. Hoạt động của chương trình:

Sơ đồ hoạt động:



B1: Trích đặc trưng cho video và lưu vào cơ sở dữ liệu

- Upload và lưu trữ dữ liệu
- Tách shot chọn frame đại diện cho các video:
- Phát hiện mặt người sử dụng Haar Cascade
- Sử dụng LBPH để rút trích đặc trưng ảnh

B2: So sánh ảnh tìm kiếm với đặc trưng của video

- Trích rút đặc trưng ảnh đầu vào
- Tính toán độ tương đồng giữa đặc trưng ảnh vào và đặc trưng video
- Phát hiện và cắt các mặt người xuất hiện trong ảnh (đã resize về cùng kích thước)
- Rút trích đặc trưng của các ảnh mặt người và lưu vào cơ sở dữ liệu.

B2: So sánh ảnh tìm kiếm với đặc trưng của video

- Trích rút đặc trưng ảnh đầu vào
- Tính toán độ tương đồng giữa đặc trưng ảnh vào và đặc trưng video

B3: Hiển thị kết quả: chọn ra kết quả có độ tương đồng thỏa mãn ngưỡng cho trước

### 2.1.Upload và lưu trữ dữ liệu:

- Các video được chọn để upload lên hệ thống được đưa vào path\_src:
- Lưu video từ folder vào database

```
def insert_video(db: Session, path_src, datetime, des):
    video = models.Video(
        video_datetime=datetime,
        video_des=des
    )
    db.add(video)
    db.flush()

    video_folder = os.path.join(config.DATA_PATH, str(video.video_id).zfill(5))
    if not os.path.exists(video_folder):
        os.mkdir(video_folder)
    else:
        rmtree(video_folder)
        os.mkdir(video_folder)
    video_path = os.path.join(video_folder, "{}.mp4".format(str(video.video_id).zfill(5)))
    copyfile(path_src, video_path)
    video.video_folder = os.path.relpath(video_folder, config.DATA_PATH)
    video.video_path = os.path.relpath(video_path, config.DATA_PATH)
    db.flush()
    db.refresh(video)
    return video
```

## 2.2. Tách shot chọn frame đại diện:

Sử dụng phương pháp ..... trong phần II.

- Đầu vào là các video, đọc từng frame trong video. Sau đó tính độ khác biệt giữa 2 frame liên tiếp dựa vào khoảng cách 2 histogram của 2 frame.

```
def histogram_color(self, frame1, frame2):

    # Histogram tren frame 1
    hsv1 = cv.cvtColor(frame1, cv.COLOR_BGR2HSV)
    hist1 = cv.calcHist([hsv1], [0, 1], None, [180, 256], [0, 180, 0, 256])
    cv.normalize(hist1, hist1, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)

    #Histogram tren frame 2
    hsv2 = cv.cvtColor(frame2, cv.COLOR_BGR2HSV)
    hist2 = cv.calcHist([hsv2], [0, 1], None, [180, 256], [0, 180, 0, 256])
    cv.normalize(hist2, hist2, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)

    # Compare histogram
    dist = cv.compareHist(hist1, hist2, cv2.HISTCMP_CHISQR)
    return dist
```

- Nếu khoảng cách đó nhỏ hơn 1 ngưỡng nhất định (HISTOGRAM\_DIST\_THRESHOLD) thì được coi là chưa phát hiện chuyển cảnh và tiếp tục đọc các frame tiếp theo,

- Nếu khoảng cách đó lớn hơn ngưỡng thì được coi là phát hiện chuyển cảnh và tiến hành chọn frame đại diện:

Tính trung bình giá trị điểm ảnh của các frame trong shot:

- Nếu trung bình này lớn hơn ngưỡng MEAN\_FRAME\_THRESHOLD (ngưỡng kiểm tra cảnh quay không quá tối) và ngưỡng LEN\_FRAME\_THRESHOLD ( khoảng thời gian của shot vì khi chuyển cảnh giữa 2 shot, cảnh chuyển từ tối dần đến sáng dần - kiểm tra ngưỡng để loại bỏ trường hợp này) thì tiếp tục duyệt từng frame:
- Kiểm tra khoảng cách giữa trung bình điểm ảnh từng frame và trung bình giá trị điểm ảnh của các frame trong shot, chọn frame có khoảng cách gần nhất làm frame đại diện cho shot:

```
if meanstack > config.MEAN_FRAME_THRESHOLD and len(stackMean) > config.LEN_FRAME_THRESHOLD:
    for i in range(len(stackMean)):
        val = stackMean[i]
        # So sánh lay frame gan voi frame trung binh nhat
        if abs(val - meanstack) < valMin:
            valMin = abs(val - meanstack)
            valRet = stackIndex[i]

    # Lưu vị trí start_shot, end_shot, fps (cho front-end tách shot)
    short_start.append(stackIndex[0])
    short_end.append(stackIndex[-1])
    fpss.append(fps)
    #Lay vị trí frame đại diện cho shot do
    selectIndex.append(valRet)
```

### 2.3. Phát hiện mặt người sử dụng Haar Cascade:

Với mỗi frame xử lý, phát hiện và cắt ra các ảnh chứa mặt người trong frame sử dụng các đặc trưng loại Haar.

- Phát hiện ra mặt người xuất hiện trong frame với các tham số:

scaleFactor = 1.3

minNeighbors = 9

minSize = ( 30, 30)

```
def faces(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Load các window đặc trưng kiểu Haar
    face_cascade = cv2.CascadeClassifier(FaceDetect.PREFIX+'haarcascade_frontalface_default.xml')
    # Detect các face trên ảnh
    faces = face_cascade.detectMultiScale(gray, 1.3, 9, minSize=(30, 30))
    return faces
```

- Cắt ảnh chứa mặt người và resize về 300x300:

```
def faces_crop(img, faces):
    rs = []
    # Cat face
    for (x, y, w, h) in faces:
        crop = img[y:y + h, x:x + w]
        crop = cv2.resize(crop, (300, 300))
        rs.append(crop)
    return np.array(rs)
```

#### 2.4.Sử dụng LBPH để rút trích đặc trưng ảnh:

Từ 1 ảnh chứa mặt người đầu vào, tiến hành rút trích đặc trưng LBP:

Tiến hành đọc ảnh:

```
def local_binary_pattern(img):
    height, width, _ = img.shape
    img = cv.cvtColor(img, cv.COLOR_RGB2GRAY)
    img_lbp = np.zeros((height, width),
                       np.uint8)
    for i in range(0, height):
        for j in range(0, width):
            img_lbp[i, j] = lbp_calculated_pixel(img, i, j)

    return img_lbp
```

So sánh 8 hàng xóm gần nhất với điểm trung tâm ( nhỏ hơn center thì bằng 0, ngược lại bằng 1)

```

def get_pixel(img, center, x, y):
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1
    except:
        pass
    return new_value

def lbp_calculated_pixel(img, x, y):
    center = img[x][y]
    val_ar = []
    # top_left
    val_ar.append(get_pixel(img, center, x - 1, y - 1))
    # top
    val_ar.append(get_pixel(img, center, x - 1, y))
    # top_right
    val_ar.append(get_pixel(img, center, x - 1, y + 1))
    # right
    val_ar.append(get_pixel(img, center, x, y + 1))
    # bottom_right
    val_ar.append(get_pixel(img, center, x + 1, y + 1))
    # bottom
    val_ar.append(get_pixel(img, center, x + 1, y))
    # bottom_left
    val_ar.append(get_pixel(img, center, x + 1, y - 1))
    # left
    val_ar.append(get_pixel(img, center, x, y - 1))
    # Now, we need to convert binary
    # values to decimal
    power_val = [1, 2, 4, 8, 16, 32, 64, 128]
    val = 0
    for i in range(len(val_ar)):
        val += val_ar[i] * power_val[i]

    return val

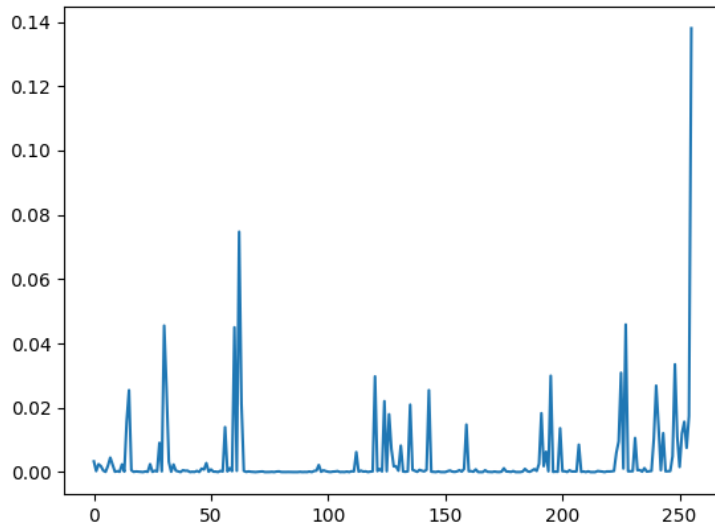
```

- Biểu đồ Histogram cho LBP:

```

def histogram_bit(img, rotation = False):
    if rotation:
        hist, _ = np.histogram(img, bins=np.arange(256+1), range=[0, 256], density=True)
    else:
        hist, _ = np.histogram(img, bins=256, range=[0, 256], density=True)
    return hist

```



•

## 2.5. Tìm kiếm video từ ảnh đầu vào

Từ 1 file ảnh đầu vào, tiến hành cắt face từ ảnh sử dụng Haar, sau đó đưa qua LBPH để rút trích đặc trưng.

Tiến hành tính khoảng cách Euclide giữa các đặc trưng của ảnh đầu vào và đặc trưng của ảnh face trong hệ thống.

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2}$$

```

def search_image_euclidean(db: Session, img_src):
    rs = []
    # Tach cac face tu anh dau vao
    faces = FaceDetect.faces(img_src)
    # Cat cac face
    faces_crop = FaceDetect.faces_crop(img_src, faces)
    # Lay ds tat ca cac lbp_feature tu db
    lbp_features: List[models.LbpFeature] = db.query(models.LbpFeature).all()
    # duyet tung face
    for facecrop in faces_crop:
        # local_binary_pattern tren moi face
        lbp_src = FeatureExtract.local_binary_pattern(facecrop)
        # histogram anh xam local_binary_pattern
        lbp_src = FeatureExtract.histogram_bit(lbp_src)
        # so sanh feature anh dau vao voi tung cac feature da co
        for lbp in lbp_features:
            path_dist = os.path.join(config.DATA_PATH, lbp.lbp_feature_path)
            #Load feature tu folder vat ly
            with open(path_dist, 'r') as file:
                lbp_dist = np.loadtxt(file)

            #So sanh theo khoang cach khoảng cách Euclid
            dist = np.sqrt(np.nansum(np.square(lbp_src - lbp_dist)))

            # print(lbp.face.frame.frame_path, dist)

            # Xet nguoi khoang cach de lua chon face/frame/shot/video chuan nhat
            if dist <= 0.04:
                print("ok", lbp.face.frame.frame_path, dist)
                rs.append({
                    "lbp": lbp,
                    "dist": dist
                })

    #Sort uu tien khoang cach gan nhat
    rs.sort(key=lambda t: t["dist"])

    return rs, faces_crop

```

### 3. Demo hệ thống

## Nhóm 12 | CSDL\_DPT

Hà Thị Kim Phụng, Nguyễn Thị Hà, Nguyễn Thùy Linh

Image input

00615.png

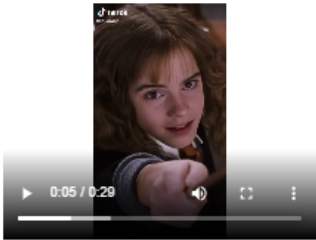

Image Search:



Faces:



Result:

#	Video Path	Video	Shot start	Shot end
1	00003\00003.mp4		5s	6s
2	00040\00040.mp4		16s	18s

*Demo khi dữ liệu có trong hệ thống*



## Nhóm 12 | CSDL\_DPT

Hà Thị Kim Phụng, Nguyễn Thị Hà, Nguyễn Thùy Linh

Image input

Chọn tệp ronaldo2.png

Image Search:



Faces:



Result:

#	Video Path	Video	Shot start	Shot end
NULL	NULL	NULL	NULL	NULL

*Dữ liệu không có trong hệ thống*