**Tìm hiểu các hàm của OpenCV**

**trong các ứng dụng của xử lý ảnh**

*Bài tiểu luận nhóm 3 – môn Xử Lý Ảnh*

*Các thành viên của nhóm:*

* *Vũ Văn Hùng*
* *Phạm Văn Yên*
* *Phạm Hoàng Tiến*
* *Nguyễn Trung Đức*

# Histogram

## **Khái niệm**

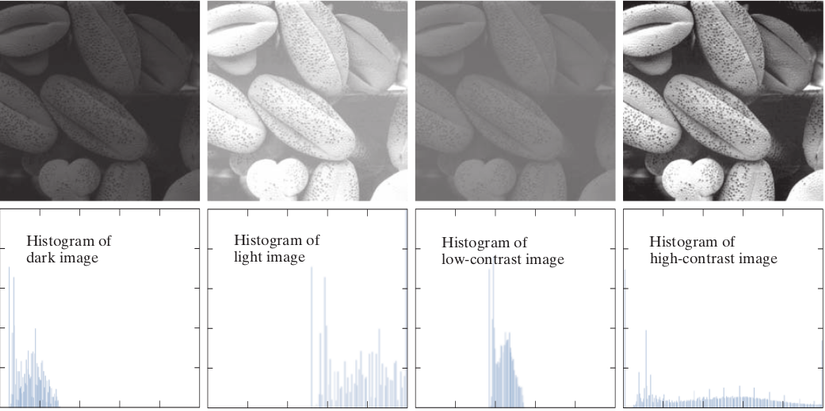
Histogram (lược đồ xám) là biểu đồ tần xuất thống kê số lần xuất hiện các mức sáng trong ảnh.

## **Cách tính histogram**

* r\_{k} là mức xám của ảnh f(x,y)
* n\_{k} là số điểm ảnh (pixels) có giá trị r\_{k}
* Biểu đồ mức xám chưa chuẩn hóa (unnormalized histogram) của $f$ được định nghĩa như sau: h(r\_{k}) = n\_{k} với k = 0, 1,..., L-1, L là số mức xám.
* Biểu đồ chuẩn hoá (normalized histogram): p(r\_{k}) = \frac{h(r\_{k})}{MN} = \frac{n\_{k}}{MN} với M, N là chiều dài và chiều rộng của f hay là của ảnh

## **Ví dụ**

Ta lần lượt có ảnh và histogram tương ứng với các ảnh drank, light, low-contrast và high-contrast



Ta có nhận xét

* Với ảnh dark thì histogram có các cột tập trung vào bên trái tương ứng với màu tối
* Với ảnh light thì histogram có tập trung vào bên phải chứa các pixel trắng
* Với ảnh độ tương phản thấp (low-contrast) thì histogram có các cột tập trung xít nhau và ở giữa
* Với ảnh độ tương phản cao (high-contrast) thì histogram san đều với các giá trị

## **Code**

Giả sử ta có bức ảnh độ tương phản như sau:



Ta bắt đầu import thư viện và load ảnh.

import numpy as np

import cv2

import matplotlib.pyplot as plt

%matplotlib inline

plt.rcParams['figure.figsize'] = [10,8]

img = cv2.imread("low-exposure.jpg", 0)

Chúng ta bắt đầu thử bắt đầu code nào :laughing:. Ta có 2 cách dùng hàm để tính histogram.

1. *Sử dụng cv2.calcHist từ thư viện OpenCV*

# using cv2.calcHist()

hist = cv2.calcHist(

[img],

channels = [0],

mask=None, # full image

histSize=[256], #full scale

ranges=[0,256]

)

plt.plot(hist)

1. *Sử dụng hàm numpy.histogram*

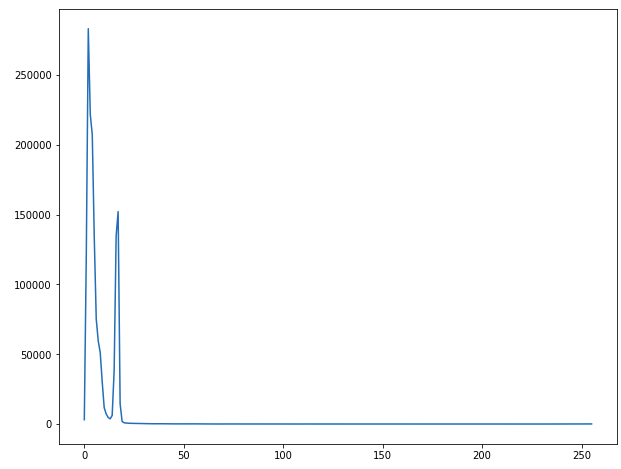
#using numpy

h2 = np.histogram(img.ravel(), bins=256, range=[0,256])

print(h2[0].shape)

plt.plot(h2[0])

Và 2 cách trên đều ra kết quả như sau:



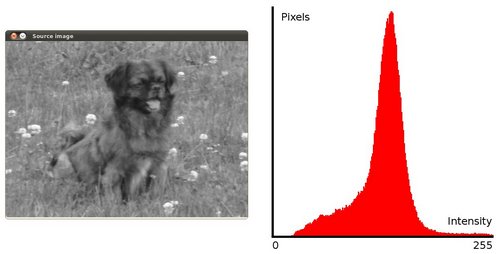
# Histogram equalization (cân bằng biểu đồ mức xám)

## **Khái niệm**

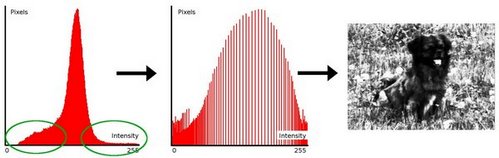
### Cân bằng biểu đồ là gì?

* Đây là một phương pháp cải thiện độ tương phản trong hình ảnh, để kéo dài phạm vi cường độ.
* Để làm rõ hơn, từ hình ảnh trên, bạn có thể thấy rằng các pixel dường như được nhóm lại xung quanh giữa phạm vi cường độ có sẵn. Những gì Cân bằng biểu đồ làm là mở rộng phạm vi này. Hãy xem hình bên dưới: Các vòng tròn màu xanh lá cây cho biết cường độ dân số thấp . Sau khi áp dụng cân bằng, chúng tôi nhận được một biểu đồ như hình ở giữa. Hình ảnh kết quả được hiển thị trong hình bên phải.

Ảnh ban đầu:



Ảnh sau khi cân bằng:



**Giải thuật**

parser = argparse.ArgumentParser (description = 'Code cho hướng dẫn cân bằng biểu đồ.' )

parser.add\_argument ( '--input' , help = 'Đường dẫn đến hình ảnh đầu vào.' , default = 'lena.jpg' )

args = parser.parse\_args ()

src = [cv.imread](https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html" \l "ga288b8b3da0892bd651fce07b3bbd3a56) ( [cv.samples.findFile](https://docs.opencv.org/3.4/d6/dba/group__core__utils__samples.html" \l "ga3a33b00033b46c698ff6340d95569c13) (args.input))

nếu src là Không :

[print](https://docs.opencv.org/3.4/df/d57/namespacecv_1_1dnn.html#a701210a0203f2786cbfd04b2bd56da47) ( 'Không thể mở hoặc không tìm thấy hình ảnh:' , args.input)

thoát ra (0)

* Chuyển nó sang thang độ xám:

src = [cv.cvtColor](https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html" \l "ga397ae87e1288a81d2363b61574eb8cab) (src, cv.COLOR\_BGR2GRAY)

* Áp dụng cân bằng biểu đồ với hàm **[cv :: equalizeHist](https://docs.opencv.org/3.4/d6/dc7/group__imgproc__hist.html" \l "ga7e54091f0c937d49bf84152a16f76d6e)** :

dst = [cv.equalizeHist](https://docs.opencv.org/3.4/d6/dc7/group__imgproc__hist.html" \l "ga7e54091f0c937d49bf84152a16f76d6e) (src)

Như có thể dễ dàng thấy, các đối số duy nhất là hình ảnh gốc và hình ảnh đầu ra (cân bằng).

* Hiển thị cả hai hình ảnh (gốc và cân bằng):

[cv.imshow](https://docs.opencv.org/3.4/df/d24/group__highgui__opengl.html#gaae7e90aa3415c68dba22a5ff2cefc25d) ( 'Hình ảnh nguồn' , src)

[cv.imshow](https://docs.opencv.org/3.4/df/d24/group__highgui__opengl.html#gaae7e90aa3415c68dba22a5ff2cefc25d) ( 'Hình ảnh được cân bằng' , dst)

* Chờ cho đến khi người dùng tồn tại chương trình

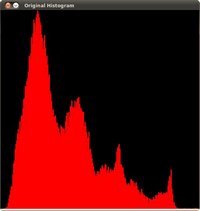
[cv.waitKey](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga5628525ad33f52eab17feebcfba38bd7) ()

## **Các kết quả**

1. Để đánh giá cao hơn kết quả cân bằng, hãy giới thiệu một hình ảnh không có nhiều độ tương phản, chẳng hạn như:



nhân tiện, có biểu đồ này:

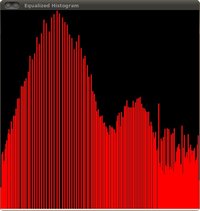


lưu ý rằng các pixel được nhóm xung quanh tâm của biểu đồ.

1. Sau khi áp dụng cân bằng với chương trình của chúng tôi, chúng tôi nhận được kết quả sau:



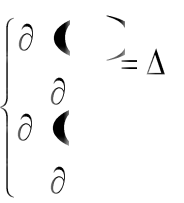
hình ảnh này chắc chắn có độ tương phản cao hơn. Kiểm tra biểu đồ mới của nó như thế này:



Lưu ý cách số lượng pixel được phân phối nhiều hơn thông qua phạm vi cường độ.

1. **Phương pháp phát hiện biên Gradient**

* **Phương pháp Gradient** là phương pháp dò biên cục bộ bằng cách tìm kiếm cực đại và cực tiểu khi lấy đạo hàm bậc nhất của ảnh trong không gian hai chiều.
* Theo định nghĩa, Gradient là một vector có các thành phần biểu thị tốc độ thay đổi giá trị của điểm ảnh:



|  |  |  |  |
| --- | --- | --- | --- |
| *f x*, *y* | *x* | *f* (*x dx*, *y*) | *f* (*x*, *y*) |
|  |  |  |
| *x* | *dx* |  |
|  |  |
| *f x*, *y* | *y* | *f* (*x*, *y dy*) | *f* (*x*, *y*) |
|  |  |  |
| *x* | *dx* |  |
|  |  |



* Trong đó dx, dy là khoảng cách giữa 2 điểm lân cận theo hướng x và y tương ứng. Trong những hình ảnh rời rạc, ta có thể xem xét dx, dy về số lượng điểm ảnh giữa hai điểm.
* Thực tế ta chọn dy = dx = 1 (khoảng cách điểm ảnh) là điểm mà tại đó tọa độ điểm ảnh là (i, j), do đó:



*x f* (*i* 1, *j*) *f* (*i*, *j*)



*y f* (*i*, *j* 1) *f* (*i*, *j*)

* Do tính chất phức tạp trong tính toán khi áp dụng phương pháp Gradient trong xử lý ảnh, người ta sử dụng kỹ thuật Gradient dùng cặp mặt nạ H1, H2 trực giao (theo 2 hướng vuông góc). Nếu định nghĩa g1, g2 là Gradient theo hai hướng x, y tướng ứng thì biên độ g(m,n) tại điểm (m,n) được tính:

*g* (*m*, *n*)  *g*12(*m*, *n*) *g*22(*m*, *n*) *A*0



*r* (*m*, *n*) *artg*(*g*2(*m*, *n*))



Để giảm độ phức tạp tính toán, A0 được tính gần đúng như sau:

*A*0 *g*1(*m*, *n*) *g*2(*m*, *n*)



* Việc xấp xỉ đạo hàm bậc nhất theo các hướng x và y được thực hiện thông qua 2 mặt nạ nhân chập tương ứng sẽ cho ta các kỹ thuật phát hiện biên khác nhau ( Sobel, Prewitt,..).

*\* Thuật toán:*

*− Với mỗi điểm ảnh I(x,y) tính:*

|  |  |  |  |
| --- | --- | --- | --- |
| *Grad*(*I* (*x*, *y*))(*I* (*x*, *y*) *H* )2 | (*I* (*x*, *y*) *H* | 2 | )2 |
| 1 |  |  |



*− Phân ngưỡng:*

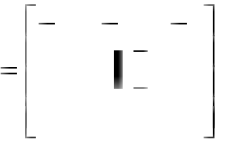
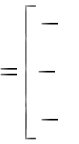
**

|  |  |  |  |
| --- | --- | --- | --- |
| *I* (*x*, *y*) | 1 | if | *I* (*x*, *y*) |
| 0 | if | *I* (*x*, *y*) |
|  |



1. **Toán tử Sobel - Phương pháp phát hiện biên Sobel**

* Ở đây chúng ta sử dụng hai mặt nạ nhân chập 3 \* 3 theo hướng x, y với hình ảnh đa mức xám.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 0 | 1 |  | 1 | 2 | 1 |
| *H x* | 2 | 0 | 2 | *H y* | 0 | 0 | 0 |
|  | 1 | 0 | 1 |  | 1 | 2 | 1 |



Hướng của gradient



*\* Thuật toán*:

* *Bước 1: tính* *I(x,y) ⊗ Hx = I1 và* *I(x,y) ⊗ Hy = I2*
* *Bước 2: Tính | I1 | +| I2 | = IS*
* *Bước 3: Hiệu chỉnh I(x,y) = Is ≥ θ ? 1 : 0*

Trong OpeCV, hàm cài đặt phép lọc này như sau:

*cv2.Sobel(*

*const Mat& src,*

*Mat& dst,*

*int ddepth,*

*int xorder,*

*int yorder,*

*int ksize,*

*double scale,*

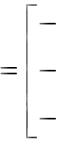
*double delta,*

*int borderType)*

* Trong đó, src và dst là ảnh gốc và ảnh qua phép lọc. ddepth là độ sâu của ảnh sau phép lọc, có thể là CV\_32F, CV\_64F … . xoder và yoder là các đạo hàm theo hướng x và y, để tính đạo hàm theo hướng nào ta đặt giá trị đó lên 1, ngược lại nếu giá trị bằng 0, hàm cài đặt sẽ bỏ qua không tính theo hướng đó. Scale và delta là hai thông số tùy chọn cho việc tính giá trị đạo hàm lưa giá trị vi sai vào ảnh sau phép lọc, chúng có giá trị mặc định là 1 và 0. borderType là tham số

1. **Toán tử Prewitt - Phương pháp phát hiện biên Prewitt**

* Toán tử Prewitt làm việc giống với cách làm của toán tử Sobel, nhưng sử dụng mặt nạ nhân chập khác so với toán tử Sobel. Mặt nạ này cho kết quả giống như Sobel và cùng hướng x, y:



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 0 | 1 |  | 1 | 1 | 1 |
| *H x* | 1 | 0 | 1 | *H y* | 0 | 0 | 0 |
|  | 1 | 0 | 1 |  | 1 | 1 | 1 |



Hướng của gradient

*Thuật toán: Các bước tính toán tương tự như Sobel*

* Ta sử dụng filer2D để tự cài đặt bộ lọc cho riêng các phương pháp khác( cụ thể ở đây là prewitt)

*cv2.filter2D(*

*const Mat& src,*

*Mat& dst,*

*int ddepth,*

*const Mat& kernel,*

*Point anchor,*

*double delta,*

*int borderType).*

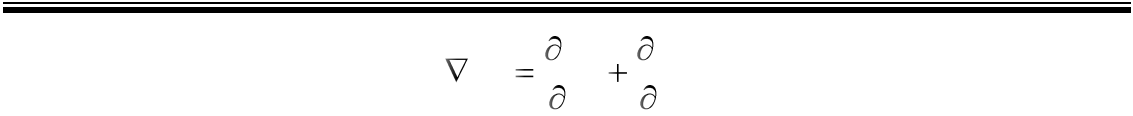
* Trong đó, src và dst là ảnh gốc và ảnh thu được qua phép lọc, kernel là ma trân lọc. Thông số anchor để chỉ ra tâm của ma trận, delta điều chỉnh độ sáng của ảnh sau phép lọc (ảnh sau phép lọc được cộng với delta và borderType là kiểu xác định những pixel nằm ngoài vùng ảnh. Hàm cv::filter2D thực chất là hàm tính toán nhân chập giữa ảnh gốc và ma trận lọc để cho ra ảnh cuối sau phép lọc. Như vậy qua trên ta thấy rằng để tiến hành việc lọc ảnh ta chỉ cần định nghĩa một ma trận lọc kernel.

Với prewiit ta sử dụng 2 kernel đó là kernel = Hx và kernely = Hy

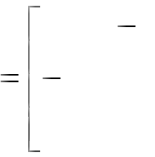
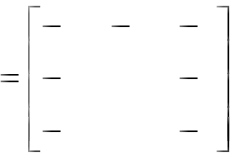
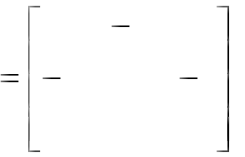
1. **Phương pháp phát hiện biên Laplace**

* Để khắc phục hạn chế và nhược điểm của phương pháp Gradient, trong đó sử dụng đạo hàm riêng bậc nhất người ta nghĩ đến việc sử dụng đạo hàm riêng bậc hai hay toán tử Laplace. Phương pháp dò biên theo toán tử Laplace hiệu quả hơn phương pháp toán tử Gradient trong trường hợp mức xám biến đổi chậm, miền chuyển đổi mức xám có độ trải rộng.
* Toán tử Laplace được định nghĩa như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 *f* | 2 *f* |  |  | 2 *f* |
| *x*2 |  |  | *y*2 |
|  |  |  |



* Toán tử Laplace dùng một số kiểu mặt nạ khác nhau nhằm tính gần đúng đạo hàm riêng bậc hai. Các dạng mặt nạ theo toán tử Laplace bậc 3x3 hay dùng:

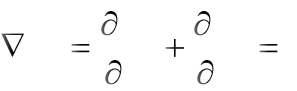
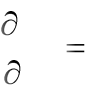
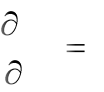


|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 0 |  | 1 | 1 | 1 |  | 1 | 2 | 1 |
| *H*1 | 1 4 | | 1 | *H*2 | 1 | 8 | 1 | *H*3 | 2 4 | | 2 |
|  | 0 | 1 | 0 |  | 1 | 1 | 1 |  | 1 | 2 | 1 |



* Ghi chú: Mặt nạ H1 còn cải biên bằng việc lấy giá trị ở tâm bằng 8 thay vì giá trị 4.
* Để thấy rõ việc xấp xỉ đạo hàm riêng bậc 2 trong không gian 2 chiều với mặt nạ H1 làm ví dụ, ta có thể tính gần đúng như sau:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 2 *f* | 2 *f* (*x*, *y*) | *f* (*x* 1, *y*) | *f* (*x* 1, *y*) |
|  |  |  |  |  |  | *x*2 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | 2 *f* | 2 *f* (*x*, *y*) | *f* (*x*, *y* 1) | *f* (*x*, *y* 1) |
|  |  |  |  |  |  | *y*2 |
|  |  |  |  |  |  |  |  |  |
|  | 2 *f* | 2 *f* |  | 2 *f* |  | 4 *f* (*x*, *y*) *f* (*x* 1, *y*) *f* (*x*, *y* 1) *f* (*x* 1, *y*) *f* (*x*, *y* 1) | | | |
| *x*2 |  | *y*2 | |
|  |  |  |  |  |  |  |



* Kỹ thuật theo toán tử Laplace tạo đường biên mảnh (có độ rộng 1 pixel). Nhược điểm của kỹ thuật này rất nhạy với nhiễu, do vậy đường biên thu được thường kém ổn định. Để khắc phục nhược điểm này người ta mở rộng toán tử Laplace – dùng hàm Gauss (Laplace of Gauss) để giảm nhiễu cho ảnh (làm trơn ảnh).
* Trong OpenCV, bộ lọc này được cài đặt qua hàm:

*cv2.Laplacian(*

*const Mat& src,*

*Mat& dst,*

*int ddepth,*

*int ksize=1,*

*double scale=1,*

*double delta=0,*

*int borderType)*

* Các thông số này có ý nghĩa giống như các thông số trong bộ lọc Sobel, chỉ khác ở chỗ ksize là một giá trị int mặc định bằng 1 và khi đó ma trận lọc laplace trên được áp dụng.

1. **Gaussian Blur**

* **Định nghĩa** : Gaussian Blur là cách làm mờ một ảnh bằng hàm Gaussian. Phương pháp này được ứng dụng một cách rộng rãi và hiệu quả trong các phần mềm xử lý đồ họa. Nó cũng là công cụ phổ biến để thực hiện quá trình tiền xử lý (preprocessing) hình ảnh dùng làm dữ liệu đầu vào tốt cho các phân tích cao cấp hơn như trong Computer Vision, hoặc cho các giải thuật được thực hiện trong một tỉ lệ khác của hình được cho. Nó có thể giúp làm giảm nhiễu (Noise) và mức độ chi tiết (không mong muốn) của hình ảnh.
* **Công thức toán học**
* Gaussian blur là một bộ lọc làm mờ hình ảnh sử dụng chức năng Gaussian (cũng thể hiện phân phối bình thường trong thống kê) để tính toán biến đổi để áp dụng cho từng pixel trong ảnh. Công thức của hàm Gaussian trong một chiều là

[https://2.bp.blogspot.com/-iGYZ-wOGKFQ/XGGVxNVL1-I/AAAAAAAAC3w/nzMKnob3FYAzRf2g8bAUUMKCE5qzpwSwgCLcBGAs/s1600/1d.PNG](https://2.bp.blogspot.com/-iGYZ-wOGKFQ/XGGVxNVL1-I/AAAAAAAAC3w/nzMKnob3FYAzRf2g8bAUUMKCE5qzpwSwgCLcBGAs/s1600/1d.PNG)

* Trong không gian hai chiều, nó là kết quả của hai hàm Gaussian 1 chiều, mỗi chiều là một trục x, y:

https://4.bp.blogspot.com/-2niQU_0iEAg/XGGVxfy6X_I/AAAAAAAAC30/kQR3jBkMqysWgS8efXB81iSFG5ft-tHRwCLcBGAs/s1600/2d.PNG

* Trong đó x là khoảng cách từ gốc tọa độ trong trục hoành, y là khoảng cách từ gốc tọa độ trong trục tung và σ là độ lệch chuẩn của phân bố Gaussian. Khi được áp dụng theo hai chiều, công thức này tạo ra một bề mặt có các đường viền là các vòng tròn đồng tâm có phân tán Gaussian từ điểm trung tâm. Các giá trị từ phân tán này được sử dụng để xây dựng ma trận tích chập được áp dụng cho ảnh gốc. Quá trình tích chập này được minh họa trực quan trong hình.

[](https://upload.wikimedia.org/wikipedia/commons/thumb/d/d7/Halftone%2C_Gaussian_Blur.jpg/220px-Halftone%2C_Gaussian_Blur.jpg)

* Giá trị mới của mỗi Pixel sau khi tính tích chập với kernel đại diện cho hàm Gaussian có thể coi là trung bình lượng giá của các pixel xung quanh nó. Ta thấy rằng giá trị lượng giá của phần tử trung tâm kernel tương ứng với pixel đang xét là lớn nhất, giá trị này sẽ nhỏ hơn đối với các phần tử tương ứng với những pixel kế cận một cách đối xứng và tỉ lệ thuận với khoảng cách của phần tử này với trung tâm. Tính chất này giúp giữ lại đường viền và biên cũng như làm mờ một cách đồng bộ hơn so với các phương pháp khác.
* Trong lý thuyết, hàm Gaussian tại mỗi điểm trên hình là khác 0. Điều này có nghĩa là Gaussian Kernel nên có kích thước bằng với hình ảnh và giá trị tại mỗi phần tử luôn khác 0. Tuy nhiên trong thực hành, do việc tính tóan dựa trên xấp xỉ rời rạc (Discrete Approximation) cho nên giá trị của các phần tử trên bề mặt Gaussian ở khỏang cách lớn hơn 3σ so với trung tâm gần như không đáng kể (tiệm cận 0). Do vậy các phân tán Gaussian ngòai bán kính này sẽ bị bỏ qua, đó cũng là lý do mà thông thường Gaussian kernel có kích thước giới hạn 3, 5, 7... (Cái này còn tùy vào giá trị phương sai chuẩn mà bạn chọn). Khoảng cách giữa hai điểm gần nhau trong Gaussian Kernel là σ.
* Do đặc tính có cấu trúc như những hình tròn đối xứng, với hình hai chiều, Gaussian blur có thể được áp dụng như là hai phép tính toán một chiều độc lập (Độc lập tuyến tính - Linearly Separable). Có nghĩa là hiệu quả thu được từ tính toán trên ma trận hai chiều có thể tương đương với cách ứng dụng tính toán một loạt các Gaussian 1 chiều theo hướng ngang và đứng. Điều này rất hữu ích trong việc giảm chi phí tính toán thể hiện khá rỏ trong hai công thức tương ứng sau:

[https://2.bp.blogspot.com/-bhuwlgWzyFw/XGGZCQA55RI/AAAAAAAAC4M/2JtQ_kMAJUQtQMZkWAe8vKmoqAB2awSYwCLcBGAs/s400/5.PNG](https://2.bp.blogspot.com/-bhuwlgWzyFw/XGGZCQA55RI/AAAAAAAAC4M/2JtQ_kMAJUQtQMZkWAe8vKmoqAB2awSYwCLcBGAs/s1600/5.PNG)

* + - thời gian (trong đó h là chiều cao và w là chiều rộng), trái ngược với

[https://4.bp.blogspot.com/-K4Acl8Fa1qQ/XGGZR53_1tI/AAAAAAAAC4Q/LWEMUPQQwo4MRDW2eWfdaHxcqkxAzseqQCLcBGAs/s320/7.PNG](https://4.bp.blogspot.com/-K4Acl8Fa1qQ/XGGZR53_1tI/AAAAAAAAC4Q/LWEMUPQQwo4MRDW2eWfdaHxcqkxAzseqQCLcBGAs/s1600/7.PNG)

* + - của 1 kernel chưa phân tách.
* Một vấn đề cần quan tâm nữa của Gaussian blur là nếu bạn sử dụng lần lượt nhiều Gaussian cho một ảnh thì kết quả cũng tương đương với khi bạn dùng một Gaussian lớn hơn có bán kính bằng căn bậc hai tổng bình phương các bán kính của các Gaussian đã dùng, ví dụ:

[https://2.bp.blogspot.com/_b1r1UJs_aVQ/Sjo-6a2YOTI/AAAAAAAAAGo/s57HK7-w3xE/s400/20090105070801687.jpg.jpg](https://2.bp.blogspot.com/_b1r1UJs_aVQ/Sjo-6a2YOTI/AAAAAAAAAGo/s57HK7-w3xE/s400/20090105070801687.jpg.jpg)

* Cũng vì mối quan hệ này mà thời gian tính toán sẽ không thể được tiết kiệm khi bạn chia nhỏ các Gaussian. Do đó khuyến cáo nên sử dụng Gaussian đơn trong tính toán nếu có thể.

Các sử dụng:

**cv.GaussianBlur(**

**src,**

**ksize,**

**sigmaX[, dst[, sigmaY[, borderType=BORDER\_DEFAULT]]] )**

| Tham số | Sự miêu tả |
| --- | --- |
| src | hình ảnh đầu vào |
| ksize | Kích thước hạt nhân Gaussian. [chiều cao chiều rộng]. chiều cao và chiều rộng phải là số lẻ và có thể có các giá trị khác nhau. Nếu ksize được đặt thành [0 0], thì ksize được tính từ các giá trị sigma. |
| sigmaX | Độ lệch chuẩn của hạt nhân dọc theo trục X (hướng ngang). |
| sigmaY | Độ lệch chuẩn của hạt nhân dọc theo trục Y (hướng dọc). Nếu sigmaY = 0, thì giá trị sigmaX được lấy cho sigmaY |
| borderType | Chỉ định ranh giới hình ảnh trong khi hạt nhân được áp dụng trên đường viền hình ảnh. Các giá trị có thể là: cv.BORDER\_CONSTANT cv.BORDER\_REPLICATE cv.BORDER\_REFLECT cv.BORDER\_WRAP cv.BORDER\_REFLECT\_101 cv.BORDER\_TRANSPARENT cv.BORDER\_REFLECT101 cv.BORDER\_DEFAULT cv.BORDER\_ISOLATED |

1. **Trích xuất đặc trưng**

* **Xử lý ảnh là gì ?**
* Như ta đã biết xử lý ảnh là quá trình tạo ra một ảnh mới từ ảnh đã có theo một cách nào đó thường là tối ưu hoặc nâng cao nội dung bức ảnh. Vậy trích xuất đặc trưng là một dạng thao tác nâng cao của qúa trình xử lý ảnh.
* **Mục đích của trích xuất đặc trưng:**
* Trích xuất đặc trưng là trích xuất các đặc tính của ảnh thông qua cường độ điểm ảnh.
* Một số đặc trưng tiêu biểu của ảnh:
* Màu sắc
* Kết cấu
* Hình dạng
* Đặc trưng cục bộ SIFT
* **Ứng dụng:**
* Ứng dụng của trích xuất đặc trưng là áp dụng cho các bài toán tìm kiếm ảnh hoặc một số bài toán nhận dạng điển hình như: nhận dạng biển số xe, nhận dạng mặt người, …

1. **Tìm hiểu chi tiết các đặc trưng được dùng để trích xuất**

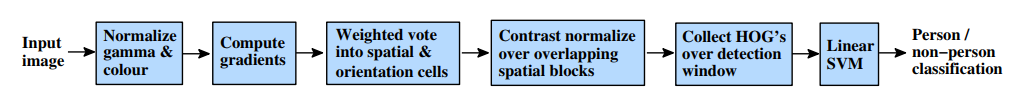
* **Đặc trưng màu sắc:**
* Được áp dụng phổ biến cho các hệ thống tìm kiếm theo nội dung – tìm kiếm ảnh theo lược đồ màu. Đây là phương pháp đơn giản, tốc độ tìm kiếm tương đối nhanh nhưng kết quả tìm kiếm có độ chính xác không cao.
* Một số lược đồ màu được sử dụng như: lược đồ màu RGB, lược đồ màu HSI, lược đồ HSI cải tiến, … trong đó lược đồ RGB phổ biến nhất.
* Lược đồ màu RGB:
* Đối với ảnh 256 màu, lược đồ màu của ảnh tương đương với lược đồ màu của ảnh xám. Đối với ảnh 24 bit màu, lược đồ miêu tả khả năng kết nối về cường độ của ba kênh màu R, G, B. Luợc đồ màu này được định nghĩa như sau:



* Trong đó N là số lượng điểm có trong ảnh. Lược đồ màu này được tính bằng cách rời rạc hóa từng màu trong ảnh, sau đó đếm số điểm ảnh của mỗi màu. Khi mà số lượng màu là có hạng, để thuận tiện hơn, người ta thường chuyển đổi ba kênh màu thành một biến giá trị duy nhất. Một cách khác để tính lược đồ màu của ảnh RGB là ta phân ra làm 3 lượt đồ riêng biệt hG[] , hR[] , hB[] . Khi đó, mỗi lược đồ được tính bằng cách đếm kênh màu tương ứng trong mỗi điểm ảnh.
* **Đặc trưng kết cấu:**
* Hiện tại, vẫn chưa có một định nghĩa chính thức cụ thể về kết cấu. Kết cấu là một đối tượng dùng để phân hoạch ảnh ra thành những vùng quan tâm để phân lớp những vùng đó.
* Kết cấu cung cấp thông tin về sự sắp xếp về mặt không gian của màu sắc và cường độ một ảnh. Kết cấu được đặc trưng bởi sự phân bổ không gian của những mức cường độ trong một khu vực láng giềng với nhau. Kết cấu gồm các kết cấu gốc hay nhiều kết cấu gộp lại đôi khi gọi là texel.
* Một số phương pháp dùng để trích xuất các đặc trưng kết cấu như:
* Kim tự tháp "có thể lái được" (the steerable pyramid)
* Biến đổi đường viền (the cotourlet transform)
* Biến đổi sóng Gabor (The Gabor Wavelet transform)
* Biểu diễn ma trận đồng hiện (co-occurrence matrix)
* Hệ thống bộ lọc định hướng phức tạp (The complex directional fillter bank)
* **Đặc trưng hình dạng:**
* Màu sắc và kết cấu là những thuộc tính có khái niệm toàn cục trong một ảnh. Trong khi đó, hình dạng không phải là một thuộc tính của ảnh. Nói tới hình dạng không phải là nhắc đến hình dạng của một ảnh. Thay vì vậy, hình dạng có khuynh hướng chỉ đến một khu vực đặc biệt trong ảnh, hay hình dạng chỉ là biên của một đối tượng nào đó trong ảnh.
* Trong tìm kiếm ảnh theo nội dung, hình dạng là một cấp cao hơn so với màu sắc và kết cấu. Nó đòi hỏi sự phân biệt giữa các vùng để tiến hành xử lý về độ đo của hình dạng. Các hệ thống tìm kiếm ảnh theo nội dung thường khai thác hai nhóm biểu diễn hình dạng sau:
* Biểu diễn hình dạng theo đường biên (cotour-based descriptor) : Biểu diễn các đường biên bao bên ngoài
* Biểu diễn theo vùng (region-based descriptor): Biểu diễn một vùng toàn vẹn
* **Đặc trưng cục bộ bất biến:**
* Phần này trình bày phương pháp trích rút các đặc trưng cục bộ bất biến SIFT của ảnh. Các đặc trưng này bất biến với việc thay đổi tỉ lệ ảnh, quay ảnh, đôi khi là thay đổi điểm nhìn và thêm nhiễu ảnh hay thay đổi cường độ chiếu sáng của ảnh. Phương pháp được lựa chọn có tên là Scale-Invariant Feature Transform (SIFT) và đặc trưng trích rút đựợc gọi là đặc trưng SIFT (SIFT Feature). Các đặc trưng SIFT này được trích rút ra từ các điểm hấp dẫn cục bộ (Local Interest Point).
* Điểm hấp dẫn (Interest Point (Keypoint)): Là vị trí (điểm ảnh) "hấp dẫn" trên ảnh. "Hấp dẫn" ở đây có nghĩa là điểm đó có thể có các đặc trưng bất biến với việc quay ảnh, co giãn ảnh hay thay đổi cường độ chiếu sáng của ảnh. Phương pháp trích rút các đặc trưng bất biến SIFT được tiếp cận theo phương pháp thác lọc, theo đó phương pháp được thực hiện lần lượt theo các bước sau:
* **Phát hiện các điểm cực trị Scale-Space** (Scale-Space extrema detection): Bước đầu tiên này tiến hành tìm kiếm các điểm hấp dẫn trên tất cả các tỉ lệ và vị trí của ảnh. Nó sử dụng hàm different-of-Gaussian để xác định tất cả các điểm hấp dẫn tiềm năng mà bất biến với quy mô và hướng của ảnh.
* **Định vị các điểm hấp dẫn** (keypoint localization): Một hàm kiểm tra sẽ được đưa ra để quyết định xem các điểm hấp dẫn tiềm năng có được lựa chọn hay không?
* **Xác định hướng cho các điểm hấp dẫn** (Orientation assignment): Xác định hướng cho các điểm hấp dẫn được chọn
* **Mô tả các điểm hấp dẫn** (Keypoint descriptor): Các điểm hấp dẫn sau khi được xác định hướng sẽ được mô tả dưới dạng các vector đặc trưng nhiều chiều

1. **Các phương pháp trích xuất đặc trưng phổ biến**

* **HOG (Histograms of Oriented Gradients)**
* Phương pháp rút trích đặc trưng hình ảnh HOG xuất bản ở hội nghị CVPR 2005 được đề xuất bởi tác giả là Dalal và Triggs
* HOG đề xuất phương pháp rút trích đặc trưng sử dụng các thống kê histogram về hướng trên ảnh gradient cho bài toán phát hiện người (human detection)
* Các bước thực hiện:

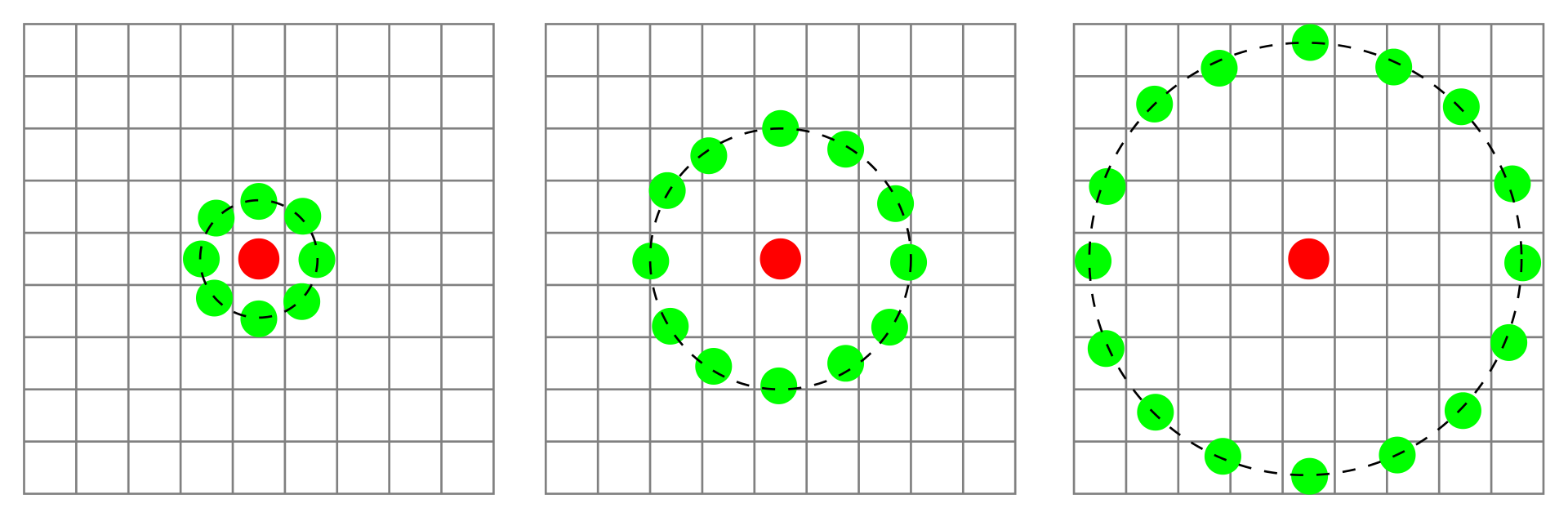


1. Đọc ảnh
2. Tiền xử lý (preprocessing): hình ảnh được đưa qua bước tiền xử lý để thực hiện các thao tác như cân bằng sáng, làm mờ, …
3. Trích đặc trưng ảnh (feature extraction): bằng cách sử dụng các phương pháp rút trích đặc trưng ảnh ta sẽ thu được **vector đặc trưng** của ảnh. Nói một cách nôm na thân quen đó chính là bạn mã hóa hình ảnh thành một vector, và vector này mang những đặc trưng (các số thực) đại diện cho ảnh đó.
4. Huấn luyện mô hình học máy (training): với phương pháp truyền thống, ta thường sử dụng mô hình SVM trong **machine learning** để phân tách các vector đặc trưng thành các lớp cần phân loại.
5. Kiểm thử (validation): sau khi huấn luyện xong mô hình học máy bạn cần phải đánh giá mô hình mình đã huấn luyện đạt độ chính xác là bao nhiêu phần trăm trên tập kiểm thử này. Khi bạn đã hài lòng với kết quả kiểm thử, ta có thể dừng quá trình huấn luyện.

* Trích đặc trưng ảnh nằm ở bước số 2. Đặc trưng hình ảnh rút trích có tốt hay không sẽ ảnh hưởng đến kết quả của độ chính xác. Vì vậy, ở các phương pháp truyền thống, họ đưa ra các thiết kế nhằm cố gắng rút trích thông tin hình ảnh một cách tốt nhất.
* **LBP (Local Binary Patterns)**
* **Local Binary Patterns** (hay còn viết tắt là LBP) là một phương pháp rút trích đặc trưng trong xử lý ảnh. Đặc trưng được rút trích sẽ tiếp tục được tiến hành chọn lọc (feature selection) thu gọn thành vector đặc trưng. Vector đặc trưng này sau đó có thể dùng để đưa vào mô hình học máy để học / phân loại.
* Các bước tiến hành trích xuất đặc trưng theo phương pháp LBP:

1. Duyệt lần lượt từng pixel trên ảnh (theo cột -> theo hàng), với pixel đang xét, ta áp dụng bước 2->4.
2. Xét lần lượt 8 pixel lân cận (hàng xóm - neighbor) của pixel đang duyệt (trung tâm - center). Mỗi pixel hàng xóm sẽ ứng với một bit trong một chuỗi 8-bit. Chuỗi 8-bit này ban đầu sẽ bằng: 00000000. Chuỗi 8-bit này sẽ được cập nhật theo mô tả ở bước 3.
3. Nếu mức sáng tại pixel hàng xóm >= mức sáng tại pixel trung tâm: vote bit ở vị trí tương ứng lên 1 trong chuỗi 8-bit đề cập ở bước 2.
4. Sau khi hoàn tất bước 2 và 3, ta sẽ có mội chuỗi 8-bit (vd: 00101100) -> đổi giá trị nhị phân này sang thập phân để lưu trữ (vd: 00101100 nhị phân = 44 thập phân)
5. Lặp hết toàn ảnh (bước 1->4), ta sẽ có kết quả đầu ra bằng kích thước với ảnh đầu vào. Mỗi giá trị trên ảnh đầu ra là đặc trưng LBP.

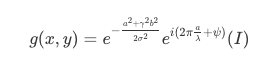
* LBP ta có thể tạm hiểu (để nhớ như sau):
* Local: thể hiện tính chất cục bộ địa phương, đó là khi ở bước 2 ta xét pixel lân cận -> mỗi đặc trưng trong output sẽ mang đặc trưng đại diện cục bộ.
* Binary Patterns: các mẫu hình nhị phân -> cách nhị phân hóa mô tả ở bước 3, 4
* Tổng quát hóa phương pháp tiếp cận LBP trên, ta sẽ có các tham số sau:
* P: Số pixel lận cận pixel trung tâm (vd: P=8).
* R: Bán kính của pixel lân cận mà ta sẽ xét - cách pixel trung tâm bao nhiêu pixel (vd: R=1 nghĩa là liền kề). Kiểu như khoảng cách của nhà bạn và nhà ông hàng xóm mà bạn "quan tâm" í :">.
* Thứ tự các pixel lân cận mã hóa vào chuỗi 8-bit sẽ theo chiều kim đồng hồ hay ngược chiều kim đồng hồ.
* Interpolation: do lấy pixel lân cận theo hình tròn, do đó tọa độ của các pixel lân cận khi tính toán ra sẽ là số thực -> quyết định lấy ra giá trị mức sáng của pixel đó theo cách nào: pixel gần nhất (nearest) hay có trọng số (bilinear). Interpolation này tương tự như khi bạn resize ảnh vậy.



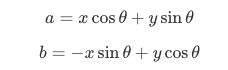
# Gabor filters

## **Trích đặc trưng Gabor filter là gì?**

Gabor filter là phương pháp trích đặc trưng dựa trên những bộ lọc, tên phương pháp này được đặt theo nhà vật lý học [Dennis Gabor](https://en.wikipedia.org/wiki/Dennis_Gabor" \t "_blank). Bộ lọc Gabor được cho là có khả năng phân tích hình ảnh tương tự hệ thống thị giác của con người. Thiết kế bộ lọc Gabor có công thức sau:



với:



Công thức Gabor biểu diễn ở dạng [số phức](https://minhng.info/toan-hoc/so-phuc-la-gi.html), mà số phức thì có phần thực và phần ảo.

Ta có:

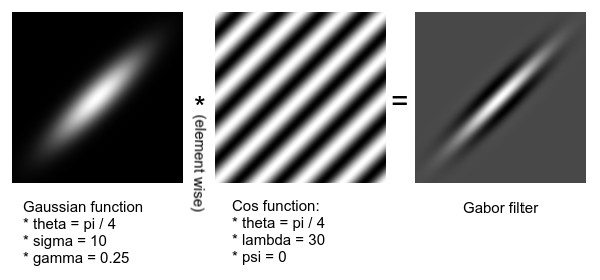


Theo (I) và (II) ta sẽ biểu diễn phần thực của bộ lọc Gabor theo công thức:



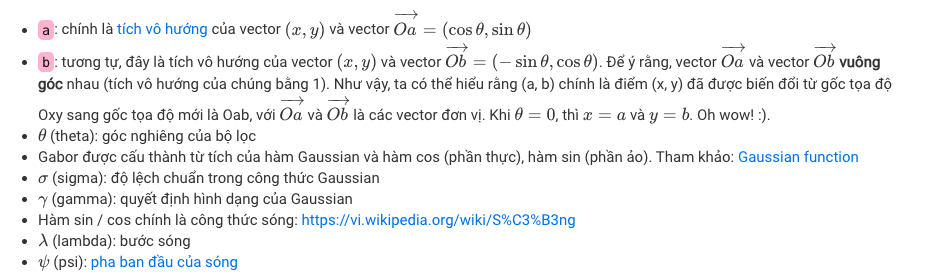
và phần ảo của bộ lọc Gabor theo công thức:





*Ảnh trực quan hóa bộ lọc Gabor filter cấu thành từ hàm Gaussian và hàm cos*

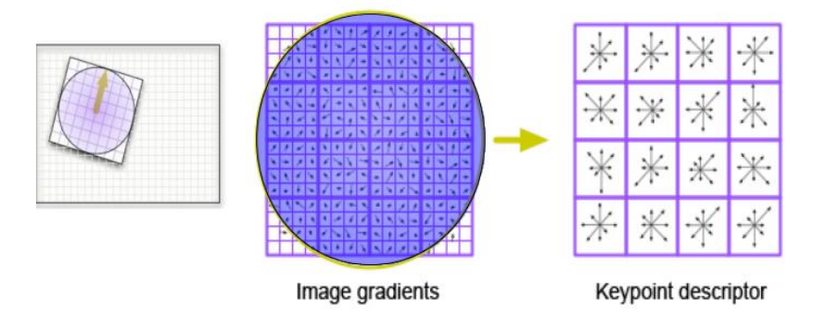
* Một số giải thích cho công thức bộ lọc Gabor:



Tinh chỉnh 5 tham số (θθ, σσ, γγ, λλ, ψψ) của bộ lọc Gabor sẽ cho ra hình dạng bộ lọc khác nhau.

## **Sử dụng Gabor filter**

Công thức Gabor với một bộ tham số sẽ tạo cho ta một bộ lọc đặc thù. Công dụng của Gabor filter là phát hiện cạnh (trích cạnh) của ảnh. Áp dụng bộ lọc Gabor vào ảnh ta sẽ sử dụng [toán tử convolution](https://minhng.info/tutorials/xu-ly-anh-convolution-la-gi.html).

* **SIFT (Scale-invariant feature transform):**
* SIFT (Scale-invariant feature transform) là một feature descriptor được sử dụng trong computer vision và xử lý hình ảnh được dùng để nhận dạng đối tượng, matching image, hay áp dụng cho các bài toán phân loại.
* Với đầu vào là một hình ảnh >>> SIFT >>> các keypoint. Mỗi đối tượng trong hình ảnh sẽ cho ra rất nhiều các keypoint khác nhau, để ta phân biệt được các keypoint này với nhau sẽ thông qua một vector 128 chiều hay còn gọi là descriptor. Các descriptor này sẽ được dùng để nhận dạng đối tượng trong ảnh, hay dùng cho các bài toán classication.
* Hình ảnh sau khi áp dụng biến đổi SIFT, ứng với mỗi keypoint ta sẽ thu được: toạ độ keypoint, scale và orientation của keypoint, descriptor. Các mũi tên trong hình dưới vẽ nhờ vào scale và orientation.  
  
* 4x4 Gradient window
* HIstogram of 4x4 samples per window in 8 directions
* Gaussian weighting around center
* 4x4x8 = 128 dimensional feature vector

##### Đặc điểm :

* Các keypoint sẽ ít bị phụ thuộc bởi cường độ sáng, nhiễu, góc xoay của ảnh do các descriptor được tạo ra từ gradients do đó nó đã bất biến với các thay đổi về độ sáng (ví dụ: thêm 10 vào tất cả các pixel hình ảnh sẽ mang lại cùng một mô tả chính xác).
* Nhanh và hiệu quả, tốc độ xử lý gần như với thời gian thực (realtime)
* Có thể xử lý khi xoay ảnh