TRƯỜNG ĐẠI HỌC THĂNG LONG

BÁO CÁO

Thực tập

TRỊNH HOÀNG ĐĂNG
dangtr0408@gmail.com

Chuyên ngành : Trí Tuệ Nhân Tạo

Giảng viên hướng dẫn : TS. Nguyễn Thị Huyền Châu

Chữ ký GVHD

Bộ môn : Trí Tuệ Nhân Tạo

Hanoi, 9-2023

You can have data without information, but you cannot have information without data.

Daniel Keys Moran

Các Kỹ Thuật Xử Lý Hình Ảnh Trong Thị Giác Máy Tính

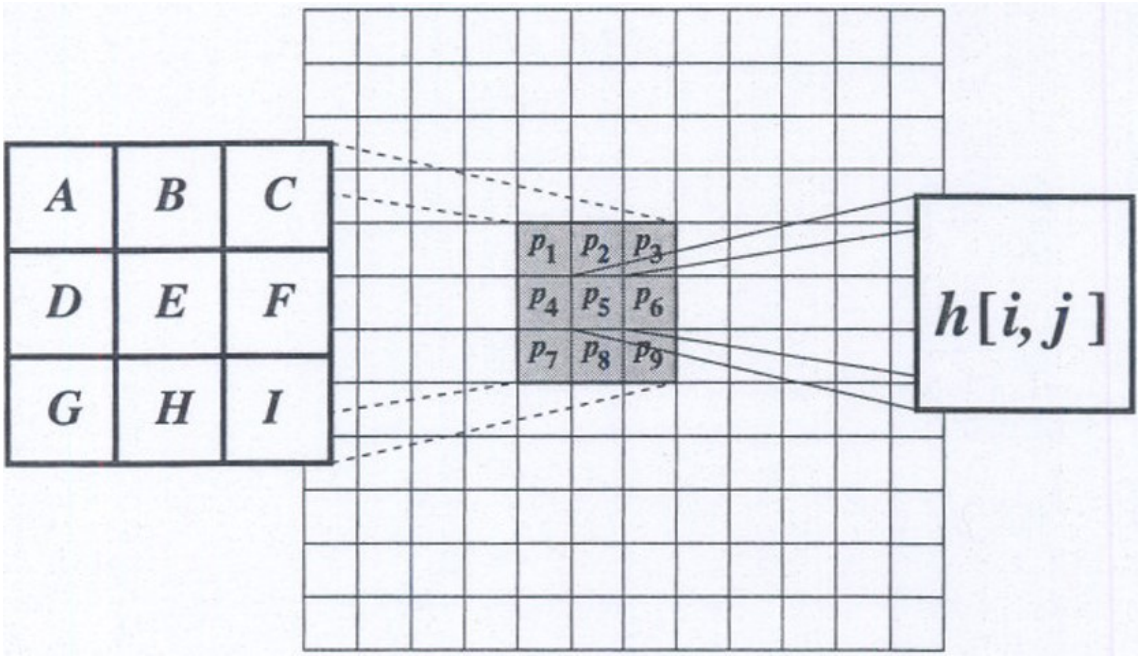
Image processing (xử lý ảnh) hay cụ thể hơn là image filtering (lọc ảnh) là các phương pháp biến đổi ảnh nói chung nhằm trích xuất thông tin và tăng cường chất lượng ảnh. Các phương pháp này được ứng dụng rộng rãi để giải quyết các bài toán thực tế như khôi phục, chụp chiếu y tế, cảm biến, phân vùng ảnh,...Đặc biệt là trong thị giác máy tính, xử lý ảnh là một quá trình không thể thiếu để người nghiên cứu có thể nắm bắt được các đặc tính cũng như xây dựng mô hình học máy trên tập dữ liệu mình đang làm việc. Bài báo cáo sau đây sẽ giới thiệu một số phương pháp xử lý ảnh thông dụng trong thị giác máy tính như mặt nạ tích chập, biến đổi Fourier và ứng dụng để làm mịn, xác định viền vật thể,...

1 Kernel (Mặt nạ tích chập)

Kernel hay còn gọi là convolution mask hoặc mặt nạ tích chập trong tiếng Việt, là một ma trận M x N chiều (thông thường là 3 x 3) nhỏ hơn ma trận ảnh gốc. Các ma trận này được sử dụng để gán các hiệu ứng vào ảnh, ví dụ như làm mờ, làm sắc, tách viền,...Cách hoạt động của kernel có thể hình dung như một cửa sổ trượt nằm trên ảnh gốc và trượt từ trái qua phải, trên xuống dưới đồng thời thực hiện phép toán tích chập với mỗi điểm tọa độ trung tâm nó trượt qua. Công thức tích chập giữa ma trận ảnh f(i,j) và kernel g(i,j) như sau:

$$h_{i,j} = f_{i,j} \cdot g_{i,j} = \sum_{k=1}^{n} \sum_{l=1}^{m} f_{k,l} \cdot g_{i-k,j-1}$$

Sau khi thực hiện phép toán tích chập xong, ảnh sẽ bị nhỏ đi do đặc tính của tích chập. Trừ khi ta muốn giảm chiều ma trận ảnh thì điều này là không tốt vì sẽ làm mất đi thông tin các cạnh của ảnh. Để khắc phục, ta có thể sử dụng các kỹ thuật padding, tức là thêm các giá trị vào biên của ma trận ảnh để duy trì kích thước ban đầu. Thông thường, các giá trị padding được chọn bằng 0 hoặc bằng giá trị biên của ma trận ảnh.



Hình 1.1: Ví dụ về một kernel 3 x 3 với trọng số là A, B,...I và đầu ra trả về một giá trị điểm ảnh (pixel) duy nhất là h[i,j] [6].

2 Smoothing (Làm mịn)

Trong xử lý ảnh, smoothing thường được dùng như một bước tiền xử lý để chuẩn bị cho các kỹ thuật khác như phát hiện cạnh, phân vùng ảnh, hay nhận dạng đối tượng. Đối với ảnh thô chưa qua xử lý, độ sáng của một pixel được đo sẽ là độ sáng thực cộng với độ sáng nhiễu. Nếu các pixels gần đó cũng có độ sáng thực tương đương thì ta có thể smoothing để giảm độ sáng nhiễu của ảnh [4]. Kỹ thuật smoothing cũng sử dụng các kernel với cách hoạt động tương tự như trên. Dưới đây là một mean filter (lọc trung bình).

$$M = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Một ví dụ đơn giản về smoothing sử dụng mean filter như sau. Ta có độ sáng thực của một vùng 4 điểm ảnh là [100, 100, 100, 100]. Độ sáng đo được sẽ là $100 + n_i$ với n_i là các biến nhiễu ngẫu nhiên. Nếu chúng ta áp dụng mean filter tính trung bình M pixels thì ta có $100 + \sum \frac{n_i}{M}$, giả sử n_i độc lập, có phân phối giống nhau và trung bình bằng 0 thì $\sum \frac{n_i}{M}$ cũng sẽ là một biến ngẫu nhiên, trung bình tại 0 nhưng phương sai lại nhỏ hơn n_i và vì vậy độ nhiễu sau khi chia trung bình cũng giảm đi. Tuy nhiên trên thực tế, các vùng pixel thường không có giá trị giống nhau, đặc biệt là tại các vùng ở viền, cạnh của vật thể, vì vậy khi lấy trung bình cũng sẽ đồng thời làm mất thông tin của các pixel và làm ảnh mờ đi. Thế nên ngoài cái tên smoothing, kỹ thuật này cũng có tên là blurring (làm mờ).

Ngoài mean filter, còn có nhiều phương pháp smoothing khác như median filter (bộ lọc trung vị), gaussian filter (bộ lọc gauss), bilateral filter (bộ lọc song phương),... Mỗi phương pháp có những ưu và nhược điểm riêng, và được áp dụng cho những loại nhiễu khác nhau (Hình 2.1).

Median filter là một biến thể của mean filter, trong đó giá trị mới của một pixel được tính bằng cách lấy trung vị (giá trị giữa) của các pixel trong vùng lân cận. Median filter có thể loại bỏ được những nhiễu kiểu muối tiêu (salt and pepper noise), tức những pixel có độ sáng rất cao hoặc rất thấp so với các pixel xung quanh. Median filter cũng ít làm mờ hơn mean filter, do không bị ảnh hưởng bởi các giá trị ngoại lai. Tuy nhiên, median filter cũng có thể làm mờ các cạnh và góc của ảnh, và khó xử lý được những nhiễu có phân phối đều.

Gaussian filter là một phương pháp smoothing dựa trên hàm gauss, một hàm toán học có dạng đường cong chuông. Hàm gauss có tính chất là giá trị càng xa trung tâm thì càng nhỏ, và tổng diện tích dưới đường cong bằng 1. Gaussian filter sử dụng một kernel có các giá trị theo hàm gauss để tính trung bình trọng số của các pixel lân cận. Pixel ở trung tâm sẽ có trọng số cao nhất, và pixel ở xa sẽ có trọng số thấp hơn. Gaussian filter có thể loại bỏ được những nhiễu có phân phối chuẩn, và giữ lại được các cạnh và góc của ảnh tốt hơn mean filter hay median filter. Gaussian filter còn được dùng để tạo ra hiệu ứng làm mờ theo chiều sâu (depth of field), khi muốn tập trung vào một vùng nào đó của ảnh.

3 Edge Detection (Xác định viền)

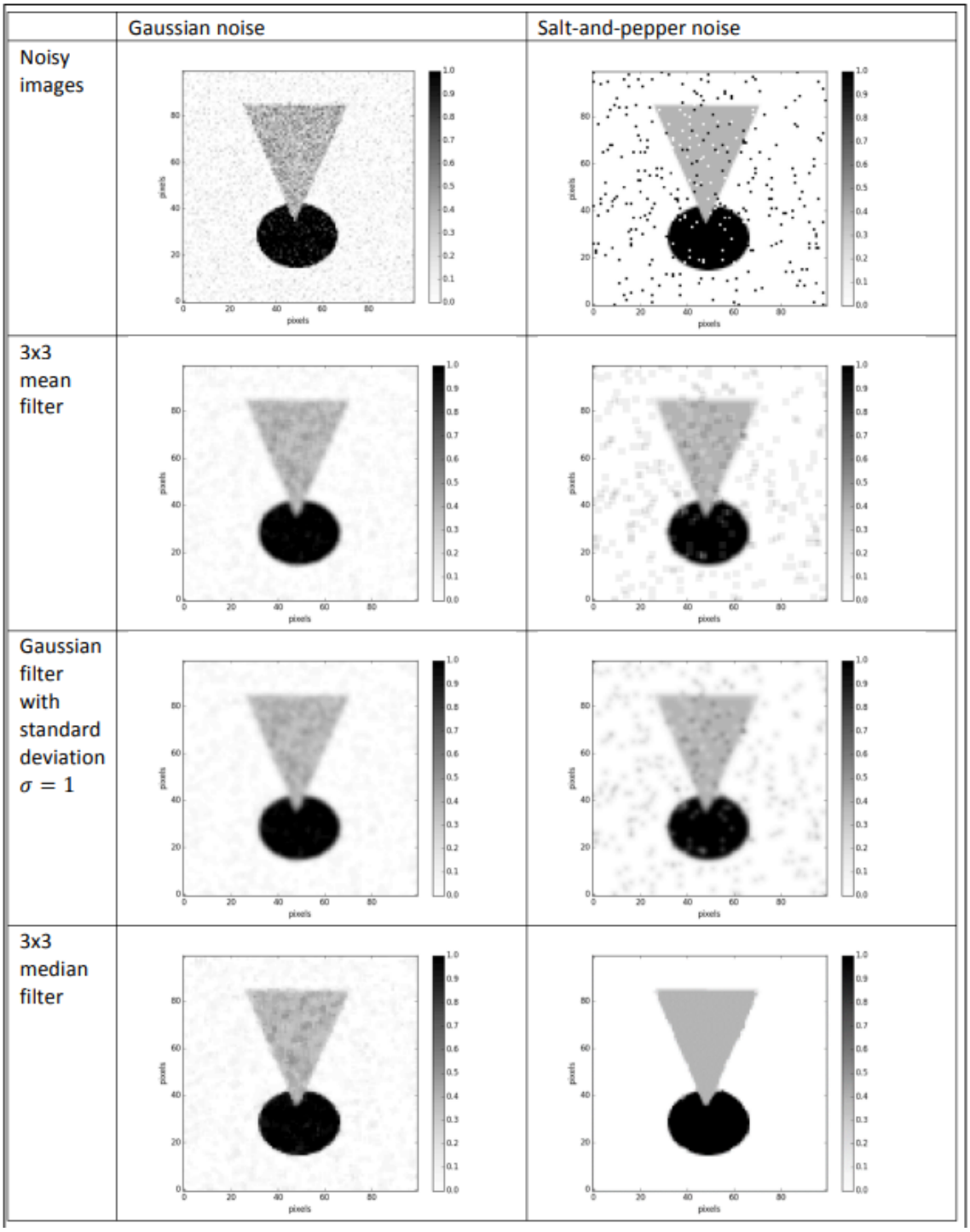
Một ứng dụng khác có sử dụng kernel đó là edge detection. Edge detection là kỹ thuật xác định các điểm biên giới giữa các vùng khác nhau trong ảnh bằng cách tính toán sự thay đổi đột ngột về độ sáng tối, ví dụ như biên giới giữa vật thể và nền, giữa hai vật thể khác nhau,... Edge detection có thể được sử dụng để nhận dạng khuôn mặt, đối tượng, giảm thông tin phải xử lý...

Có nhiều phương pháp phát hiện cạnh khác nhau, phần lớn có thể được nhóm vào hai danh mục: Gradient và Laplacian. Gradient tìm cực đại và cực tiểu từ đạo hàm bậc nhất của ảnh để xác định viền, trong khi đó Laplacian xác định các điểm zero crossings (điểm cắt trục hoành) của đạo hàm bậc hai (Hình 3.1) [2]. Tuy nhiên, ảnh là tổ hợp của các pixel rời rạc nên không thể đạo hàm được, do đó ta sẽ sử dụng các kernel để xấp xỉ đạo hàm. Dưới đây là ví dụ về kernel của hai thuật toán Sobel vào Prewitt, cả hai thuật toán đều dùng phương pháp gradient.

$$Sobel_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Sobel_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$Prewitt_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad Prewitt_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Các kernel này được áp dụng lên ma trận ảnh theo công thức tích chập đã nêu trên,



Hình 2.1: Hiệu ứng của filter mean, gauss và median trên ảnh có nhiễu. Có thể thấy mean, gauss filter lọc nhiễu gauss khá, trong khi đó median filter lọc nhiễu kiểu muối tiêu rất tốt. [3].

sau đó ta có thể kết hợp kết quả của hai kernel theo chiều x và y để thu được ma trận viền của ảnh. Công thức kết hợp có thể là:

$$E = \sqrt{(Sobel_x)^2 + (Sobel_y)^2}$$

hoặc

$$E = |Sobel_x| + |Sobel_y|$$

có thể tính được hướng của Gradient bằng công thức:

$$\theta = \arctan\left(\frac{Sobel_y}{Sobel_x}\right)$$

4 Fourier Transform (Biến đổi Fourier)

Fourier transform là một công cụ rất thông dụng trong xử lý ảnh. Nó có liên quan tới Fourier transform khá nổi tiếng trong xử lý, phân tích tín hiệu thành các dao động. Trong lĩnh vực xử lý ảnh, Fourier transform giúp ta phân tích ảnh thành các thành phần sin và cos. Sau khi biến đổi, kết quả (đầu ra) sẽ đại diện cho miền tần số của ảnh trong khi đó ảnh gốc (đầu vào) là miền không gian. Việc chuyển đổi này có nhiều ứng dụng trong xử lý ảnh, như lọc nhiễu, nén ảnh, phân tích tần số, nhận dạng hình ảnh,...

Do ảnh là một tập hợp hữu hạn các điểm ảnh, nên ta sử dụng công thức của Discrete Fourier Transform (DFT) để thực hiện biến đổi Fourier. Công thức DFT cho một ảnh $f(m, n)$ có kích thước $M \times N$ là:

$$F(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-i\,2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

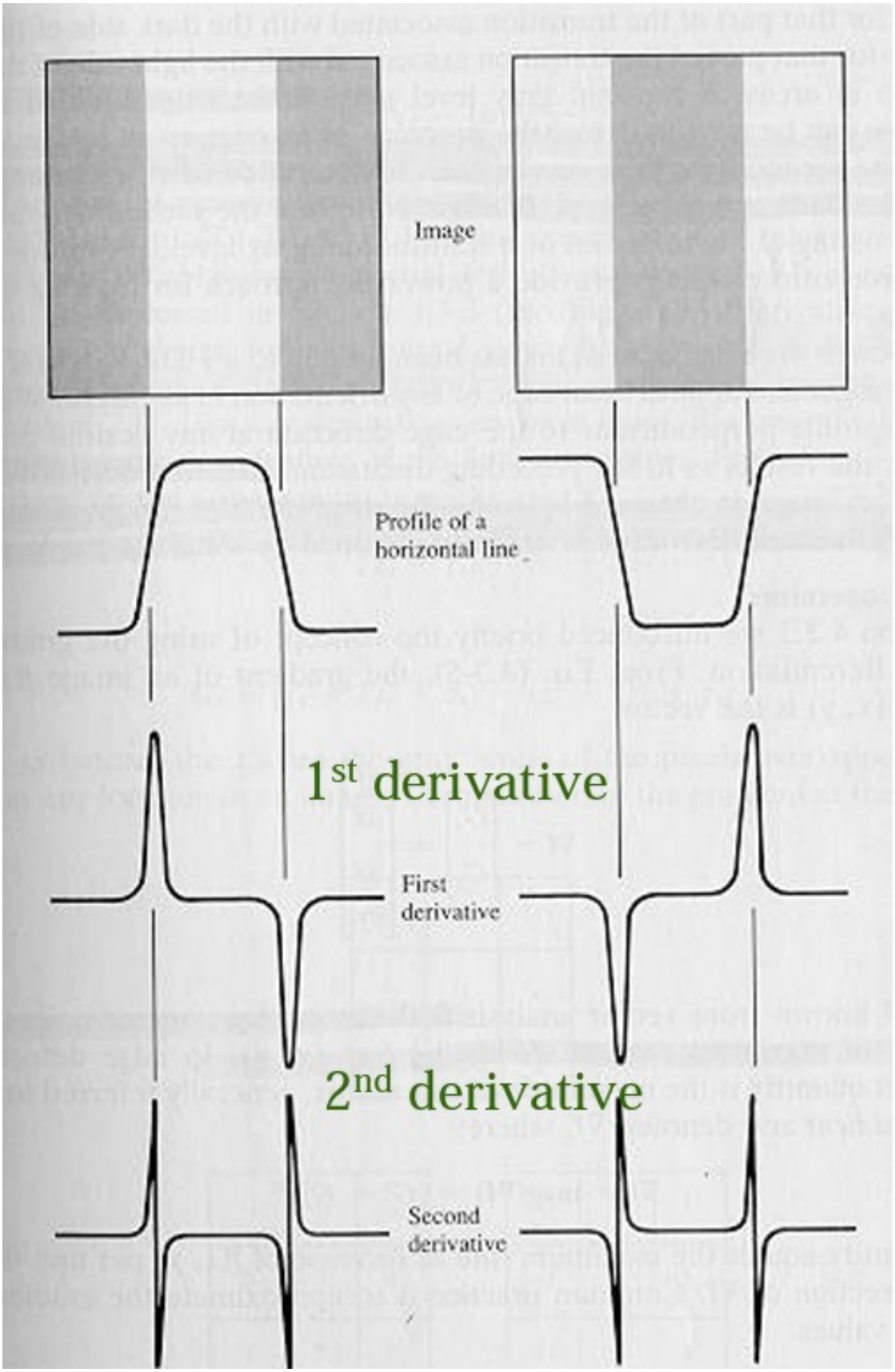
Trong đó $F(u, v)$ là biểu diễn miền tần số của ảnh $f(m, n)$. Để chuyển ngược từ miền tần số về miền không gian, ta sử dụng công thức Inverse Discrete Fourier Transform (IDFT):

$$f(m, n) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{+i\,2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

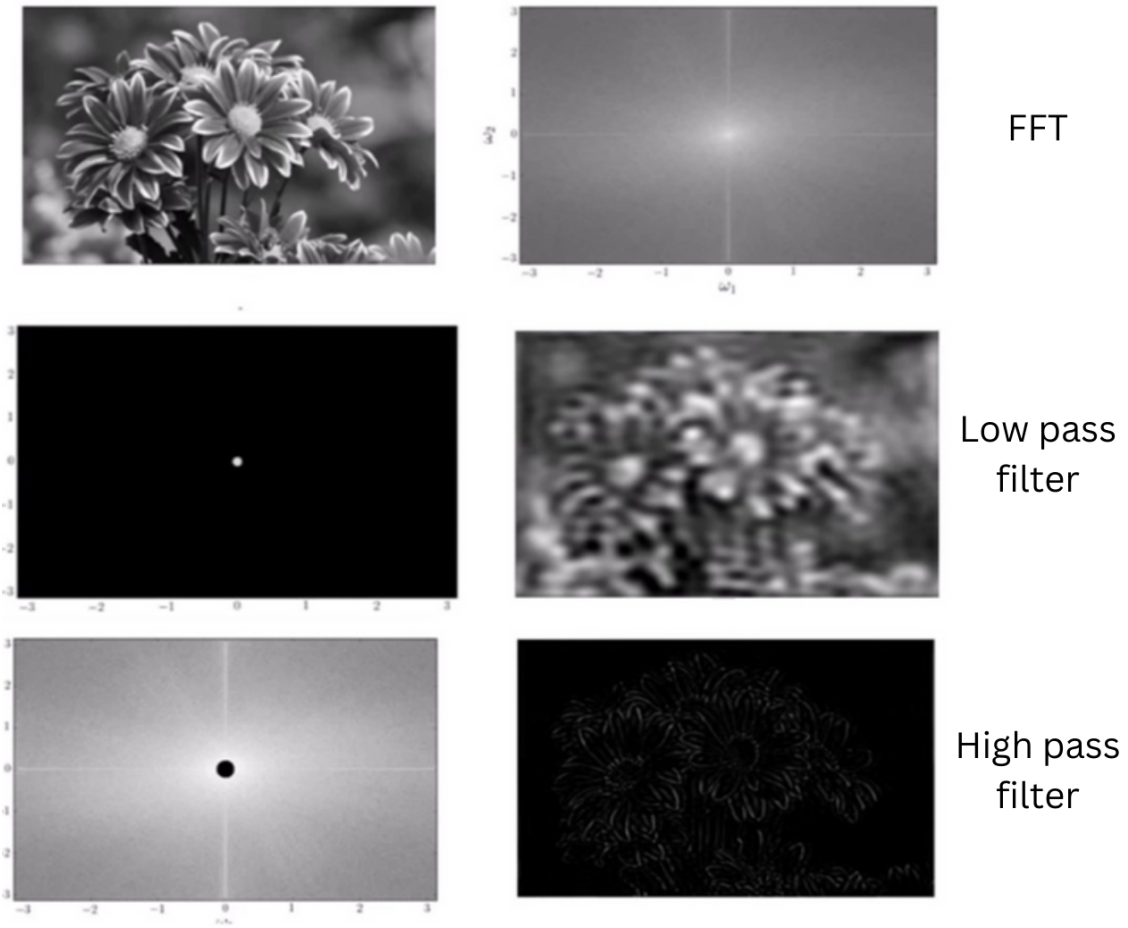
Một trong những ứng dụng quan trọng của Fourier transform trong xử lý ảnh là lọc nhiễu. Ta có thể sử dụng các bộ lọc (filter) để loại bỏ hoặc giữ lại các thành phần sóng có tần số cao hoặc thấp trong miền tần số của ảnh. Có hai loại bộ lọc chính là low pass filter và high pass filter. Low pass filter là bộ lọc chỉ cho qua các thành phần sóng có tần số thấp và loại bỏ các thành phần sóng có tần số cao. High pass filter là bộ lọc ngược lại, chỉ cho qua các thành phần sóng có tần số cao và loại bỏ các thành phần sóng có tần số thấp. Low pass filter có thể được sử dụng để làm mờ ảnh, giảm nhiễu muối tiêu (salt and pepper noise) trong khi đó high pass filter có thể được sử dụng để xác định viền, hoặc phát hiện các đặc trưng chi tiết của ảnh (Hình 4.1).

5 Phương Pháp Xác Định Cạnh Và Góc Kết Hợp

Đây là một phương pháp cải tiến từ thuật toán Moravec được đề ra trong tờ báo khoa học ”A COMBINED CORNER AND EDGE DETECTOR”[8] của hai tác giả Chris Harris và



Hình 3.1: Đồ thị đạo hàm của phương pháp gradient và zero crossings.



Hình 4.1: Ví dụ về low pass và high pass filter trong xử lý ảnh miền tần số [7].

Mike Stephens vào năm 1988. Nhắc lại một chút về Moravec, ý tưởng chính của phương pháp này là đo sự thay đổi của ảnh khi dịch chuyển một cửa sổ nhỏ trên ảnh theo các hướng khác nhau. Phương pháp Moravec có thể được thực hiện theo các bước sau:

- Chọn một cửa sổ nhỏ (ví dụ 3x3 pixel) và dịch chuyển nó theo bốn hướng: trái, phải, lên, xuống, với một khoảng cách nhỏ (ví dụ 1 pixel).
- Tính toán sự thay đổi của cửa sổ gốc và cửa sổ đã dịch chuyển theo công thức sau:

  $$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

  Trong đó, $E(u,v)$ là sự thay đổi khi dịch chuyển cửa sổ theo vector (u,v), (x,y) là tọa độ ban đầu,
- Lặp lại các bước trên cho tất cả các điểm ảnh trong ảnh và tạo ra một bản đồ độ nhạy của ảnh.

Sẽ có ba trường hợp được cân nhắc với mỗi điểm đo độ nhạy:

- Nếu vùng ảnh trong cửa sổ không phải là một góc hoặc một cạnh thì tất cả các dịch chuyển đều không thay đổi hoặc thay đổi một lượng nhỏ $E(u,v)$.
- Nếu cửa sổ gặp một cạnh thì các dịch chuyển theo chiều của cạnh gần như không thay đổi. Nhưng các dịch chuyển vuông góc sẽ có sự thay đổi lớn.
- Nếu gặp một góc, dịch chuyển theo cả bốn hướng đều sẽ tạo sự thay đổi lớn cho $E(u,v)$.

Tuy nhiên, theo như tờ báo khoa học trên, thuật toán Moravec gặp phải một số vấn đề cần phải khắc phục. Thứ nhất, thuật toán chỉ xem xét duy nhất 4 hướng rời rạc. Thứ hai, thuật toán dễ bị nhiễu làm sai lệch vì sử dụng cửa sổ nhị phân. Cuối cùng, thuật toán quá nhạy với cạnh vì chỉ một phần nhỏ E được cân nhắc. Vì những lý do này, Harris và Stephens đã đề xuất một thuật toán cải tiến để khắc phục những hạn chế trên.

Để tránh bị ảnh hưởng bởi nhiễu, cửa sổ hình chữ nhật w của thuật toán Moravec có thể thay thế bằng một cửa sổ Gaussian. Tiếp đó, để mở rộng bộ phát hiện Moravec ra tất cả các hướng, không giới hạn ở bốn hướng ban đầu, tác giả khai triển chuỗi Taylor trên các cửa sổ con đã dịch chuyển $I(x+u,y+v)$:

$$I(x+u,y+v) \approx I(x,y) + u\partial_u I(x,y) + v\partial_v I(x,y)$$

Thay xấp xỉ trên vào công thức Moravec gốc ta thu được:

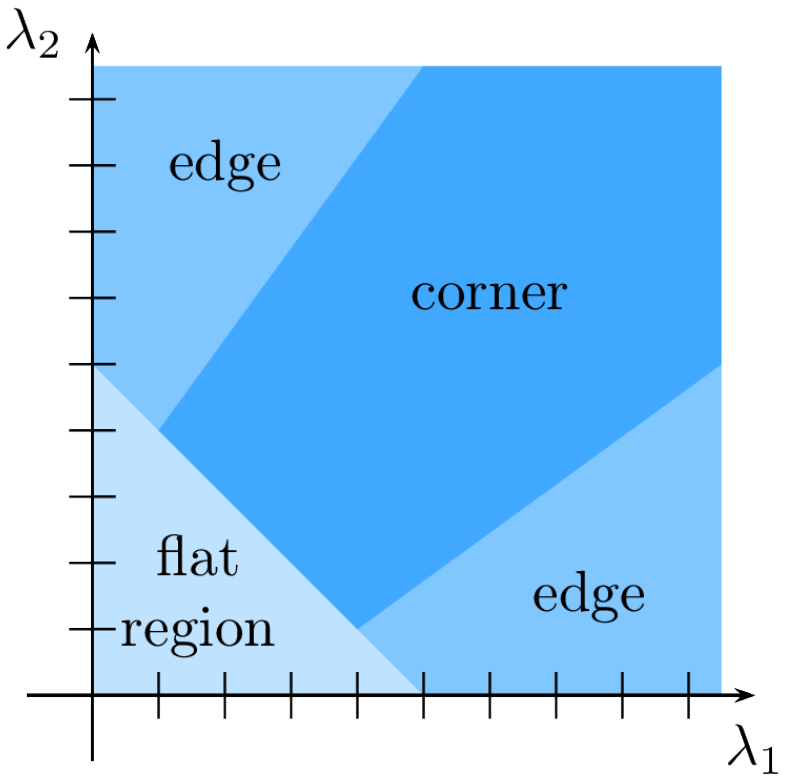
$$E(u,v) \approx \sum_{x,y} w(x,y)[u\partial_u I(x,y) + v\partial_v I(x,y)]^2$$

Công thức này cũng có thể được biểu diễn dưới dạng ma trận:

$$E(u,v) \approx \begin{pmatrix} u & v \end{pmatrix} M \begin{pmatrix} u \\ v \end{pmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{pmatrix} (\partial_u f)^2 & \partial_u f\, \partial_v f \\ \partial_u f\, \partial_v f & (\partial_v f)^2 \end{pmatrix}$$

Và cuối cùng, để khắc phục thuật toán quá nhạy với các cạnh, tác giả đề xuất một phương thức mới để xác định cạnh và góc. Thay vì sử dụng ba tiêu chí để xác định góc cạnh như



Hình 5.1: Sử dụng các giá trị riêng của ma trận M để xác định góc, cạnh [5].

ban đầu, ta có thể phân tích các giá trị riêng (eigenvalues) của ma trận M để xác định ba trường hợp góc, cạnh và phẳng (Hình 5.1). Nhưng do tính giá trị riêng có thể phức tạp nên thông thường ta sẽ sử dụng công thức sử dụng det và trace sau thay thế, giúp tính toán đơn giản và nhanh hơn:

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k(\text{trace}(M))^2$$

$$0.04 < k < 0.06$$

Theo đó, nếu R mang giá trị nhỏ, ta có thể kết luận là một vùng phẳng. Nếu R mang giá trị dương, đó sẽ là một góc. Và ngược lại, R mang giá trị âm là một cạnh.

Dựa trên những cải tiến mà Harris và Stephens đã đề xuất, thuật toán Moravec đã được cải thiện đáng kể. Cải tiến này đã được chứng minh là có hiệu quả và ổn định trong nhiều ứng dụng thực tế ngày nay.

6 Xác Định Blob (Blob Detection)

Trong thị giác máy tính, blob detection là một phương pháp xác định các vùng ảnh khác biệt về đặc tính như là độ sáng, màu so với các vùng xung quanh. Thông thường, các điểm trong một blob thường giống nhau [1]. Blob dectection có rất nhiều ứng dụng thực tế ví dụ như nhận diện, theo dõi và đếm số lượng vật thể, phát hiện các điểm bất thường trong ảnh, thường sử dụng trong ảnh chụp y tế (X-ray, MRI).

Sau khi khử nhiễu và xác định màu ảnh, blob detection thường là bước cuối cùng trong chuẩn bị dữ liệu ảnh cho xây dựng mô hình. Phương pháp phổ biến để thực hiện blob detection đó là sử dụng bộ lọc Laplacian of Gaussian (LoG). Công thức của bộ lọc LoG như sau:

$$LoG = \frac{-1}{\pi\sigma^4}(1 - \frac{x^2+y^2}{2\sigma^2}) \exp -\frac{x^2+y^2}{2\sigma}$$

Bộ lọc LoG có thể phát hiện các vùng có sự thay đổi đột ngột về độ sáng hoặc màu sắc trong ảnh. Đây chính là các vùng có khả năng là blob. Tuy nhiên, tính toán hàm LoG có thể phức tạp và tốn nhiều tài nguyên tính toán nên ta có thể dùng hàm Difference of Gaussian (DoG) để xấp xỉ hàm LoG. Công thức của hàm DoG có thể diễn giải như sau:

$$G_\sigma = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2+y^2}{2\sigma^2}$$

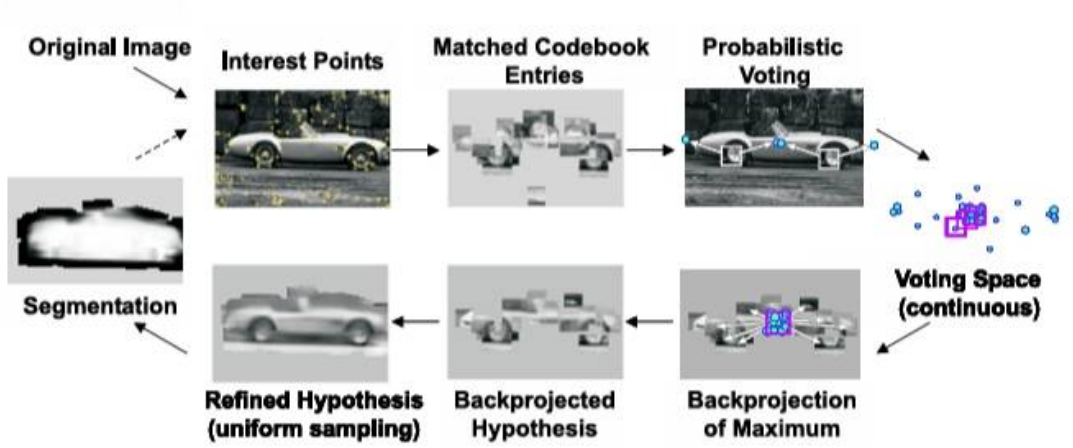
$$LoG \approx G_{1.6\sigma} - G_\sigma$$

Trong thư viện OpenCV - một thư viện nổi tiếng trong xử lý ảnh, đã được tích hợp sẵn hàm blob detection sử dụng bộ lọc LoG và DoG để thuận tiện trong việc ứng dụng. Sau đây là một đoạn code python ngắn triển khai hàm LoG trong OpenCV.

```
import cv2
import numpy as np

image = cv2.imread('blob.jpg', cv2.IMREAD_GRAYSCALE)

blobs_log = cv2.Laplacian(image, cv2.CV_64F)
blobs_log = np.uint8(np.absolute(blobs_log))

params = cv2.SimpleBlobDetector_Params()
params.minThreshold = 10
params.maxThreshold = 200
params.filterByArea = True
params.minArea = 100
params.filterByCircularity = True
params.minCircularity = 0.9
params.filterByConvexity = True
params.minConvexity = 0.2
params.filterByInertia = True
params.minInertiaRatio = å0.m

detector = cv2.SimpleBlobDetector_create(params)
keypoints = detector.detect(blobs_log)

im_with_keypoints = cv2.drawKeypoints(image, keypoints, np.array([]),

cv2.imshow("Keypoints", im_with_keypoints)
```

7 Phương Pháp Kết Hợp Phân Loại Và Phân Vùng

Phương pháp này lần đầu được giới thiệu trong tờ báo khoa học ”Combined Object Categorization and Segmentation with an Implicit Shape Model”của Bastian Leibe et al. Giới thiệu một phương pháp phân loại và phân vùng đối tượng dựa trên Mô hình Ẩn (Implicit Shape Model). Mô hình này bao gồm một sổ mã hay còn gọi là codebook cho các lớp đối tượng, trong đó chứa các hình dạng đặc trưng của lớp đó. Và một phân phối xác suất không gian, chỉ ra vị trí có thể xuất hiện của mỗi đặc trưng trên đối tượng. Phân phối xác suất phải thỏa mãn hai yêu cầu: Thứ nhất, phân phối được xác định độc lập cho mỗi đặc trưng, điều đó giúp cho phương pháp linh hoạt hơn, vì nó cho phép kết hợp các bộ phận đối tượng được quan sát trên các ví dụ huấn luyện khác nhau. Thứ hai là phân phối xác suất cho mỗi hình dạng cục bộ được ước lượng theo cách phi tham số, cho phép mô hình hóa phân phối thực tế một cách chi tiết nhất có thể từ dữ liệu huấn luyện, thay vì giả định phân phối Gauss đơn giản.



Hình 7.1: Quá trình ảnh được xử lí bởi Implicit Shape Model.

Ta có thể hiểu ngắn gọn các bước như sau:

- Hình gốc mới (Original Image): Đây là điểm khởi đầu, nơi chúng ta có một hình ảnh của một chiếc xe. Codebook đã được train trước đó bằng cách phân ảnh thành các patches nhỏ sau đó cluster những patch giống nhau, tính điểm trung tâm mỗi patch và lưu vào codebook.
- Điểm quan tâm (Interest Points): Trong bước này, các điểm quan trọng hoặc đặc trưng trong hình ảnh gốc được xác định. Các điểm có thể là góc, cạnh hoặc các cấu trúc khác biệt khác trong hình ảnh. Trong tờ báo các tác giả đã sử dụng thuật toán Harris đã giải thích ở phía trên để xác định các điểm này.
- Khớp sổ mã (Matched Codebook Entries): Các điểm quan tâm sau đó được so sánh với các điểm trong codebook bằng thuật toán NGC để trích ra các đặc trưng và xác định vật thể.
- Bỏ phiếu (Probabilistic Voting): Mỗi điểm khớp với codebook sẽ bỏ phiếu bằng công thức (6) mà tác giả đề ra để tìm ra vị trí trung tâm của vật thể.

- Không gian bỏ phiếu (Voting Space): Tất cả các phiếu bầu được tích lũy trong một không gian bỏ phiếu liên tục, nơi các cụm phiếu bầu chỉ ra vị trí và sự hiện diện tiềm năng của đối tượng.
- Phản chiếu cực đại (Backprojection of Maximum): Giả thuyết có số phiếu cao nhất được phản chiếu lại ảnh gốc để xác minh và tinh chỉnh.
- Giả thuyết phản chiếu (Backprojected Hypothesis): Hiển thị các giả thuyết tinh chỉnh sau khi phản chiếu lại, cung cấp vị trí đối tượng chính xác hơn.
- Giả thuyết tinh chỉnh (Refined Hypothesis): Tinh chỉnh thêm bằng cách sử dụng lấy mẫu đồng nhất để loại bỏ ngoại lệ và cải thiện độ chính xác.
- Phân vùng vật thể (Segmentation): Cuối cùng, thực hiện phân vùng để tách rõ đối tượng khỏi nền của nó.

Tài liệu tham khảo

[1] Blob detection. https://en.wikipedia.org/wiki/Blob_detection.

[2] Patrice Delmas. Edge detection, 2016. The University of Auckland.

[3] Ron Dror. Image analysis, 2015. Stanford University.

[4] David W. Jacobs. Smoothing and convolution, 2016. University of Maryland.

[5] Vincent Mazet. Corner detection. https://vincmazet.github.io/bip/detection/corners.html, 2020.

[6] Brian G. Schunck Ramesh Jain, Rangachar Kasturi. *MACHINE VISION*. McGraw-Hill, 1995.

[7] Sidd Singal. Image processing with fourier transform. https://github.com/siddthesquid/ImageTransformer, 2016.

[8] Chris Harris Mike Stephens. A combined corner and edge detector. 1988.