BordesContornos

Tomas Lopez Pérez

June 2024

Contents

1 Introduction

El procesamiento de imágenes es un campo fundamental en la visión por computadora y en numerosas aplicaciones tecnológicas modernas. Uno de los aspectos más críticos de este campo es la detección de bordes y contornos, ya que estos elementos proporcionan información esencial sobre la forma, estructura y características de los objetos presentes en una imagen. Los bordes representan transiciones bruscas en la intensidad de la imagen, mientras que los contornos delinean los límites de los objetos, facilitando su identificación y análisis.

La detección de bordes y contornos es una etapa preliminar crucial en diversos procesos como el reconocimiento de patrones, la segmentación de imágenes, la reconstrucción tridimensional y la navegación autónoma de robots. A través de técnicas como el filtrado, la convolución, y algoritmos específicos como Canny, Sobel y Laplaciano, es posible extraer estas características de manera precisa y eficiente.

Este reporte tiene como objetivo la implementacion diferentes métodos de detección de bordes y contornos.

2 Fundamento teórico

2.1 Procesamiento de imágenes

El Procesamiento de imágenes es un campo de investigación muy extenso que involucra diversas áreas del conocimiento, ya que involucra diversos procesos, tales como: la adquisición, transmisión, representación y procesamiento. En términos generales el procesamiento de imágenes se utiliza para modificar una imagen con el fin de mejorar la apariencia visual para un observador y para resaltar convenientemente el contenido de la imagen de cara a la percepción por parte de máquinas, es decir, se hace una manipulación de imágenes con objeto de producir nuevas imágenes que son mejores, en algún sentido. El procesamiento de imágenes comprende distintas técnicas que se utilizan para mejorar, restaurar e inclusive comprimir imágenes ya existentes; la implementación y desarrollo de dichas técnicas agregadas a sistemas ya existentes nos permiten editar imágenes de forma más sencilla y precisa [**?**]. La Figura 1 muestra el flujo de las etapas del procesamiento digital de imágenes propuestas por Gonzalez en donde se establece como la etapa inicial la adquisición de la imagen digital, lo que se puede hacer mediante algún sistema de sensores que permitan digitalizar la imagen u obtener las imágenes de algún repositorio o colección. El conjunto de imágenes adquiridas conforma inicialmente la base de conocimiento, ya que para algunas aplicaciones basta con tener la información pura, sin embargo en la mayoría de las aplicaciones del procesamiento de imágenes, es necesario enriquecer la base del conocimiento con al menos las etapas de procesado y segmentación, lo cual requiere aplicar una serie de operadores que transformen las imágenes para resaltar características relevantes, en el caso del procesado, o separar los diferentes elementos que componen la imagen, en el caso de la segmentación, de modo que los resultados de estas dos etapas permiten tener una base del conocimiento más amplia y robusta. Respecto a la etapa de representación y descripción. las imágenes procesadas y/o segmentadas son usadas para extraer características más especificas sobre la imagen, tales como cantidad, dimensión, forma, posición, etc. de los objetos de interés en las imágenes, esta información también puede formar parte de la base del conocimiento, de modo que en la ultima etapa, reconocimiento e interpretación, se puedan obtener respuestas a partir de las imágenes y la información generada por estas.

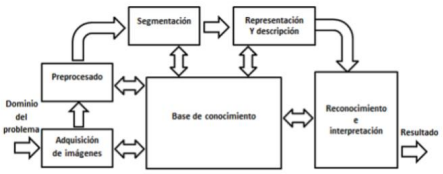

Figure 1: Etapas fundamentales del procesamiento digital de imágenes.

2.2 Convolución

La convolución es una operación de filtrado digital de señales, que combina dos señales correspondientes a la imagen original $f(t)$ y una función filtro $g(t)$. La convolución es una operación que representa la magnitud en que se superpone una función $g(t)$ a medida que se desplaza sobre la función $f(t)$, y se define mediante una tercera función $f_c(t)$ de modo que para una señal en términos de una variable se tiene que

$$f_c(t) = f(t) * g(t) = \int_{-\infty}^{\infty} f(t)g(t-x)dx \tag{1}$$

Lo anterior, trasladado al procesamiento de imágenes, donde tenemos que la imagen de entrada es finita, discreta y representada por la función de intensidades en termino de dos variables$f(i,j)$ y la función de filtrado $w(m,n)$ dada por la matriz:

$$w = \begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n-1} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n-1} \\ \vdots & \vdots & & \vdots \\ w_{m-1,0} & w_{m-1,1} & \dots & w_{m-1,n-1} \end{bmatrix} \tag{2}$$

tenemos que la convolución discreta se define como:

$$f_c(i,j) = w(m.n) * f(i,j) = \sum_{m=-k}^{k} \sum_{n=-p}^{p} w(m.n) f(i-m, j-n) \tag{3}$$

donde w es la matriz deslizante de dimensiones $(2k+1) \times (2p+1)$, también llamada kernel o máscara de convolución, y las variables i y j corresponde al renglón y columna de la imagen original [**?**].

la Figura 2 tomada de [**?**] ejemplifica la operación de convolución aplicada a una ventana de una imagen y una máscara w de 3×3.

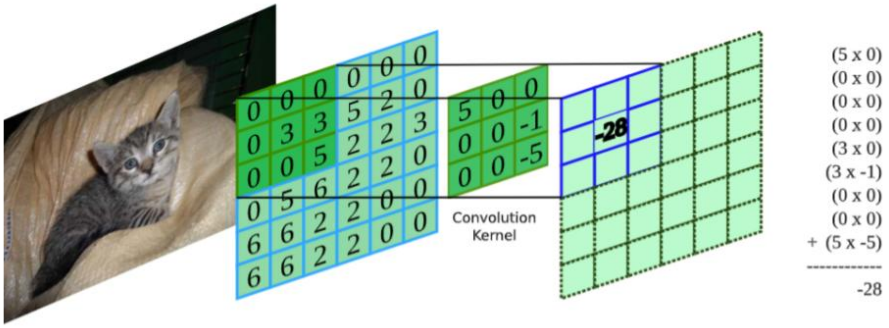

Figure 2: Ejemplo de la operación de convolución.

3 Metodologia

La metodologia propuesta en este trabajo constituye en la implementacion de algoritmos que nos permite la deteccion de bordes, Sobel, Prewitt, Kirsch, Canny, por mencionar algunos.

3.1 Encontrar las derivadas en X y Y utilizando convolucion

Para sobel las matrices H_{sx} y H_{sy} son las siguientes:

$$H_{sx} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$H_{sy} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Para prewitt las matrices H_{sx} y H_{sy} son las siguientes:

$$H_{sx} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H_{sy} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Para Kirsch se necesitan 8 matrices

$$Hk0 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Hk4 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$Hk1 = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \qquad Hk5 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$Hk2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad Hk6 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$Hk3 = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \qquad Hk7 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

3.2 Obtener la magnitud del gradiente

Para obtener la magnitud del gradiante se necesita aplica la formular

$$E(u,v) = \sqrt{\big(D_x(u,v)\big)^2 + \big(D_y(u,v)\big)^2}$$

Figure 3: Magnitud del gradiente

Para la implementación de Canny se requiere utilizar la pendiente θ del gradiente

$$\Phi(u,v) = \tan^{-1}\Big(\frac{D_y(u,v)}{D_x(u,v)}\Big) = \mathrm{ArcTan}\big(D_x(u,v), D_y(u,v)\big)$$

Figure 4: Pendiente θ del gradiente

4 Codigos

Para el algoritmo de Sobel 5, 6 , 7, se realizo los mismos pasos, Utilizar dos matrices Hsk y Hsy para encontrar las derivadas en los ejes X y Y, por medio de la convolucion y luego aplicando 4 podemos obtener los bordes de la imagen

Para la implementacion de kirsch se necesitan 8 matrices se realiza la convolucion 2d para cada una de las imagenes y para obtener la imagen final se selecciona la derivada Maxima y con eso podemos obtener la imagen con los bordes.

5 Resultado

6 Conclusion

la detección de bordes en el procesamiento de imágenes es un área crucial que desempeña un papel fundamental en numerosas aplicaciones, desde la visión por computadora hasta el análisis de imágenes médicas. Los operadores clásicos como Sobel, Prewitt, Kirsch y Canny han sido ampliamente estudiados y utilizados debido a su efectividad y versatilidad en la detección de discontinuidades en la intensidad de los píxeles. La elección del operador adecuado depende de diversos factores, como el tipo de imagen, el nivel de ruido, la precisión requerida y los recursos computacionales disponibles. En última instancia, la implementación y comprensión de estos operadores son fundamentales para desarrollar sistemas de visión por computadora y análisis de imágenes más avanzados y eficaces.

```
imgGray = double(imread("lena_gray_256.tif"));
%Sobel
Hsx = [-1 0 1; -2 0 2; -1 0 1];
Hsy = [-1 -2 -1; 0 0 0; 1 2 1];
% 1 es 3x3
% 2 es 5x5
nneighbors = 1;
derivateX = convolucion2D(imgGray,nneighbors,Hsx);
derivateX = abs(derivateX);
derivateY = convolucion2D(imgGray,nneighbors,Hsy);
derivateY = abs(derivateY);

gradient_magnitude = sqrt( derivateX .^ 2 + derivateY .^ 2);
gradient_magnitude = (gradient_magnitude * 255) /max(max(gradient_magnitude));

subplot(1, 4, 1), imshow(uint8(imgGray)), title('Original Image');
subplot(1, 4, 2), imshow(derivateX, []), title('Sobel X');
subplot(1, 4, 3), imshow(derivateY, []), title('Sobel Y');
subplot(1, 4, 4), imshow(uint8(gradient_magnitude), []), title('E(u,v)');
```

Figure 5: Sobel

```
imgGray = double(imread("lena_gray_256.tif"));
%Prewitt
Hsy = [-1 0 1; -1 0 1; -1 0 1];
Hsx = [-1 -1 -1; 0 0 0; 1 1 1];
% 1 es 3x3
% 2 es 5x5
nneighbors = 1;
derivateX = convolucion2D(imgGray,nneighbors,Hsx);
derivateX = abs(derivateX);

derivateY = convolucion2D(imgGray,nneighbors,Hsy);
derivateY = abs(derivateY);

gradient_magnitude = sqrt( derivateX .^ 2 + derivateY .^ 2);
gradient_magnitude = (gradient_magnitude * 255) /max(max(gradient_magnitude));

subplot(1, 4, 1), imshow(uint8(imgGray)), title('Original Image');
subplot(1, 4, 2), imshow(uint8(derivateX), []), title('Prewitt X');
subplot(1, 4, 3), imshow(uint8(derivateY), []), title('Prewitt Y');
subplot(1, 4, 4), imshow(uint8(gradient_magnitude), []), title('E(u,v)');
```

Figure 6: Prewitt

```
imgGray = imread("lena_gray_256.tif");

%Noise reduction
size = 5;
sigma = 1;
HG = gaussianKernel(size, sigma);
nneighbors = 2;
imgNR = convolucion2D(imgGray,nneighbors,HG);


%Gradient calculation
Hsx = [-1 0 1; -2 0 2; -1 0 1];
Hsy = [-1 -2 -1; 0 0 0; 1 2 1];

nneighbors = 1;
derivateX = convolucion2D(imgNR,nneighbors,Hsx);
derivateX = abs(derivateX);
derivateY = convolucion2D(imgNR,nneighbors,Hsy);
derivateY = abs(derivateY);

gradient_magnitude = sqrt( derivateX .^ 2 + derivateY .^ 2);
gradient_magnitude = uint8((gradient_magnitude * 255) /max(max(gradient_magnitude)));

theta = atan2(double(derivateY), double(derivateX));

E2 = NMSuppression(gradient_magnitude,theta);
res = Doublethreshold(E2);
```

Figure 7: Canny

```
imgGray = double(imread("lena_gray_256.tif"));

Hk0 = [-1 0 1; -2 0 2; -1 0 1];
Hk1 = [-2 -1 0; -1 0 1; 0 1 2];
Hk2 = [-1 -2 -1; 0 0 0; 1 2 1];
Hk3 = [0 -1 -2; 1 0 -1; 2 1 0];

Hk4 = [1 0 -1; 2 0 -2; 1 0 -1];
Hk5 = [2 1 0; 1 0 -1; 0 -1 -2];
Hk6 = [1 2 1; 0 0 0; -1 -2 -1];
Hk7 = [0 1 2; -1 0 1; -2 -1 0];

nneighbors = 1;
HkFull = {Hk0, Hk1, Hk2, Hk3};
Derivate = cell(1, 8);
maginitud = cell(1 ,4);

for i = 1:4
    derivate = convolucion2D(imgGray,nneighbors,HkFull{i});
    Derivate{i} = derivate;
    Derivate{i+4} = -derivate;
end

[m, n] = size(imgGray);
gradient_magnitude = zeros(m, n);
for i = 1:8
    gradient_magnitude = max(gradient_magnitude, abs(Derivate{i}));
end
gradient_magnitude = (gradient_magnitude * 255) /max(max(gradient_magnitude));

figure;
    subplot(3, 4, 1);
    imshow(uint8(imgGray));
    title('Imagen Original');
    for i = 1:8
        subplot(3, 4, i+1);
        imshow(uint8(Derivate{i}));
        title(['D ', num2str(i)]);
    end
    subplot(3, 4, 10);
    imshow(uint8(gradient_magnitude));
    title('Imagen Original');
```

Figure 8: kirsch

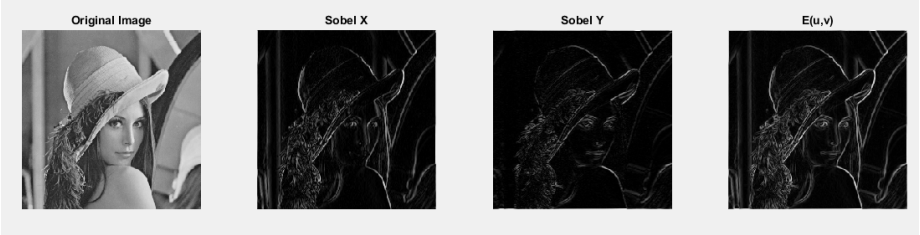

Figure 9: Resultado de Sobel

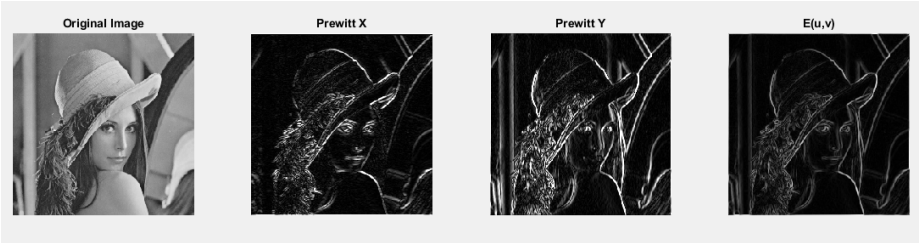

Figure 10: Resultado de prewitt

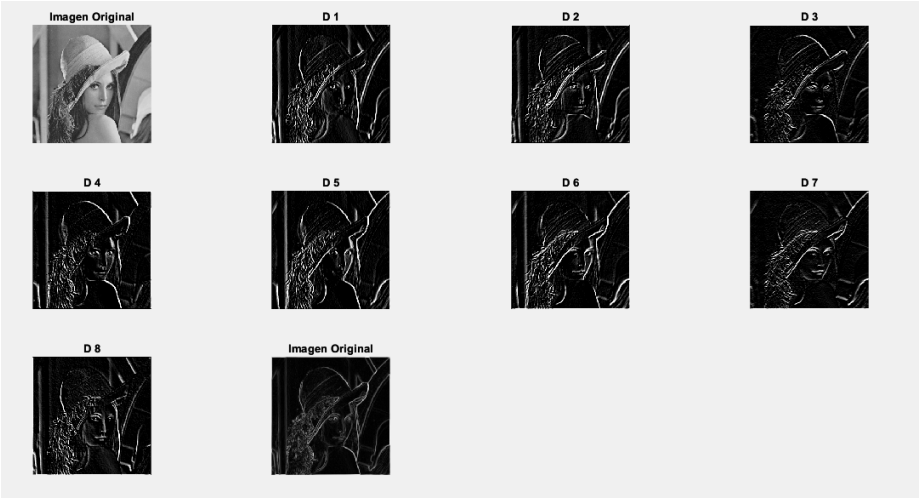

Figure 11: Resultado de kirsh

Figure 12: Resultado de Canny