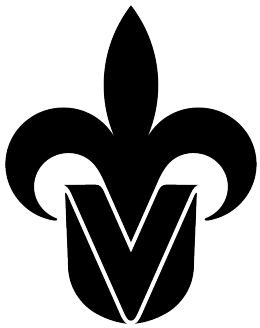


Universidad Veracruzana

Maestría en Inteligencia Artificial
Visión por computadora

Tarea 4
Obtención del gradiente de una imagen utilizando el operador de Sobel

Ulises Jiménez Guerrero

20 de marzo de 2025

Reporte tarea 4

1. Objetivos

- Desarrollar un programa capaz de obtener los vectores gradiente utilizando el operador de Sobel sobre una imagen.
- Mostrar los componentes G_X, G_y, ∇f y θ.
- Mostrar los vectores (magnitud y dirección) como flechas perpendiculares a la superficie (utilizar el comando `quiver(Gx,Gy)` de Matlab).

2. Metodología

2.1. Materiales utilizados

Se proporcionó la imagen 1 para realizar pruebas sobre el sistema diseñado, obtenida de [1] . Esta representa a un lente de contacto, iluminado de tal forma que se realcen los defectos. Estos son más notables en la parte inferior derecha del lente.

Para la codificación del algoritmo, se decidió utilizar Matlab, versión R2024b, para uso académico. Aunque se tienen instaladas las *toolbox* de visión por computadora, no se requirió de su uso para esta actividad. Se decidió utilizar este lenguaje debido a su rapidez a la hora de manejar matrices y operaciones sobre ellas, además de para simplificar el uso de `quiver(Gx,Gy)`.

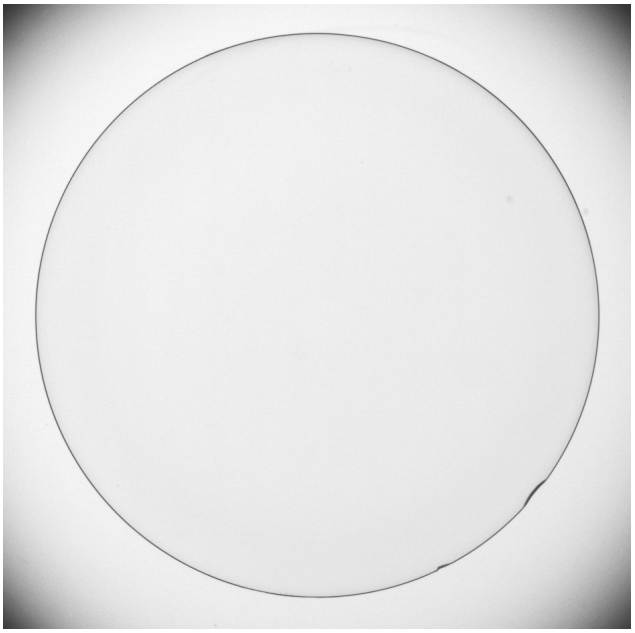


Figura 1: Figura 3.45(a), donde se probará el algoritmo implementado. Nótese las imperfecciones mencionadas.

2.2. Algoritmos e implementación

El primer paso para resolver el problema es desarrollar un algoritmo que aplique la convolución a una imagen dada con un kernel especificado. Para ello, se siguió la definición dada en clase, siendo esta

$$g(x,y) = f * K = \sum_{u=-h}^{h} \sum_{v=-h}^{h} f(x-u, y-v)K(u,v).$$

En esta fórmula, $f(x,y)$ indica la intensidad del pixel ubicado en la posición (x,y) y K indica el kernel a aplicar en la imagen. En nuestro caso, estos son los **operadores de Sobel**, aproximaciones a las derivadas parciales en x e y en la imagen. Estos están definidos como:

$$g_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad g_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Notemos que para aplicar un kernel de convolución sobre toda una imagen, cada pixel debe alinearse con el centro del kernel. Por lo tanto, los bordes no se pueden evaluar de manera directa, dado que se saldrían del rango delimitado por la imagen. Para lidiar con esto, se recurrió al uso de *padding,* agregando filas y columnas de ceros alrededor de la imagen. De esta forma, se pueden evaluar todos los pixeles sin perder información y sin alterar los resultados.

La implementación se realizó de manera directa y sencilla en Matlab. Se itera sobre cada pixel en la imagen original, aplicando la operación matricial adecuada, y se guardan los resultados en los píxeles correspondientes a las imágenes resultantes para el componente x e y del gradiente en cada píxel.

Para encontrar la magnitud y la dirección del vector gradiente, se utilizaron las fórmulas

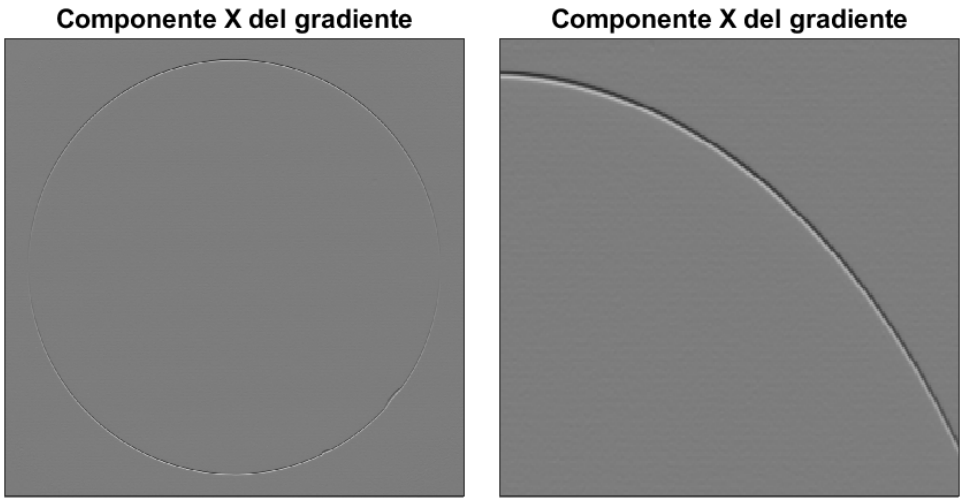
$$||\nabla f|| = \sqrt{g_x^2 + g_y^2}, \quad \theta = \arctan\left(\frac{g_y}{g_x}\right).$$

Estas operaciones se pudieron aplicar de manera eficaz en Matlab, utilizando operaciones punto a punto sobre las matrices necesarias. Finalmente, para la muestra de los vectores gradiente como flechas se siguió la recomendación y se aplicó la función de Matlab `quiver(g_x, g_y)`.

Cabe recordar que todos los operadores de derivada, ya sea en primer o segundo orden, nos dan imágenes más nítidas, con más detalles en las zonas de transición de intensidad. También destacan más el ruido en la imagen, y se pueden utilizar para el resaltado de bordes. Por lo tanto, el resultado esperado para la imagen de prueba es que se resalte el borde de la lente, apagando las tonalidades grises que se encuentran presentes en la imagen original.

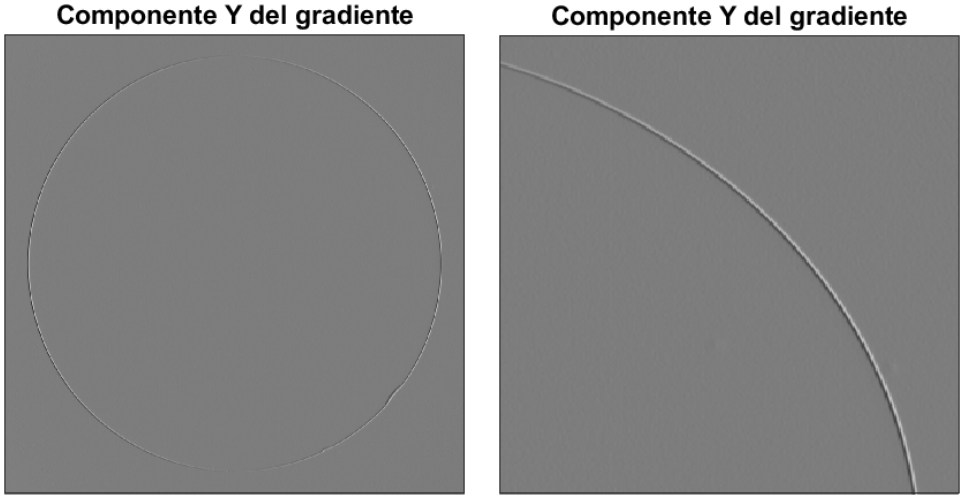
3. Resultados

A continuación se muestran los resultados obtenidos para la imagen de prueba, siendo esto: componente G_x de la gradiente 2 , componente G_y de la gradiente 3 , ∇f 4 y finalmente θ 5 . Además, se muestra el resultado de utilizar la función *quiver()* para mostrar los vectores como flechas en la imagen 6.



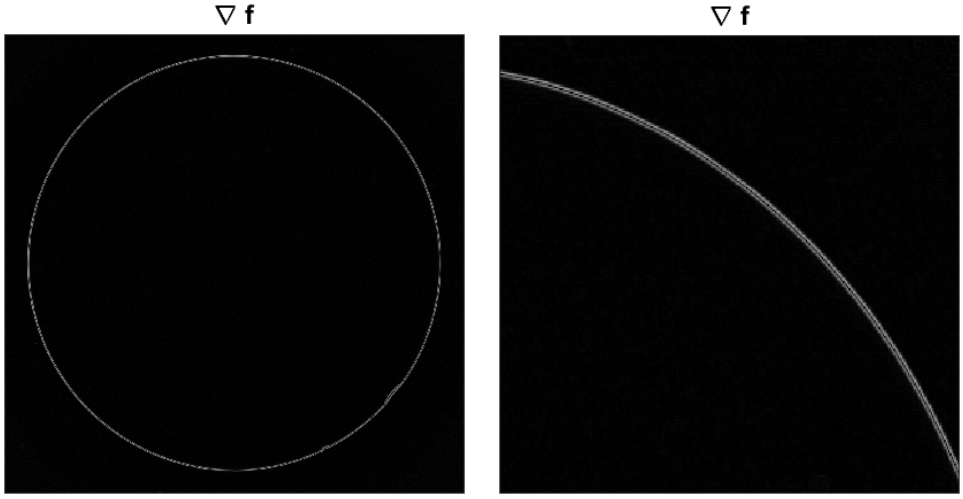
(a) Componente G_x de la imagen (b) Acercamiento

Figura 2: Resultados para G_x



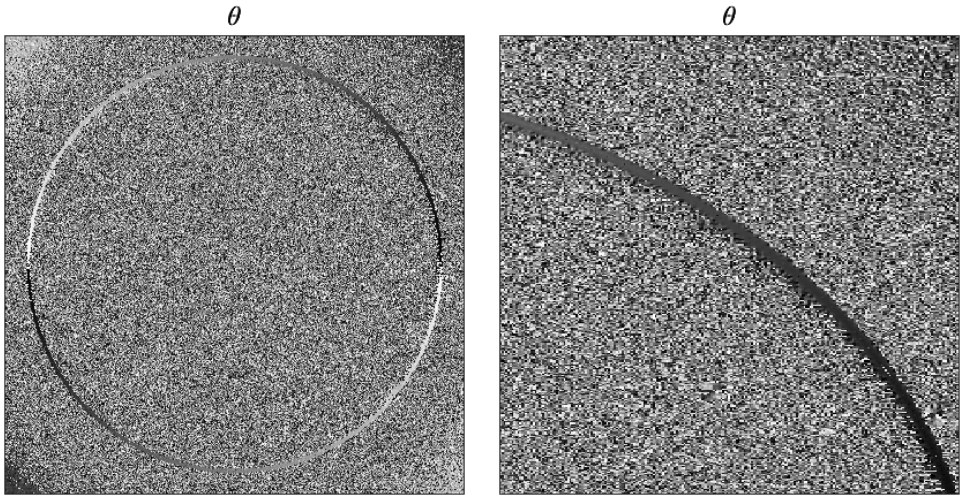
(a) Componente G_y de la imagen (b) Acercamiento

Figura 3: Resultados para G_y



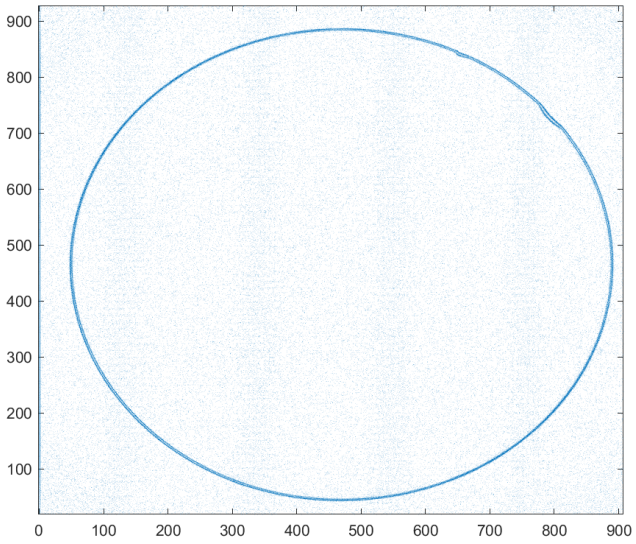
(a) ∇f (b) Acercamiento

Figura 4: Resultados para ∇f

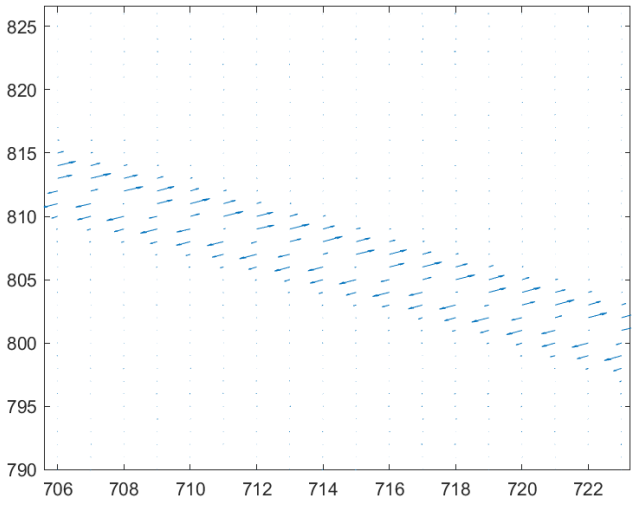


(a) θ (b) Acercamiento

Figura 5: Resultados para θ



(a) *Quiver*



(b) Acercamiento

Figura 6: Resultados para *quiver*

Los componentes G_x y G_y, a pesar de ser visibles, son muy delgados y se pierden entre todos los píxeles de la imagen. Ambos presentan valores similares, lo que era de esperar dado que el objeto es circular, con valores similares en los componentes de la gradiente. De igual forma, el valor del ángulo de la gradiente deja denotar el contorno del objeto, pero este se pierde entre todo el ruido que se tiene en el fondo de la imagen. El mejor resultado para visualizar el borde se da en ∇f, donde los únicos píxeles iluminados de blanco se corresponden a los bordes, mientra que el fondo toma un color negro. De esta manera, se visualiza con gran detalle la forma del lente, y se destacan los errores presentes.

Se destaca la visualización del vector en sí obtenida mediante *quiver*. Denota de forma excelente el contorno del lente, a pesar del ruido presente en los píxeles de fondo. Sin embargo, esta función aplica un escalado automático a cada flecha, y dado que existe una para cada pixel en la imagen, ocasiona que estas sean muy pequeñas para apreciarse a simple vista. Sin embargo, al acercarse a una zona en específico se puede observar la dirección de la derivada ubicada en el borde del objeto.

4. Conclusiones

Se obtuvieron buenos resultados para el cálculo del gradiente en la imagen utilizando los operadores de Sobel, permitiendo observar a detalle el borde del objeto presente en la imagen. La aplicación de la convolución no se hizo de una forma eficaz, y sin embargo su cálculo es rápido, dado que Matlab esta optimizado para operaciones matriciales de este tipo. A pesar de estos resultados, se necesitan métodos más refinados para obtener una mejor detección de bordes, dado que se presenta ruido en varias zonas del fondo de la imagen.

Referencias

[1] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Pearson, New York, fourth edition, global edition edition, 2017.

5. Anexo

5.1. Código

```
% Se empieza leyendo la imagen y convirtiendola a double%
img = imread("Fig3.45(a).jpg");
img = double(img);

% Funcion para hacer padding y agregar ceros a los extremos de la imagen
% De esta manera, hacemos la convolucion sin perder informacion
function padded_img = padding(m)
    padded_img = [zeros(1,size(m,2)+2); [zeros(size(m,1),1), m, ...
        zeros(size(m,1),1)]; zeros(1,size(m,2)+2)];
end

% Imagen con ceros agregados
pad_img = padding(img);

% Operadores de sobel para parcial en x e y
sobel_x = [-1 -2 -1; 0 0 0; 1 2 1];
sobel_y = [-1 0 1; -2 0 2; -1 0 1];

% Creacion de las matrices donde se guardaran los resultados
[rows, cols] = size(pad_img);
gx = zeros(size(img));
gy = zeros(size(img));

% Proceso de convolucion. Se realiza con ambos operadores para
% ahorrar tiempo de computo. Al hacerlo sobre pad_img, no se
% pierde informacion
for i=2:rows-1
    for j=2:cols-1
        f = pad_img(i-1:i+1, j-1:j+1);
        gx(i-1, j-1) = sum(sum(sobel_x.*f));
        gy(i-1,j-1) = sum(sum(sobel_y.*f));
    end
end

% Obtenemos la magnitud del gradiente
grad_norm = sqrt(gx.^2 + gy.^2);

% Obtenemos las direcciones de los vectores
grad_dir = atan(gy ./ gx);

%%%% Graficacion de los resultados %%%

% Componentes x e y
f1 = figure('Name', 'Resultados');
subplot(2,2,1);
imagesc(gx);
title('Componente␣X␣del␣gradiente');
set(gca,'XTick',[], 'YTick', []);
axis square;
```

```
subplot(2,2,2);
imagesc(gy);
title('Componente␣Y␣del␣gradiente');
set(gca,'XTick',[], 'YTick', []);
axis square;


%%% Valor de la norma y direccion del gradiente
subplot(2,2,3);
imagesc(grad_norm);
title('\nabla␣f');
set(gca,'XTick',[], 'YTick', []);
axis square;


subplot(2,2,4);
imagesc(grad_dir);
title('\theta');
set(gca,'XTick',[], 'YTick', []);

colormap("gray");
axis square;

%%% Nueva figura, resultado del quiver para la gradiente
f2 = figure('Name', 'Vector␣con␣quiver');
q = quiver(gx, gy, 1);
```