Detección de bordes

Algoritmo Canny

El algoritmo Canny es el siguiente

1. Calcula la derivada en x y y de la imagen

2. Calcula la magnitud y orientación del gradiente en cada pixel

3. Elimina los pixels que no son máximos locales en su magnitud en la dirección del gradiente

4. Umbrales por histéresis
    1. Selecciona los pixels mayores a un umbral M > Th
    2. Recolecta los pixels M > Tl (umbral menor) que tienen vecinos de los puntos recolectados

En las siguientes secciones describiremos cada uno de los pasos de este algoritmo.

Dr. Carlos López Franco

1. Cálculo de derivada en x y y de la imagen

Dado que las imágenes contienen ruido es necesario utilizar un filtro suavizador en combinación con un filtro derivador.

Una opción es el filtro Prewitt, el cual combina el filtro pasa bajas (promedio) con un derivador (diferencias centrales).

Los filtros Prewitt son los siguientes

Hx=

-1	**0**	**1**
-1	0	1
-1	0	1

Hy=

-1	-1	-1
0	0	0
1	1	1

Similarmente tenemos los filtros Sobel, los cuales son una combinación de el filtro Gaussiano y un derivador (diferencias centrales).

Los filtros Sobel son los siguientes

Hx=

-1	**0**	**1**
-2	0	2
-1	0	1

Hy=

-1	-2	-1
0	0	0
1	2	1

2. Calcula la magnitud del gradiente en cada pixel

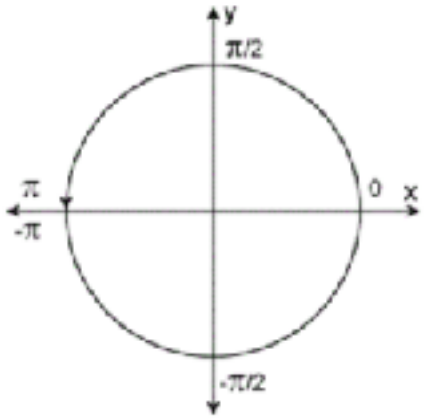
$$\|\nabla F\| = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2}$$

$$\theta = \tan^{-1} \frac{\partial F}{\partial y} \Big/ \frac{\partial F}{\partial x}$$

Nota: para calcular el ángulo del gradiente se recomienda usar la función atan2

3. **Elimina los pixels que no son máximos locales en su magnitud en la dirección del gradiente**
    1. **Normalización de los ángulos del gradiente**

Los ángulos del gradiente están definidos en el rango de la función atan2, es decir -pi a pi



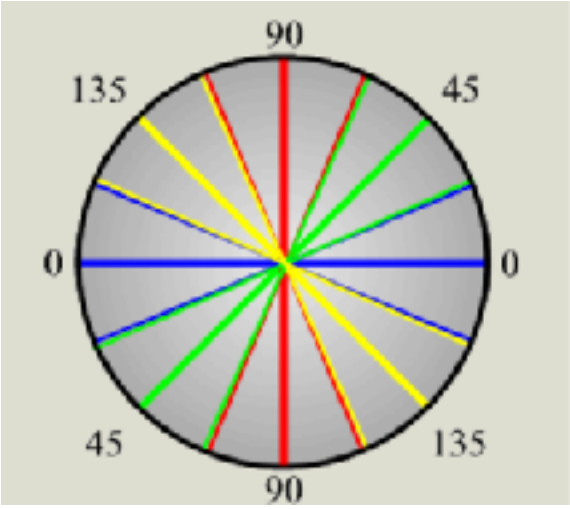
Dado que en nuestro caso no es necesario distinguir entre 90º o -90º sumamos 180 a todos los ángulos negativos, así tendremos solo ángulos positivos.

En el caso de una matriz cuadrada solo tenemos 4 opciones,

Las cuales representan 0º, 45º, 90º, 135º.

Ahora, deberas convertir todos los ángulos a solo estos cuatro valores. Para convertirlos utilizaremos el rango de 22.5º si es menor pasa al valor inmediato inferior y si es mayor o igual pasa al siguiente valor.

Por ejemplo, si el valor es 15º el valor deberá ser 0º. Si el valor es 22.6 el valor será 45º.



2. Indices de pixels vecinos

Una vez que todos los ángulos son alguno de los valores 0º,45º,90º o 135º, debemos calcular los pixels vecinos.

Para cada pixel (color rojo), se debe calcular los indices de los vecinos dependiendo de la dirección del gradiente (ángulos 0º,45º,90º o 135º)

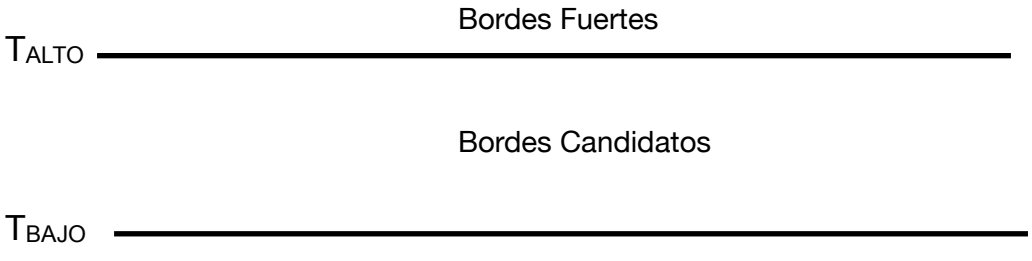
3. **Supresión de pixels que no son máximos locales en su magnitud en la dirección del gradiente**

Este paso requiere de los vecinos de cada pixel, por lo que cada pixel es comparado con sus dos pixels vecinos (vecinos en la orientación del gradiente). Si la magnitud del gradiente del pixel central es mayor que la magnitud del gradiente de sus vecinos entonces se mantiene su valor, de lo contrario se hace cero.

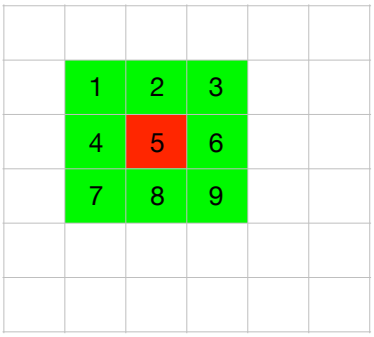
Utilizando lo anterior, se construye una nueva imagen la cual contendrá solo los pixels cuya magnitud sea mayor.

4. **Filtrado mediante umbrales por histéresis**

La salida de la supresión de pixels no máximos sigue conteniendo máximos locales con ruido.
Lo que se necesita es eliminar los bordes más débiles sin dejar desconectado los bordes más fuertes con los bordes candidatos.
Por lo anterior, se aplica un doble umbral T_{ALTO} y T_{BAJO}, los bordes que pasan el umbral alto se almacenan, los bordes que son menores al umbral bajo se eliminan. Los bordes intermedios, que reciben el nombre de bordes candidatos necesitan validarse, y solo se mantienen si son vecinos de un borde fuerte.

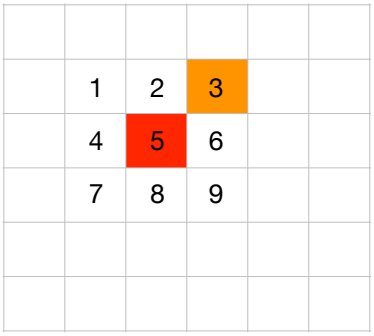


Como podemos observar en la figura anterior aceptar los bordes fuertes y descartar los bordes débiles es simple. En el caso de los bordes candidatos, necesitamos hacer una revisión más, esta revisión nos ayudará a decidir si el borde candidato se mantiene o se descarta. La forma en la que tomamos esta decisión es si el borde candidato es vecino de un borde fuerte.



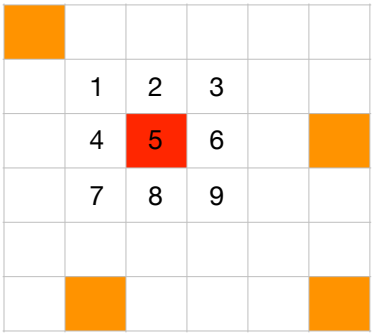
En la imagen siguiente, supongamos que el pixel central es el pixel número 5. Sus vecinos serían entonces los pixels 1,2,3,4,6,7,8,9 (pixels verdes). A esto se le llama vecindad de conexión 8.

Supongamos que el pixel rojo es un borde candidato, y el borde naranja



representa un borde fuerte.

En este caso, el borde candidato se mantiene dado que es vecino de un borde fuerte.



Por el contrario, supongamos que tenemos el siguiente caso.

Dado que en este caso el borde candidato no es vecino de ningún borde fuerte, el borde se descarta.

1. Construye una matriz para almacenar el resultado
2. Revisa la matriz de supresión de bordes no máximos y decide lo siguiente
    1. Si la magnitud del borde es mayor que el umbral T_{ALTO} se mantiene, (coloca un 1 en la imagen resultado)
    2. Si la magnitud del borde es menor que el umbral T_{BAJO} se descarta, (coloca un 0 en la imagen resultado)
    3. Si el borde es un borde candidato, entonces revisa la magnitud de sus bordes vecinos. Si existe por lo menos un vecino que sea un borde fuerte entonces el borde candidato se mantiene (coloca un 1 en la imagen resultado).

La imagen final es el resultado de la aplicación del algoritmo de detección de bordes Canny.

Nota: Una cuestión importante es como definir los umbrales, T_{ALTO} y T_{BAJO}, lo cierto es que ambos umbrales dependen de la imagen, por lo que varian en cada caso.

Sin embargo una opción muy utilizada es definirlos con base a la máxima magnitud del gradiente en toda la imagen.

$$T_{ALTO} = Factor_{Alto} Mag_{Max}$$
$$T_{BAJO} = Factor_{Bajo} Mag_{Max}$$

Donde $Factor_{Alto}$ = 0.175 y $Factor_{Bajo}$= 0.075