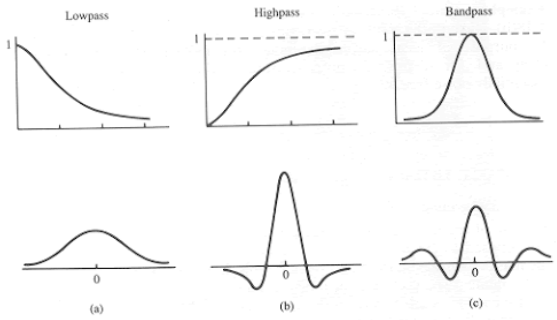
Practica 14 – Filtro Canny

Asignatura: Procesamiento Digital de Imágenes
Profesor: Dr.Sc. Gerardo García Gil
Alumno: René Francisco Coss y León Monterde
Registro: 17310066 **Semestre:** 2020-B
Ingeniería en Desarrollo de Software

INTRODUCCIÓN.

Filtrado (Imágenes Digitales)

Son técnicas dentro del preprocesamiento de imágenes para obtener, a partir de una imagen, otra que sea más adecuada para una aplicación específica, mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella.



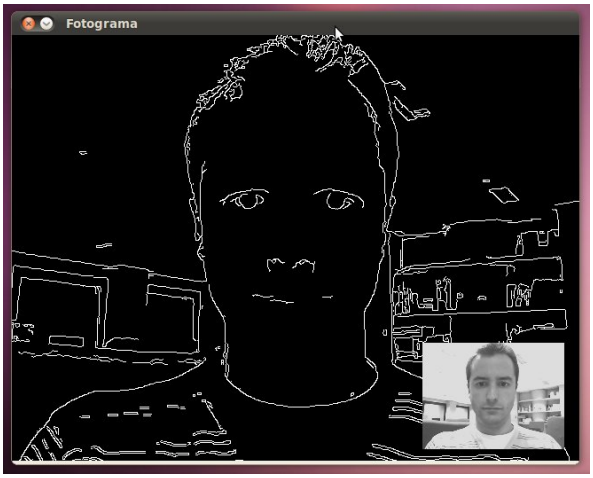
Filtro (Imágenes Digitales)

Es la aplicación de un filtrado de imagen para lograr el cambio en una imagen, el cual no depende únicamente el píxel original, sino de otros píxeles que están en una determinada vecindad en relación a este.

(Ejemplo: Filtro Laplaciano y Sobel)

Detección de bordes

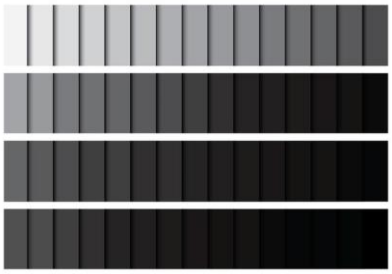
Los bordes de una imagen digital se definen como transiciones entre dos regiones de niveles de gris significativamente distintos.



Permiten reconocer las fronteras de los objetos y segmentar la imagen, así como detectar patrones para reconocer objetos.

Gradiente

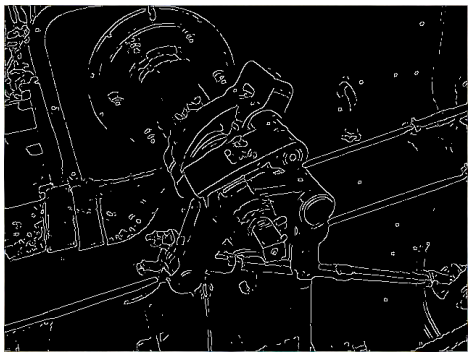
El gradiente de una imagen mide cómo esta cambia en términos de color o intensidad.



La magnitud del gradiente indica la rapidez con la que la imagen cambia, mientras que la dirección del gradiente indica la dirección en la que está cambiando.

Filtro Roberts

Filtro utilizado para la detección de bordes y formas, que utiliza un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes.



La función óptima en el algoritmo de Canny es descrito por la suma de cuatro términos exponenciales, pero se puede aproximar por la primera derivada de una gaussiana.

Matemáticamente

El algoritmo de Canny utiliza cuatro filtros para detectar horizontal, vertical y diagonal en los bordes de la imagen borrosa.

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2} \qquad \mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Pseudocódigo

Algoritmo CANNY {
  $S \leftarrow G_\sigma * I$
  $[S_x\ S_y] \leftarrow \nabla(S)$
  Para cada *pixel* (i, j)
    $|\nabla S(i,j)| \leftarrow \sqrt{S_x^2(i,j) + S_y^2(i,j)}$
    $S_\phi(i,j) \leftarrow \arctan \frac{S_x(i,j)}{S_y(i,j)}$
  SUPRESIÓN
  HYSTERESIS
  Devolver $\{|\nabla S|, S_\phi\}$
}

DESARROLLO.

Se obtiene la imagen a escala de grises con las funciones ya vistas, el proceso de lectura de un archivo de imagen es el mismo que se ha utilizado hasta el momento.

La codificación es muy repetitiva, es cíclica y cuenta con demasiados ciclos basados en el pesudocódigo.

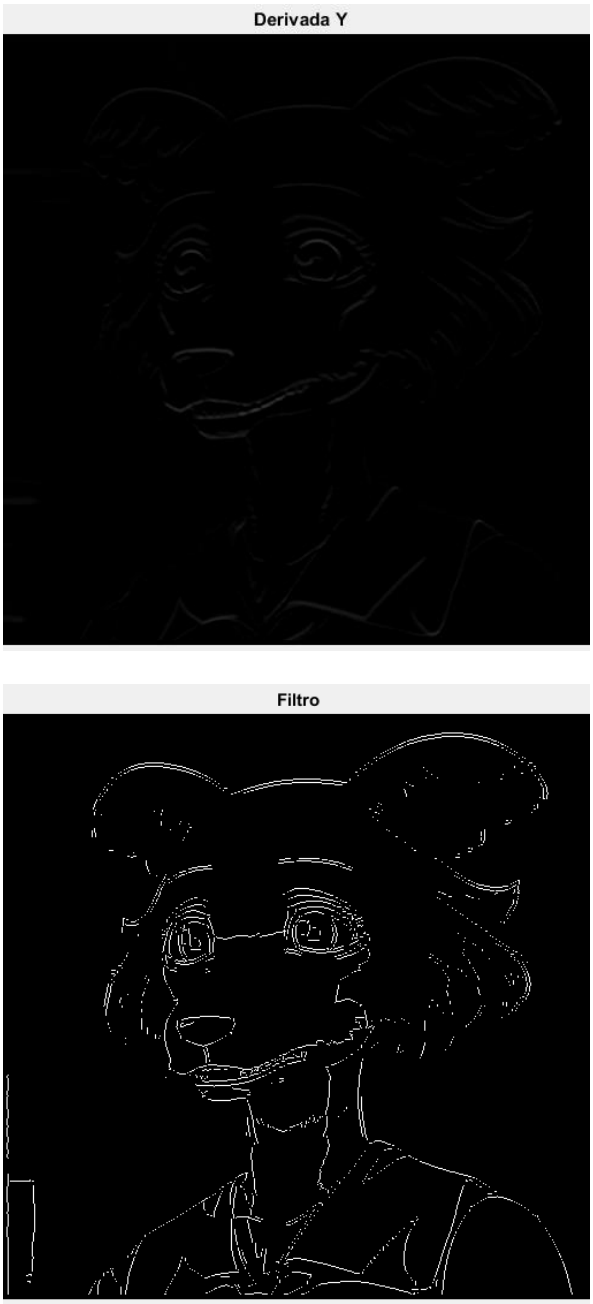


Fig1. Comparación de la imagen original (Primera) con la derivación en ambos ejes y el resultado final.

Al igual que la práctica de Sobel, la función de este filtro no se guarda en una variable, por lo que se vuelve un proceso en sí mismo y la reutilización del código a futuro es muy ineficiente. Es decir, que a nivel código, esta función no es modular, pero esto no tiene impacto en el efecto del filtro.

Código

```
clear all; clc;

A = imread('junobest.jpg');

gr = escalagris(A);
figure(1);
imshow(gr);
title("Grises Original");

canny_edges(1.5, 0.05, 1.0);

function
canny_edges(max_hysteresis_thresh,
min_hysteresis_thresh, sigma)

ORIGINAL_IMAGE=imread('junobest.jpg'
);

ORIGINAL_IMAGE=im2double(ORIGINAL_IM
AGE);

[H,W]=size(ORIGINAL_IMAGE);

derivative_x=zeros(H,W);
derivative_y=zeros(H,W);

size_of_kernel = 6*sigma+1;
adjust = ceil(size_of_kernel/2);
Y_GAUSSIAN=zeros(size_of_kernel,size
_of_kernel);
X_GAUSSIAN=zeros(size_of_kernel,size
_of_kernel);

for i=1:size_of_kernel
    for iiii=1:size_of_kernel
        Y_GAUSSIAN(i,iiii) = -( (i-
((size_of_kernel-1)/2)-1)/( 2* pi *
sigma^3 ) ) * exp ( - ( (i-
((size_of_kernel-1)/2)-1)^2 + (iiii-
((size_of_kernel-1)/2)-1)^2 )/
(2*sigma^2) );
    end
end

for i=1:size_of_kernel
    for iiii=1:size_of_kernel
        X_GAUSSIAN(i,iiii) = -(
(iiii-((size_of_kernel-1)/2)-1)/( 2*
pi * sigma^3 ) ) * exp ( - ( (i-
((size_of_kernel-1)/2)-1)^2 + (iiii-
((size_of_kernel-1)/2)-1)^2 )/
(2*sigma^2) );
    end
end

GRADIENT =  zeros(H,W);
non_max = zeros(H,W);
post_hysteresis = zeros(H,W);

for r=1+ceil(size_of_kernel/2):H-
ceil(size_of_kernel/2)
    for
c=1+ceil(size_of_kernel/2):W-
ceil(size_of_kernel/2)
        reference_row=  r-
ceil(size_of_kernel/2);
        reference_colum=  c-
ceil(size_of_kernel/2);
        for yyy=1:size_of_kernel
            for
yyy_col=1:size_of_kernel
                derivative_x(r,c) =
derivative_x(r,c) +
ORIGINAL_IMAGE(reference_row+yyy-1,
reference_colum+yyy_col-
1)*X_GAUSSIAN(yyy,yyy_col);
            end
        end
    end
end

for r=1+ceil(size_of_kernel/2):H-
ceil(size_of_kernel/2)
    for
c=1+ceil(size_of_kernel/2):W-
ceil(size_of_kernel/2)
        reference_row=  r-
ceil(size_of_kernel/2);
        reference_colum=  c-
ceil(size_of_kernel/2);
        for yyy=1:size_of_kernel
            for
yyy_col=1:size_of_kernel
                derivative_y(r,c) =
derivative_y(r,c) +
ORIGINAL_IMAGE(reference_row+yyy-1,
reference_colum+yyy_col-
1)*Y_GAUSSIAN(yyy,yyy_col);
            end
        end
    end
end

for r=1+ceil(size_of_kernel/2):H-
ceil(size_of_kernel/2)
    for
c=1+ceil(size_of_kernel/2):W-
ceil(size_of_kernel/2)
```

```
        GRADIENT(r,c) = sqrt
(derivative_x(r,c)^2 +
derivative_y(r,c)^2 );
    end
end

non_max = GRADIENT;

for r=1+ceil(size_of_kernel/2):H-
ceil(size_of_kernel/2)
    for
c=1+ceil(size_of_kernel/2):W-
ceil(size_of_kernel/2)
        if (derivative_x(r,c) == 0)
            tangent = 5;
        else
            tangent =
(derivative_y(r,c)/derivative_x(r,c)
);
        end

        if (-0.4142<tangent &
tangent<=0.4142)

if(GRADIENT(r,c)<GRADIENT(r,c+1) |
GRADIENT(r,c)<GRADIENT(r,c-1))
                non_max(r,c)=0;
            end
        end

        if (0.4142<tangent &
tangent<=2.4142)

if(GRADIENT(r,c)<GRADIENT(r-1,c+1) |
GRADIENT(r,c)<GRADIENT(r+1,c-1))
                non_max(r,c)=0;
            end
        end

        if ( abs(tangent) >2.4142)

if(GRADIENT(r,c)<GRADIENT(r-1,c) |
GRADIENT(r,c)<GRADIENT(r+1,c))
                non_max(r,c)=0;
            end
        end

        if (-2.4142<tangent &
tangent<= -0.4142)

if(GRADIENT(r,c)<GRADIENT(r-1,c-1) |
GRADIENT(r,c)<GRADIENT(r+1,c+1))
                non_max(r,c)=0;
            end
        end
    end
end

post_hysteresis = non_max;

for r=1+ceil(size_of_kernel/2):H-
ceil(size_of_kernel/2)
    for
c=1+ceil(size_of_kernel/2):W-
ceil(size_of_kernel/2)

if(post_hysteresis(r,c)>=max_hystere
sis_thresh)
            post_hysteresis(r,c)=1;
        end


if(post_hysteresis(r,c)<max_hysteres
is_thresh &
post_hysteresis(r,c)>=min_hysteresis
_thresh)
            post_hysteresis(r,c)=2;
        end


if(post_hysteresis(r,c)<min_hysteres
is_thresh)
            post_hysteresis(r,c)=0;
        end
    end
end

vvvv = 1;

while (vvvv == 1)
    vvvv = 0;
    for
r=1+ceil(size_of_kernel/2):H-
ceil(size_of_kernel/2)
        for
c=1+ceil(size_of_kernel/2):W-
ceil(size_of_kernel/2)
            if
(post_hysteresis(r,c)>0)

if(post_hysteresis(r,c)==2)
                    if(
post_hysteresis(r-1,c-1)==1 |
post_hysteresis(r-1,c)==1 |
post_hysteresis(r-1,c+1)==1 |
post_hysteresis(r,c-1)==1 |
post_hysteresis(r,c+1)==1 |
post_hysteresis(r+1,c-1)==1 |
post_hysteresis(r+1,c)==1 |
post_hysteresis(r+1,c+1)==1 )
post_hysteresis(r,c)=1;
                        vvvv == 1;
```

```
                    end
                end
            end
        end
    end
end

for r=1+ceil(size_of_kernel/2):H-
ceil(size_of_kernel/2)
    for
c=1+ceil(size_of_kernel/2):W-
ceil(size_of_kernel/2)
        if(post_hysteresis(r,c)==2)
            post_hysteresis(r,c)==0;
        end
    end
end

figure(2)
imshow(derivative_x);
title("Derivada X");

figure(3)
imshow(derivative_y);
title("Derivada Y");

figure(4)
imshow(post_hysteresis);
title("Filtro");
end

function gris = escalagris(img)
  [r c z] = size(img);

  for i=1:r
    for j=1:c
      gris(i, j) = img(i,j,1)*0.2989
+ img(i,j,2)*0.5870 +
img(i,j,3)*0.1140;
    end
  end
end
```

CONCLUSIONES

El filtro Canny permite la visualización de los bordes de manera refinada cuando se hacen los suficientes ciclos, los efectos de este filtro son muy notorios, puesto las líneas que toma son mucho más delgadas, y si se tiene comparación con los filtros previamente mencionados (Sobel Roberts y Prewitt) se puede notar el cambio en el grosor de la línea.

En sí, este filtro es el más complicado de realizar, aunque es el que mejores resultados otorga, lo cual lo vuelve objetivamente mejor que los demás, puesto aunque sea mayor su dificultad y lógica de entendimiento, las bases siguen siendo las mismas, y los bordes obtenidos definen mejor las formas.

BIBLIOGRAFÍA

colaboradores de Wikipedia. (2020, 5 agosto). Procesamiento digital de imágenes. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Procesamiento_digital_de_im%C3%A1genes

Filtrado espacial. (s. f.). uniovi. Recuperado 1 de octubre de 2020, de http://www6.uniovi.es/vision/intro/node41.html

Sosa-Costa, A. (2019, 30 agosto). Gradiente de Imágenes. ▷ Cursos de Programación de 0 a Experto © Garantizados. https://unipython.com/gradiente-de-imagenes/

Rodriguez, M. R. (s. f.). Canny Edge Detector. Canny Edge Detector. Recuperado 30 de octubre de 2020, de http://www.cs.ucf.edu/%7Emikel/Research/Edge_Detection.htm

Apuntes tomados en clase.