UNIVERSIDAD DE CÓRDOBA

INGENIERÍA TÉCNICA EN INFORMÁTICA SISTEMAS
PERCEPCIÓN

EJERCICIO FINAL

ALGORITMO DE CANNY

Manual Técnico

Autor: Raúl Pérula Martínez

Fecha: 09 de Febrero de 2009

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 ENUNCIADO

Implemente el algoritmo de Canny para imágenes en nivel de gris. Se deberá crear un programa que permita aplicar el algoritmo a una imagen indicando los parámetros por línea de comandos.

En relación a la selección de umbrales, estos deberán ser indicados como un porcentaje de la magnitud del gradiente en lugar de utilizar umbrales fijos.

1.2 GLOSARIO

Suavizado gaussiano: cuando se aplica una máscara de tipo gaussiana a una imagen lo que se hace es alisar y volver más uniforme dicha imagen.

Pixel: superficie homogénea más pequeña de las que componen una imagen, que se define por su brillo y color.

Gradiente: razón entre la variación del valor de una magnitud en dos puntos próximos y la distancia que los separa.

Convolución: acción de aplicar una máscara a una imagen.

Máscara: matriz con un conjunto de números determinados que sirve para aplicar un efecto o filtrado a una imagen.

Umbral: valor mínimo de una magnitud a partir del cual se produce un efecto determinado.

Ruido: interferencia o deterioro en una imagen.

2 ANTECEDENTES

2.1 DETECCIÓN DE BORDES

Hay muchas maneras para la detección de puntos, líneas y formas. Existen diferentes métodos, ya que depende del objeto que se muestre en una imagen.

Estos métodos son utilizados, como por ejemplo, en industrias como comprobación de defectos en los elementos fabricados, en robótica, en medicina y en muchas más áreas en las cuales la visión artificial es muy útil.

Comencemos con el reconocimiento más simple, el de un punto; un punto puede ser reconocido en una imagen cuando al aplicar una máscara como esta:

-1	-1	-1
-1	8	-1
-1	-1	-1

se consigue que el resultado de aplicarla sea un valor; si ese valor se encuentra por encima de umbral establecido, se reconoce que es un punto.

De una manera similar, con otros tipos de máscaras se puede detectar el contorno de una línea. En este caso, hará falta la dirección en que la línea continúa para poder realizar el reconocimiento. Algunas de las máscaras que existen son:

-1	-1	-1
2	2	2
-1	-1	-1

-1	-1	2
-1	2	-1
2	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

2	-1	-1
-1	2	-1
-1	-1	2

Los principales pasos para la detección de bordes son:

- Suavizado Gausiano: empleado para eliminar el ruido.
- Aplicación de un filtro derivativo, como por ejemplo Sobel, Roberts o Prewitt.
- Detección de un borde:
  - Primero: poner un umbral para decidir que pixel es borde y cuáles no.
  - Segundo: reducir los bordes a tamaño uno.

2.2 SOBEL, ROBERTS Y PREWITT

Los filtros diferenciales realizan la derivada de la función f(x, y) en cada punto. Para ello se hace uso del gradiente.

El cálculo se divide en las componentes horizontales y verticales. Aquí es donde entran Sobel, Roberts y Prewitt. Cada uno tiene sus propias características, y cada uno aplica una máscara diferente.

Roberts es quizá el más sencillo, ya que aplica una máscara de 2x2 de la siguiente forma:

Vertical

1	0
0	-1

Horizontal

0	1
-1	0

Aquí tenemos un ejemplo de cómo se vería una imagen con un filtro de Roberts aplicado:

Imagen original

Filtro Roberts

Figura 1: Ejemplo Roberts

También tenemos el filtro de Prewitt, este filtro tiene una máscara más amplia, ya que aumenta a tamaño 3x3. Las dos componentes tiene la siguiente forma:

Vertical

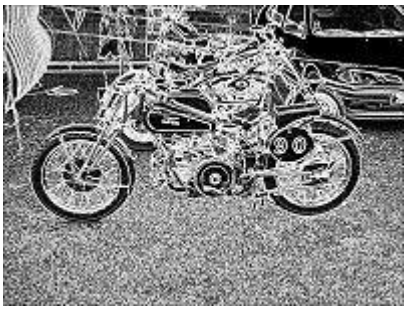
-1	-1	-1
0	0	0
1	1	1

Horizontal

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

Como puede observarse ahora lo que tiene menos importancia es lo que se encuentra en el centro. Un ejemplo de este filtro sería:

Imagen original



Filtro Prewitt

Figura 2: Ejemplo Prewitt

Por último, tenemos el filtrado de Sobel, este filtrado realza los bordes, es uno bastante utilizado. Sus máscaras tienen la siguiente forma:

Vertical

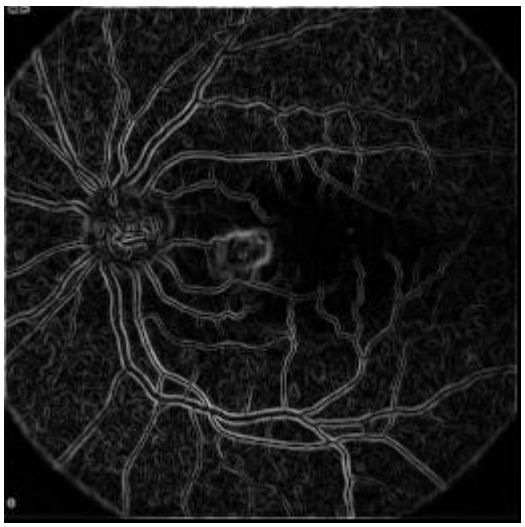
-1	-2	-1
0	0	0
1	2	1

Horizontal

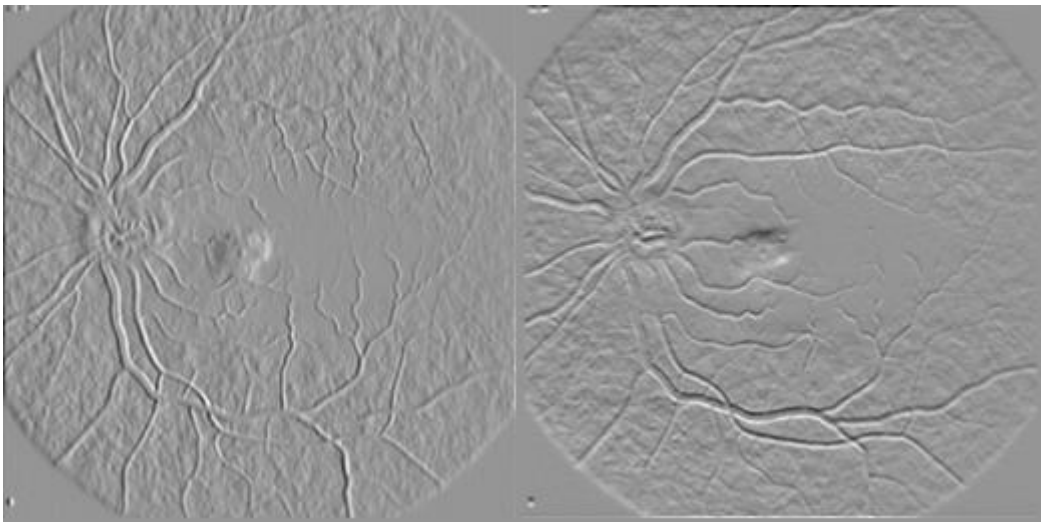
-1	0	1
-2	0	2
-1	0	1

Un ejemplo de la aplicación de este filtrado es:

Imagen original



Filtro Sobel



Gradiente horizontal

Gradiente vertical

Figura 3: Ejemplo Sobel

A partir de las componentes verticales y horizontales se puede conocer tanto el ángulo como el módulo del gradiente:

$$\emptyset(x,y) = atan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

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

2.3 SUAVIZADO GAUSSIANO

Este filtro es muy comúnmente utilizado sobre todo para la eliminación de ruido en la detección de bordes en imágenes. Este filtrado o suavizado lo que hace es aplicar una máscara la cual le da mucha más importancia al centro que al resto de pixeles.

La máscara tendrá un tamaño predefinido por las variables "media" y "desviación típica", en el caso general, la media se toma como cero, y lo que hace que el tamaño de la máscara varíe es la desviación típica, es decir, el tamaño de la máscara será: 4*sigma+1.

Una imagen de la distribución normal que es la que sigue la función gaussiana:

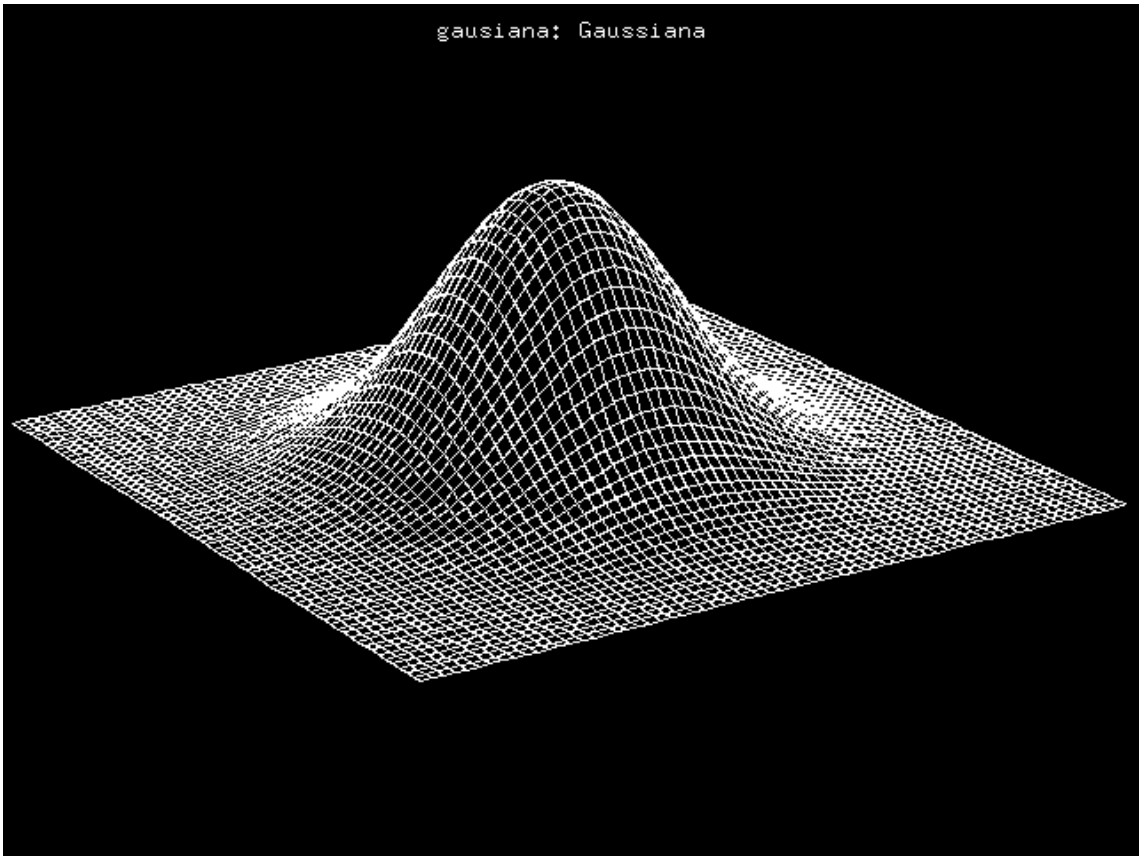


Figura 4: Distribución Normal

3 ALGORITMO DE CANNY

En 1986, Canny propuso un método para la detección de bordes, el cual se basaba en tres criterios, estos son:

- Un criterio de detección expresa el hecho de evitar la eliminación de bordes importantes y no suministrar falsos bordes.
- El criterio de localización establece que la distancia entre la posición real y la localizada del borde se debe minimizar.
- El criterio de una respuesta que integre las respuestas múltiples correspondientes a un único borde. [6.]

3.1 PRIMER PASO: SUAVIZADO GAUSIANO

Lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original con el objetivo de suavizar la imagen y tratar de

eliminar el posible ruido existente. Sin embargo, se debe de tener cuidado de no realizar un suavizado excesivo, pues se podrían perder detalles de la imagen y provocar un pésimo resultado final. Este suavizado se obtiene promediando los valores de intensidad de los píxels en el entorno de vecindad con una máscara de convolución de media cero y desviación estándar sigma.

Dos máscaras muy comúnmente utilizadas para la aplicación del suavizado gaussiano son:

(a)

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

(b)

$\frac{1}{115}$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Figura 5: Máscaras gaussianas

3.2 Segundo Paso: Obtención del Gradiente

Para la obtención del gradiente se puede aplicar cualquiera de las máscaras que se han mostrado antes cuando se ha explicado Roberts, Prewitt o Sobel, cualquiera de estas sirven para la obtención del gradiente.

Una vez aplicada la máscara, ya tenemos las componentes horizontal y vertical, así que ahora ya podemos calcular la magnitud y el ángulo del gradiente.

Como se comentó anteriormente, para el cálculo de la magnitud y el ángulo se procederá con estas dos expresiones:

$$\emptyset(x,y) = atan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

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

3.3 Tercer Paso: Supresión No Máxima

Las dos imágenes generadas en el paso anterior sirven de entrada para generar una imagen con los bordes adelgazados.

El procedimiento es el siguiente: se consideran las 8 direcciones identificadas por las orientaciones de 0º, 45º, 90º, 135º, 180º, 225º, 270º, 315º (en radianes, 0, π/4, π/2, 3π/4, π, 5π/4, 3π/2, 7π/4) con respecto al eje horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente y se sectoriza con un valor entre 0 y 7, este valor se almacenará en el gradiente sustituyendo al del ángulo.

Posteriormente se observa si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior. De ser así se asigna el valor 0 a dicho píxel, en caso contrario se asigna el valor que tenga la magnitud del gradiente.

3.4 Cuarto Paso: Histéresis (Umbralización)

La imagen obtenida en el paso anterior suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral.

El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo.

Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayores al primer umbral. Así se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. [6.]

3.2 Último Paso

Frecuentemente, es común que un cuarto y último paso se realice en el algoritmo de Canny, este paso consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido.

Un método muy utilizado es el algoritmo de Deriche y Cocquerez. Este algoritmo utiliza como entrada una imagen binarizada de contornos de un píxel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con otro extremo abierto.

El procedimiento consiste en buscar para cada píxel uno de los ocho patrones posibles que delimitan la continuación del contorno en tres direcciones posibles. Esto se logra con la convolución de cada píxel con una máscara específica. Cuando alguno de los tres puntos es ya un píxel de borde se entiende que el borde se ha cerrado, de lo contrario se elige el píxel con el valor máximo de gradiente y se marca como nuevo píxel de borde y se aplica nuevamente la convolución. Estos pasos se repiten para todo extremo abierto hasta

encontrar su cierre o hasta llegar a cierto número de iteraciones determinado. [6.]

4 DISEÑO

4.1 ANTECEDENTES

Antes de explicar de forma algorítmica la función de Canny hay que explicar un poco y de manera clara donde se va a implementar dicha función.

La función de Canny formará parte de un conjunto de funciones y clases definidas previamente. Existe una clase *Image*, la cual contendrá los métodos relacionados con las imágenes en ppm o pgm, algunos de estos son la carga y el salvado de una imagen, así como los correspondientes constructores. También existe una clase *FImage*, esta clase tiene los mismos métodos que la clase *Image*, la diferencia se encuentra en que esta clase trata imágenes de manera flotante, ya que se podrán realizar operaciones con ellas.

La clase *ImageArith* contiene todas las operaciones posibles a realizar con objetos de las clases *Image* y *FImage*. Algunas de estas operaciones son, suma, diferencia, multiplicación, etc.

También existe una clase *Histogram*, esta clase contiene los métodos necesarios para crear histogramas de los objetos de las clases *Image* y *FImage*.

Existen también dos clases, una llamada *LinearFunction* y otra *LinearContrast*, la clase *LinearContrast* modifica el contraste de una imagen dada una función linear, esta función linear es tratada en la clase *LinearFunction*.

Otra de las clases creadas es la clase *MorphologicalUtils*, esta clase tiene implementados los métodos relacionados con la segmentación de imágenes, contiene por ejemplo la erosión, la dilatación, la apertura o cierre de una imagen.

Por último, tenemos la clase *ImageUtils*, esta clase va a ser la que contenga la función de Canny, en ella se pueden ver métodos tales como la convolución, el algoritmo de Sobel y alguno más.

También hay que decir que todas estas clases, en sus métodos, tienen control de errores, este control se ha realizado gracias a la clase *ImageException*, la clase *ImageException* detiene el programa y muestra un mensaje cuando existe un error en un método.

4.2 Función Algoritmo de Canny

Algoritmo: Suavizado Gaussiano

Entradas:

- Imagen de entrada original I
- Máscara G, con media cero y desviación típica sigma

Salidas:

- Imagen suavizada IS

Pasos:

1. Suavizar la imagen I con G mediante un filtro gaussiano y obtener IS como imagen de salida.

Algoritmo: Obtención del gradiente

Entradas:

- Imagen suavizada IS
- Máscaras para la obtención de las componentes horizontal y vertical (H y V)

Salidas:

- Imagen *mod* de la magnitud del gradiente
- Imagen *grad* de la orientación del gradiente

Pasos:

1. Para cada pixel (i, j) en IS, obtener la magnitud y orientación del gradiente basándose en las expresiones para su cálculo:

$$grad(i,j) = atan\left(\frac{V(i,j)}{H(i,j)}\right)$$

$$mod(\mathrm{i},\mathrm{j}) = \sqrt{(H(i,j))^2 + (V(i,j))^2}$$

2. Obtener *mod* a partir de la magnitud del gradiente y *grad* a partir de la orientación, de acuerdo a las expresiones anteriores.

Algoritmo: Supresión no máxima

Entradas:

- Imagen *mod* de la magnitud del gradiente
- Imagen *grad* de la orientación del gradiente

Salidas:

- Imagen sectorizada *Ngrad*
- Imagen con máximos suprimidos *Nmod*

Consideraciones:

Ocho direcciones d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8 identificadas por las orientaciones de 0º, 45º, 90º, 135º, 180º, 225º, 270º, 315º (en radianes, 0, π/4, π/2, 3π/4, π, 5π/4, 3π/2, 7π/4) con respecto al eje horizontal.

Pasos:

1. Para cada pixel (i, j):
   1.1. Encontrar la dirección d_k que mejor se aproxima a la dirección de *Ngrad(i, j)*, que viene a ser la perpendicular al borde.
   1.2. Si *mod(i, j)* es más pequeño que al menos uno de sus dos vecinos en la dirección d_k, al pixel (i, j) de *Nmod(i, j)* se le asigna el valor 0, *Nmod(i, j)* = 0 (supresión), de otro modo *Nmod(i, j)* = *mod(i, j)*.
2. Devolver *Nmod* y *Ngrad*.

Algoritmo: Histéresis de umbral

Entradas:

- Imágenes *Nmod* y *Ngrad* obtenidas del paso anterior
- Umbral t_1
- Umbral t_2, donde $t_1<t_2$

Salidas:

- Imagen de salida O, con los bordes conectados de contornos

Pasos:

1. Para todos los puntos de *Nmod* y explorando *Nmod* en orden fijo:
   1.1. Localizar el siguiente punto de borde no explorado previamente, *Nmod(i, j)*, tal que *Nmod(i, j)*>t_2.
   1.2. Comenzar a partir de *Nmod(i, j)*, seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal de borde, siempre que *Nmod(i, j)*>t_1.
   1.3. Marcar todos los puntos explorados y, salvar todos los puntos en el entorno conectado encontrado.
2. Devolver O formada por el conjunto de bordes conectados de contornos de la imagen, así como la magnitud y orientación, describiendo las propiedades de los punto de borde.

4.3 MAIN

El programa principal es un menú, este menú tiene diferentes opciones, ya sea aplicar el algoritmo de Canny, el algoritmo de Sobel y crear un histograma de una imagen. Este menú está restringido a estas tres cosas ya que lo que se intenta evaluar es el algoritmo de Canny y las diferentes pruebas del mismo.

Es obvio que con un menú restringido como este no se podrán aplicar todas las funcionalidades que pueden ofrecer todas las clases

que contiene el programa. Ni que decir tiene, que cualquier programador con conocimientos en C++ podrá modificar o rehacer un programa principal para sus necesidades ya que el código se encuentra en código abierto.

5 PRUEBAS

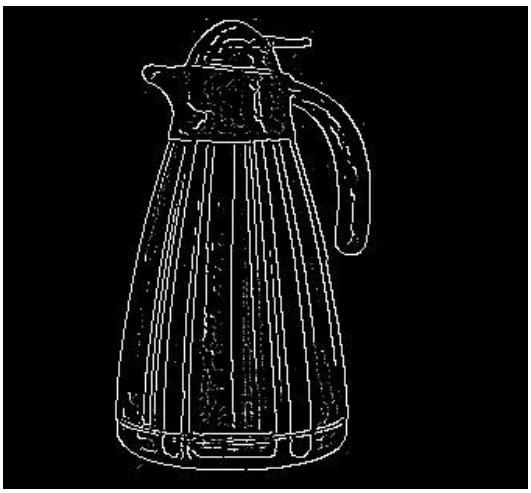
En esta sección se van a mostrar las pruebas realizadas con diferentes imágenes y seguidamente las diferencias con el algoritmo de Sobel, que se ha elegido como muestra de que el algoritmo de Canny obtiene un perfilado de los bordes más claro.

Para cada prueba se han utilizado diferentes umbrales dependiendo de la imagen original, ya que con unos umbrales mal definidos, la imagen resultante con el algoritmo de Canny puede ser poco clara para diferenciar los bordes del objeto principal.

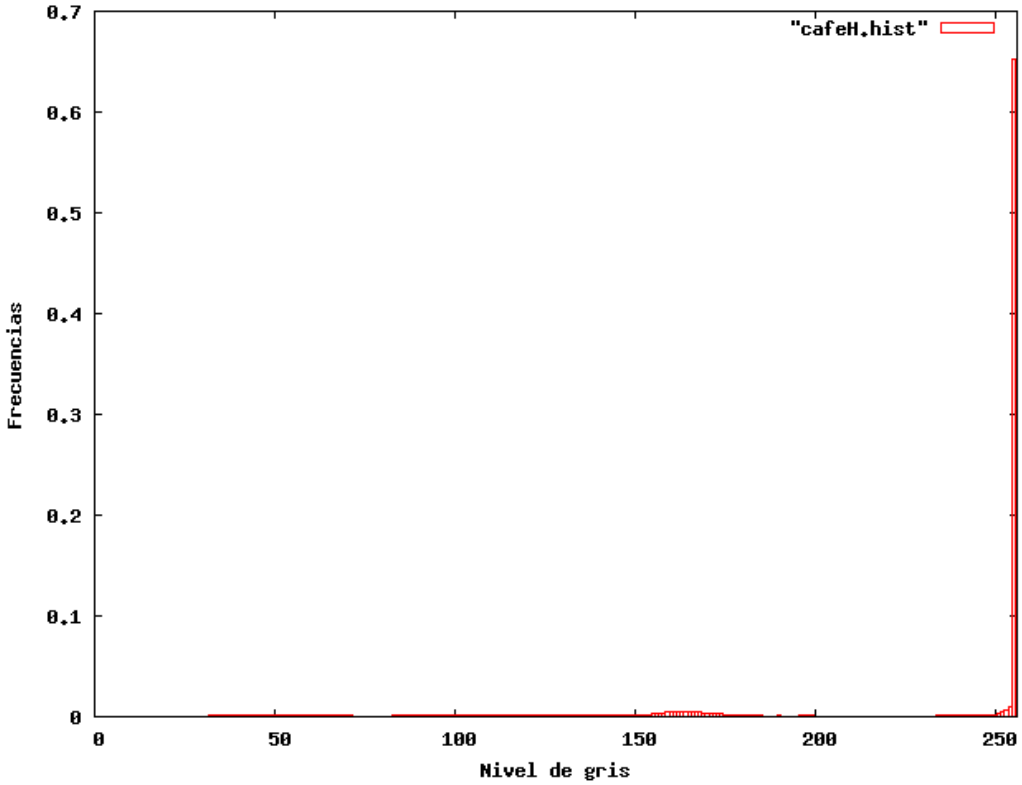
5.1 PRUEBA 1

Suponiendo un umbral bajo del 5% y alto del 65% se puede observar que el algoritmo de Canny pone mucho ruido en la imagen aunque ya se pueden diferenciar bien los bordes del objeto.

Imagen original



Filtro Canny



Histograma original

Figura 6: Prueba 1 – Canny café 65-75

Con un umbral bajo del 65% y alto del 95%, se puede observar que los bordes han disminuido su intensidad y que hay menos ruido en la imagen.

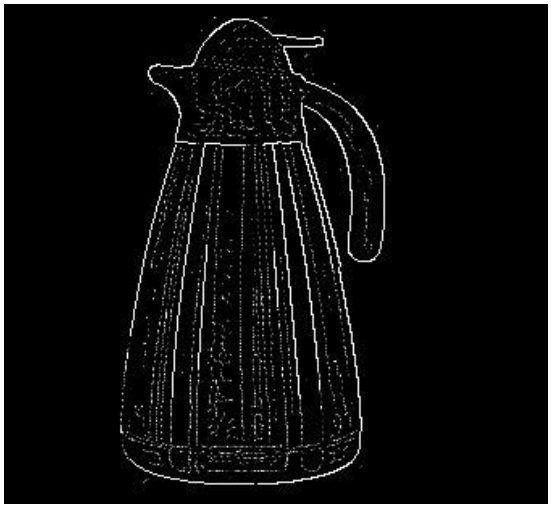


Imagen original Filtro Canny

Figura 7: Prueba 1 – Canny café 65-95

Por último, con un umbral bajo del 75% y un alto del 95%, se puede observar que disminuye considerablemente el ruido de la imagen.

Imagen original Filtro Canny

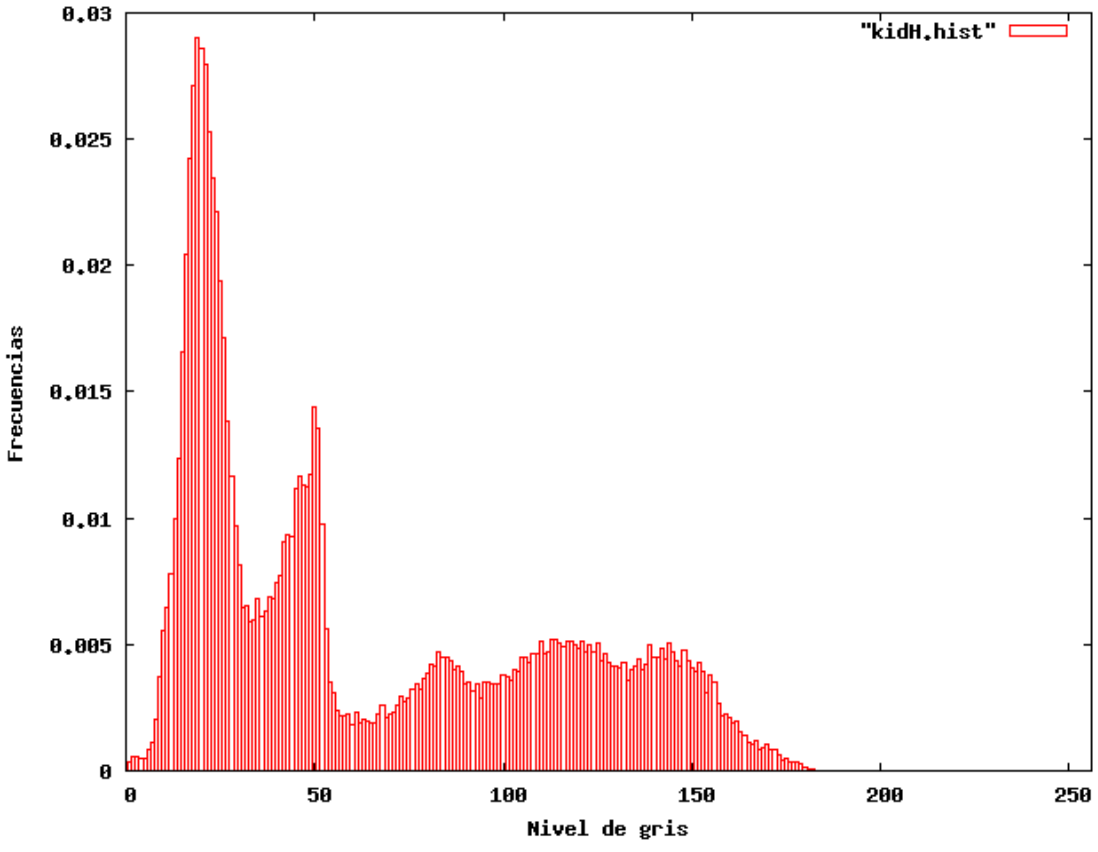
Figura 8: Prueba 1 – Canny café 75-95

5.2 PRUEBA 2

En esta imagen que tiene tantas líneas, con un umbral bajo del 5% y un umbral alto del 65% se puede observar cómo se muestran todas las líneas de los contornos.

Imagen original

Filtro Canny



Histograma original

Figura 9: Prueba 2 – Canny kid 5-65

Con un umbral bajo del 5% y uno alto del 95%, se puede ver que el número de líneas sigue siendo el mismo pero que la intensidad de los bordes disminuye.

Imagen original Filtro Canny

Figura 10: Prueba 2 – Canny kid 5-95

Por último, con un umbral bajo del 75% y uno alto del 95%, se observa que el ruido ya ha desaparecido casi por completo y que los bordes están bien definidos.

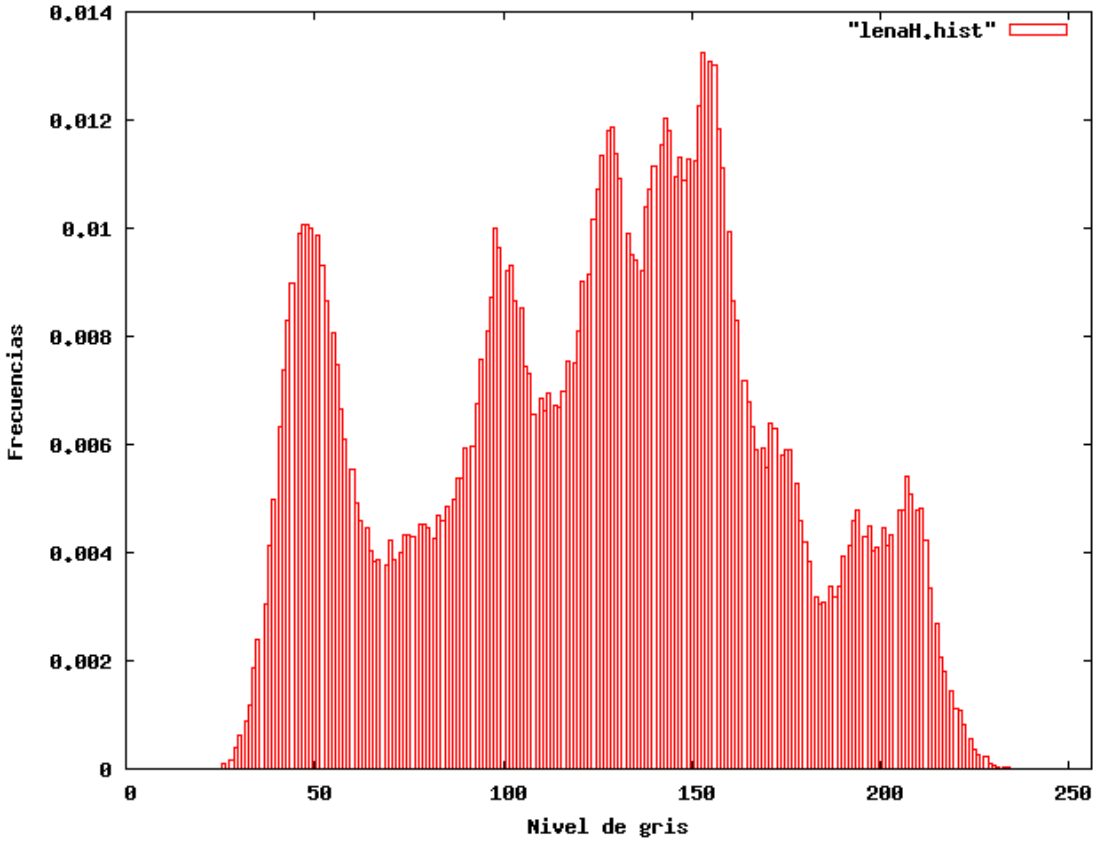
Imagen original Filtro Canny

Figura 11: Prueba 2 – Canny kid 75-95

5.3 PRUEBA 3

En esta imagen lo que más líneas genera es el pelo, por eso podemos observar, que con un umbral bajo del 5% y uno alto del 65%, las líneas del contorno se encuentran definidas aunque existe ruido.

Imagen original Filtro Canny



Histograma original

Figura 12: Prueba 3 – Canny lena 5-65

Con un umbral bajo del 45% y uno alto del 85%, se puede observar que el ruido va desapareciendo, pero existe perdida de líneas del contorno.

Imagen original Filtro Canny

Figura 13: Prueba 3 – Canny lena 45-85

Por último, con un umbral bajo del 55% y uno alto del 95%, se puede observar que ya existe una pérdida de líneas del contorno bastante grande.

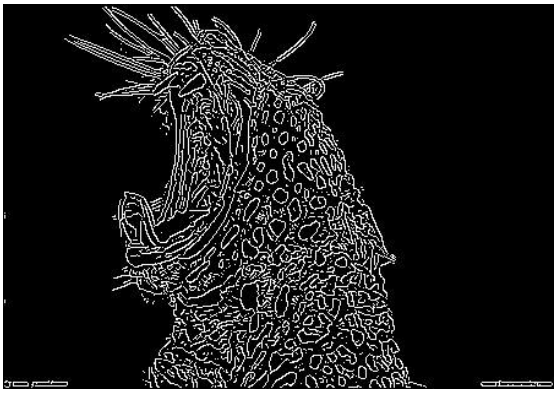
Imagen original Filtro Canny

Figura 14: Prueba 3 – Canny lena 55-95

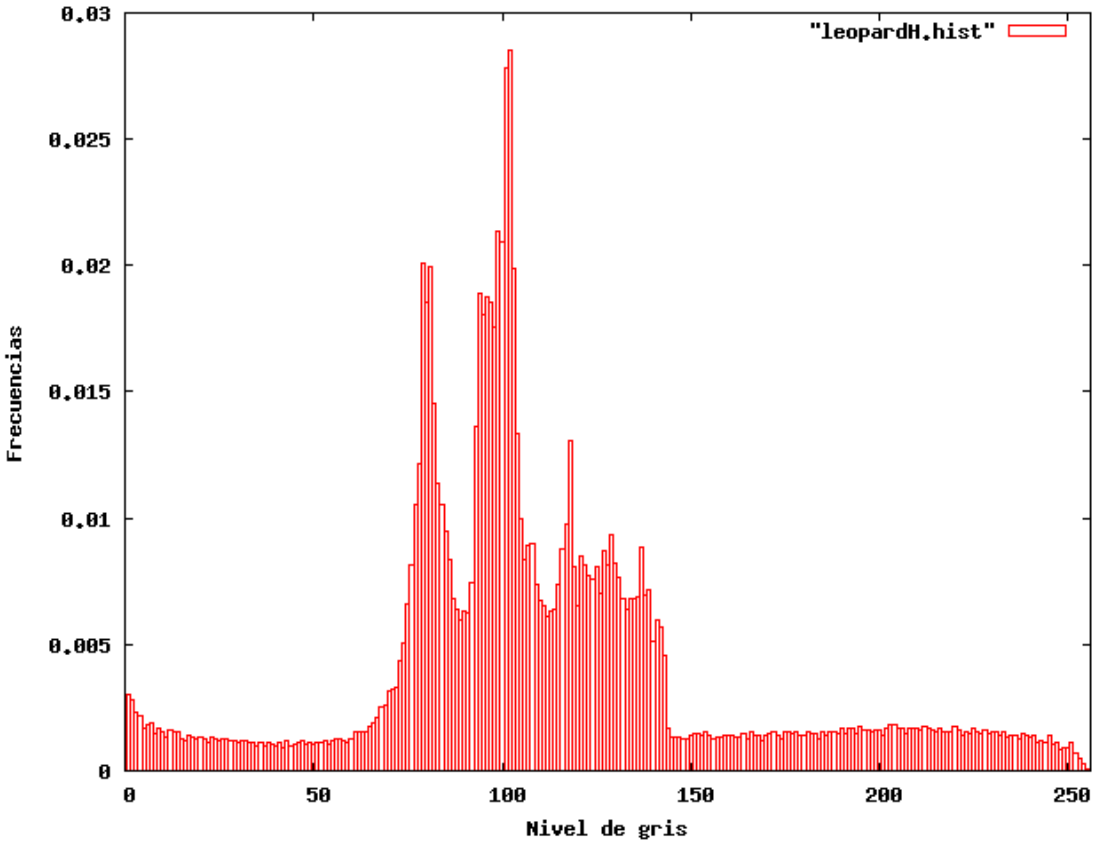
5.4 PRUEBA 4

En esta imagen lo más significativo que tiene es las manchas del leopardo, con un umbral bajo del 5% y uno alto del 65%, se puede apreciar ruido y los contornos bien definidos.

Imagen original



Filtro Canny



Histograma original

Figura 15: Prueba 4 – Canny leopard 5-65

Con un umbral bajo del 45% y alto del 75%, se puede observar cómo se elimina algo de ruido quedando mejor definidos los contornos de las manchas.

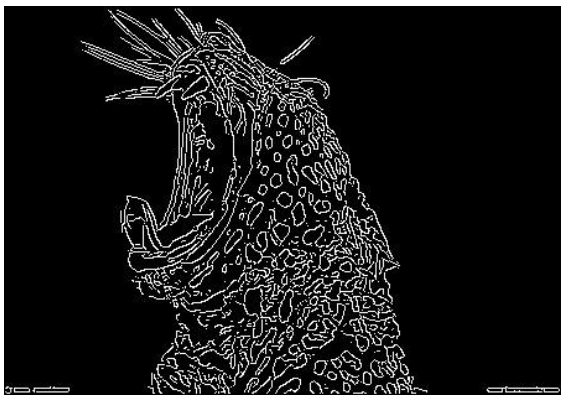



Imagen original Filtro Canny

Figura 16: Prueba 4 – Canny leopard 45-75

Por último, con un umbral bajo del 55% y uno alto del 95%, se puede observar como ya se pierde información de los contornos de la imagen.


Imagen original Filtro Canny

Figura 17: Prueba 4 – Canny leopard 55-95

5.5 PRUEBA 5

Esta imagen tiene bastantes contornos, pero lo que más puede interesar son las letras, con un umbral bajo del 5% y alto del 65%, se puede observar que hay mucho ruido, pero aun así ya se puede observar los contornos de las letras.

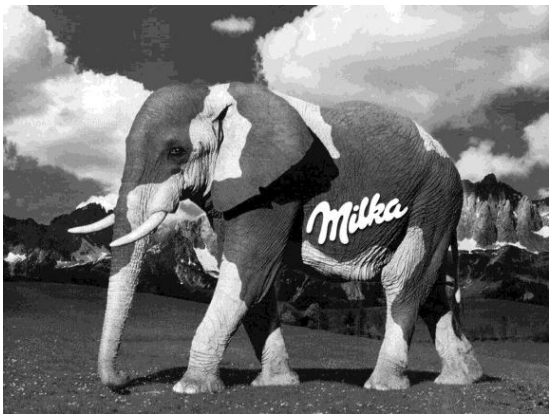
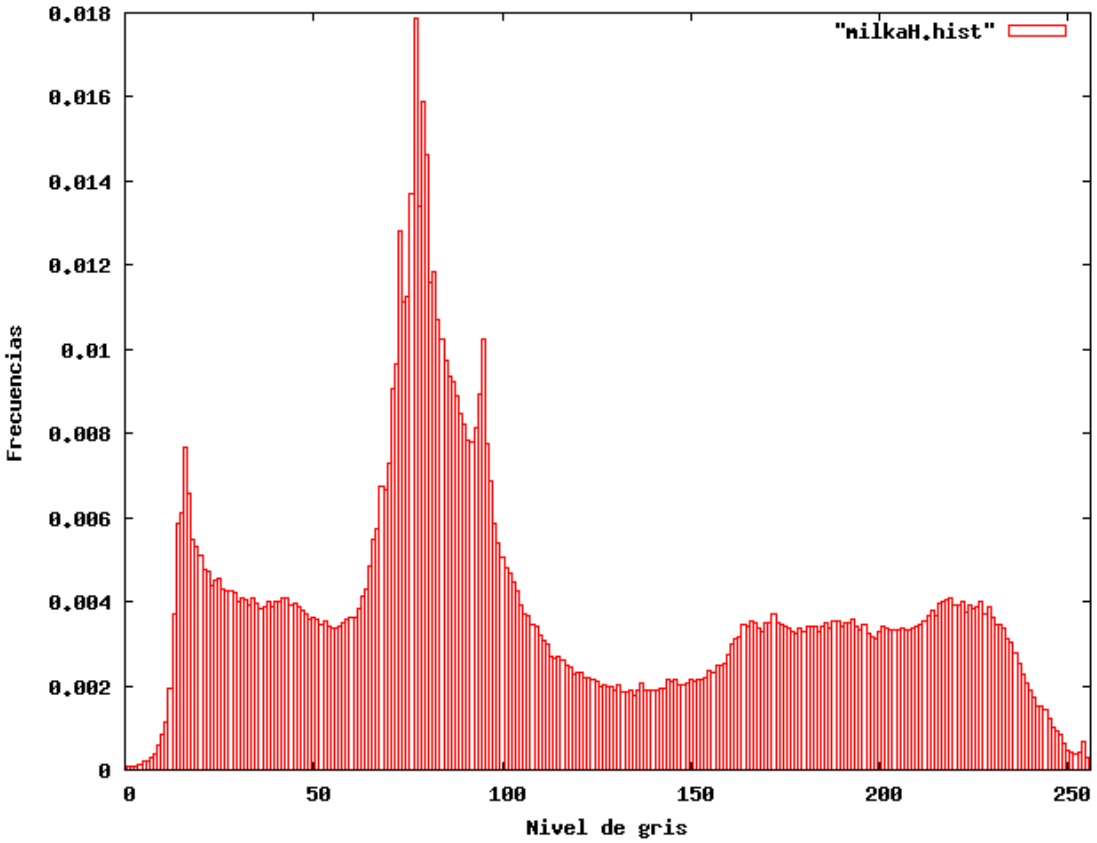


Imagen original

Filtro Canny



Histograma original

Figura 18: Prueba 5 – Canny milka 5-65

Con un umbral bajo del 45% y uno alto del 85%, se puede observar que la imagen ya tiene menos ruido, aunque todavía es excesivo, los contornos de las letras se encuentran más claros.

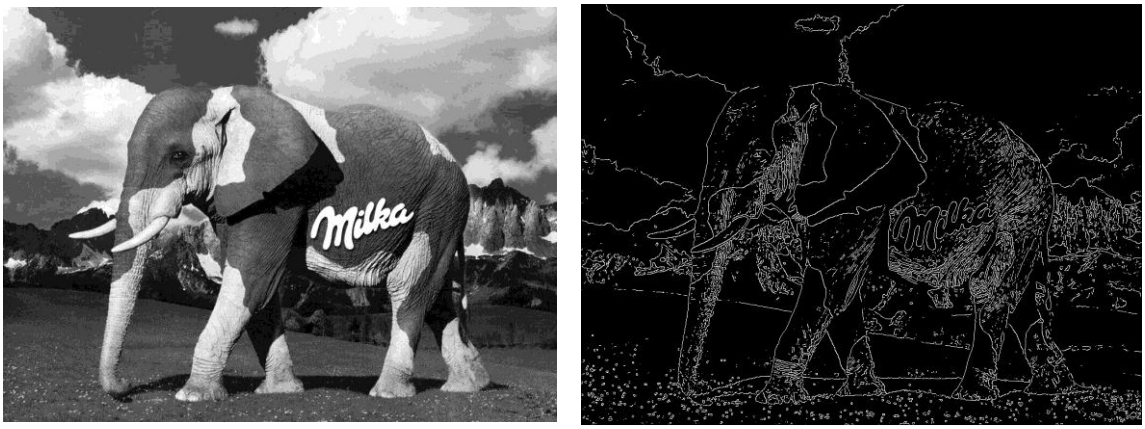


Imagen original Filtro Canny

Figura 19: Prueba 5 – Canny milka 45-85

Por último, con un umbral bajo del 55% y uno bajo del 95%, se puede observar que aunque ya el ruido es bastante poco, se ha perdido bastante información de los contornos, eso sí, si lo que se pretendía es un buen reconocimiento del contorno de las letras ahora se pueden observar con total claridad.

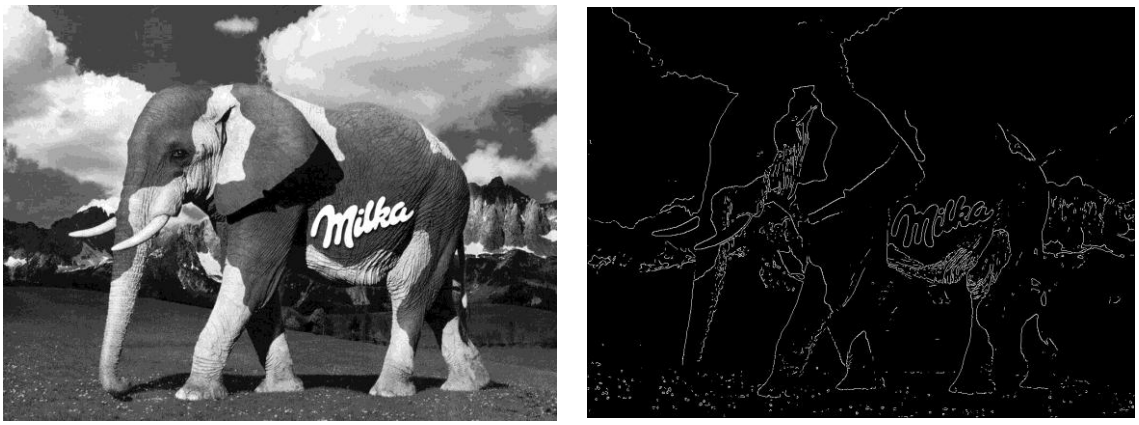


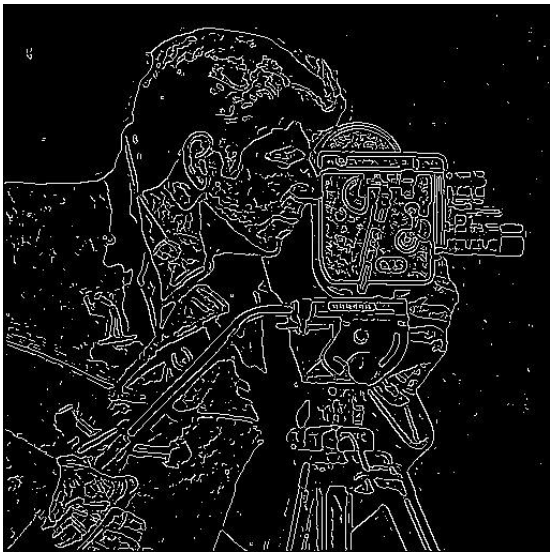
Imagen original Filtro Canny

Figura 20: Prueba 5 – Canny milka 55-95

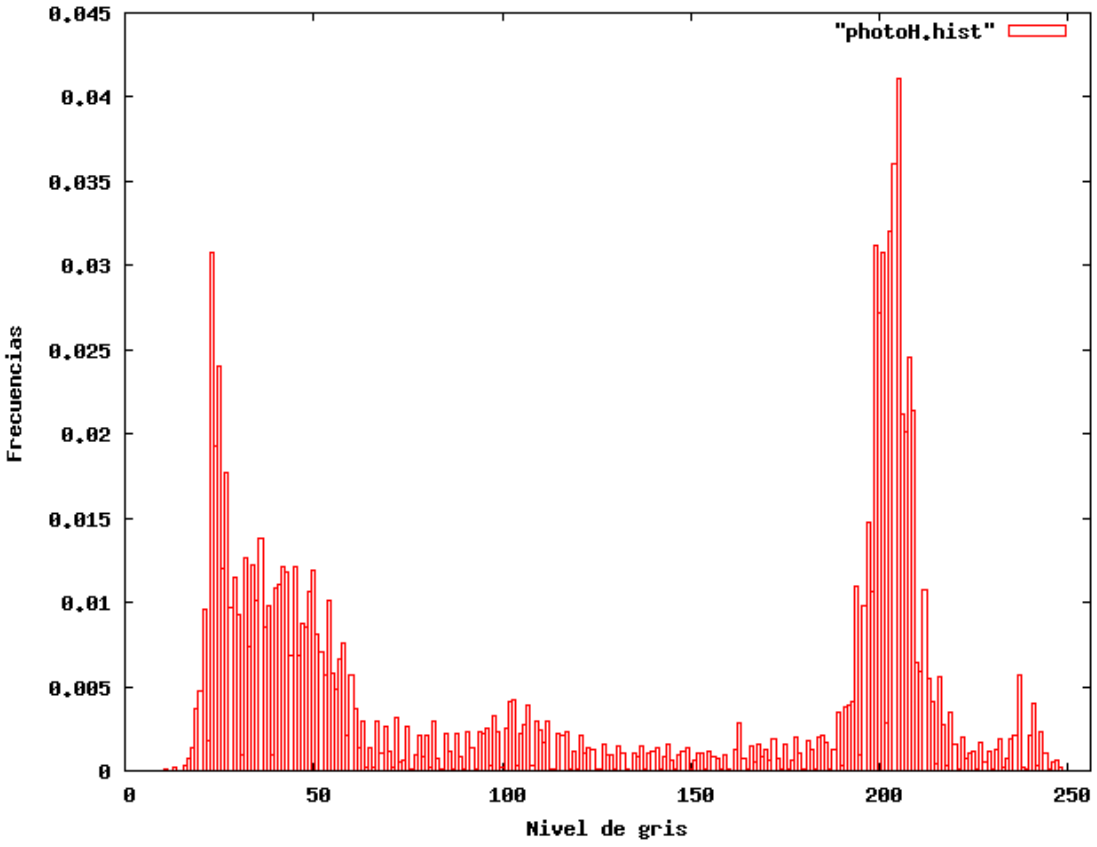
5.6 PRUEBA 6

Esta imagen es buena para ver los contornos, con un umbral bajo del 5% y uno alto del 65%, se puede observar que el contorno ya está definido aunque tiene ruido.

Imagen original



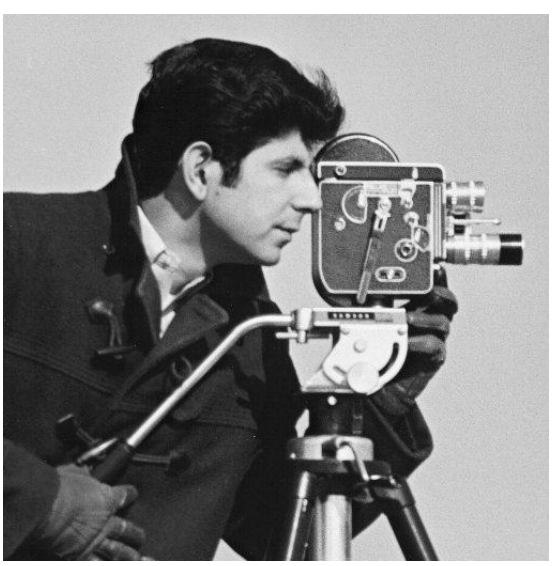
Filtro Canny



Histograma original

Figura 21: Prueba 6 – Canny photo 5-65

Con un umbral bajo del 45% y uno alto del 85%, se pude observar como el contorno se encuentra perfectamente definido y el ruido que existe es prácticamente ínfimo.



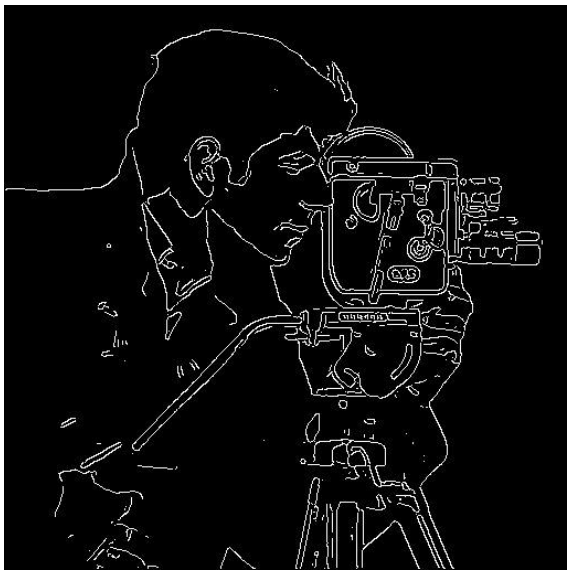
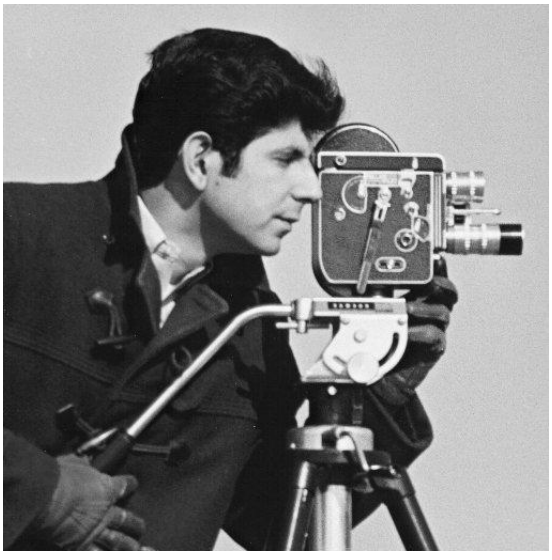


Imagen original Filtro Canny

Figura 22: Prueba 6 – Canny photo 45-85

Por último, con un umbral bajo del 55% y uno alto del 95%, se puede observar como existe el contorno aunque con pérdida de información.



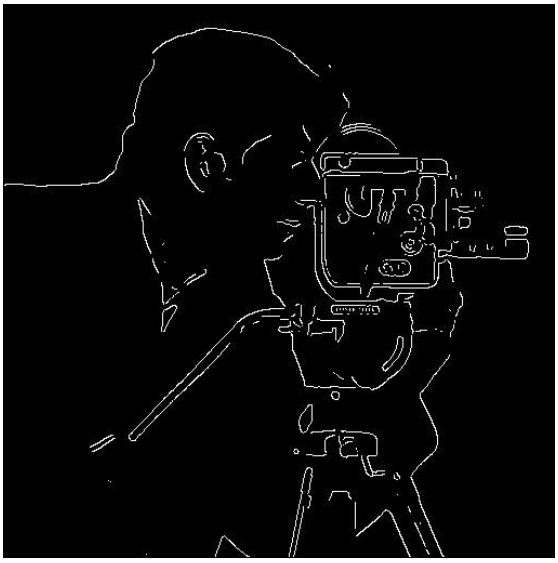
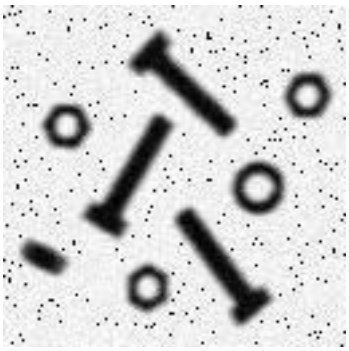


Imagen original Filtro Canny

Figura 23: Prueba 6 – Canny photo 55-95

5.7 PRUEBA 7

Esta imagen es muy buena para reconocimiento de contornos, son objetos en un color completamente diferente al del fondo, con un umbral bajo del 5% y uno alto del 65%, se puede observar como los contornos están definidos pero existe un nivel de ruido bastante grande.



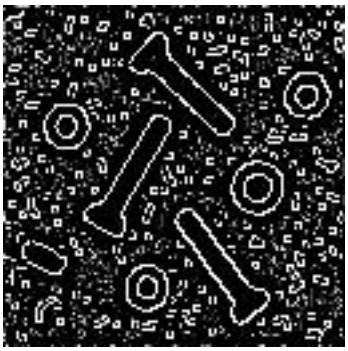
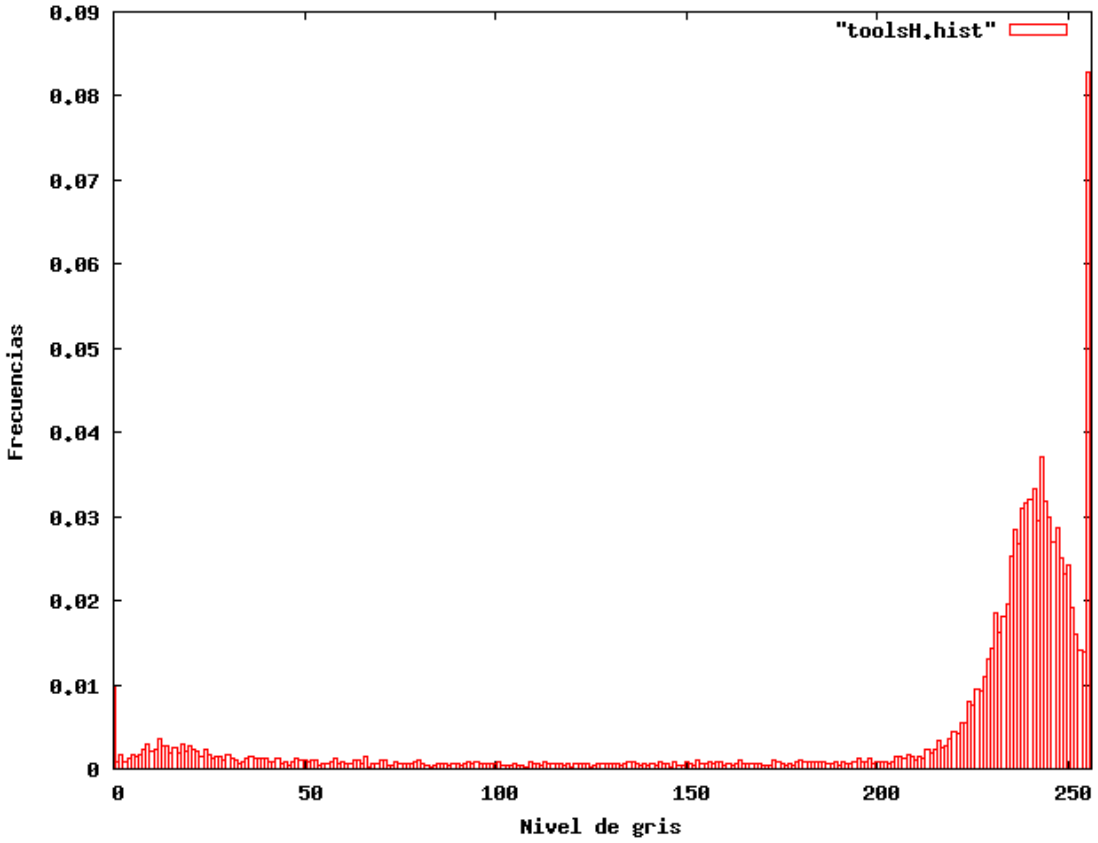


Imagen original Filtro Canny



Histograma original

Figura 24: Prueba 7 – Canny tools 5-65

Con un umbral bajo del 55% y uno alto del 95%, se puede observar como los contornos, aunque definidos, tienen algo de pérdida, existe ruido de los pizcos de la imagen.

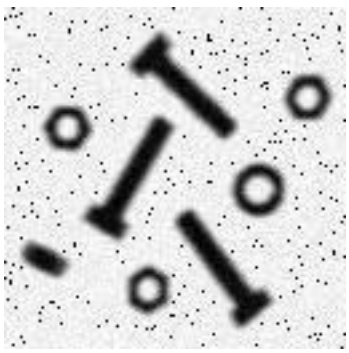
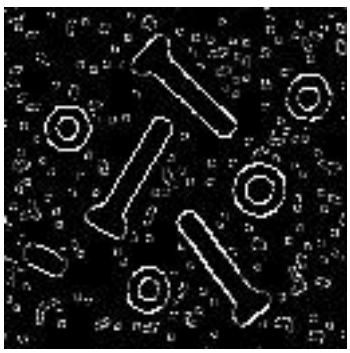


Imagen original



Filtro Canny

Figura 25: Prueba 7 – Canny tools 55-95

Por último, con un umbral bajo del 85% y uno alto del 95%, se puede observar como los contornos están perfectamente definidos, el ruido es prácticamente nulo.

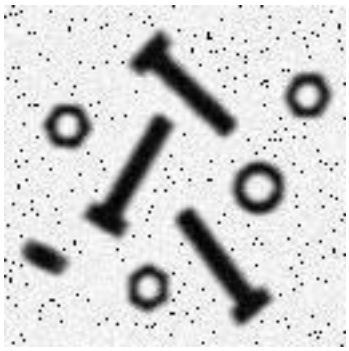
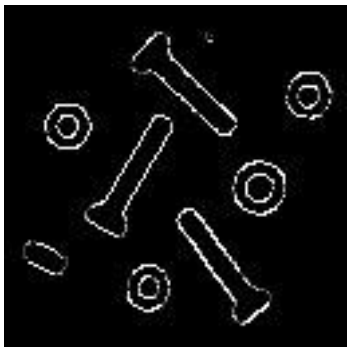


Imagen original



Filtro Canny

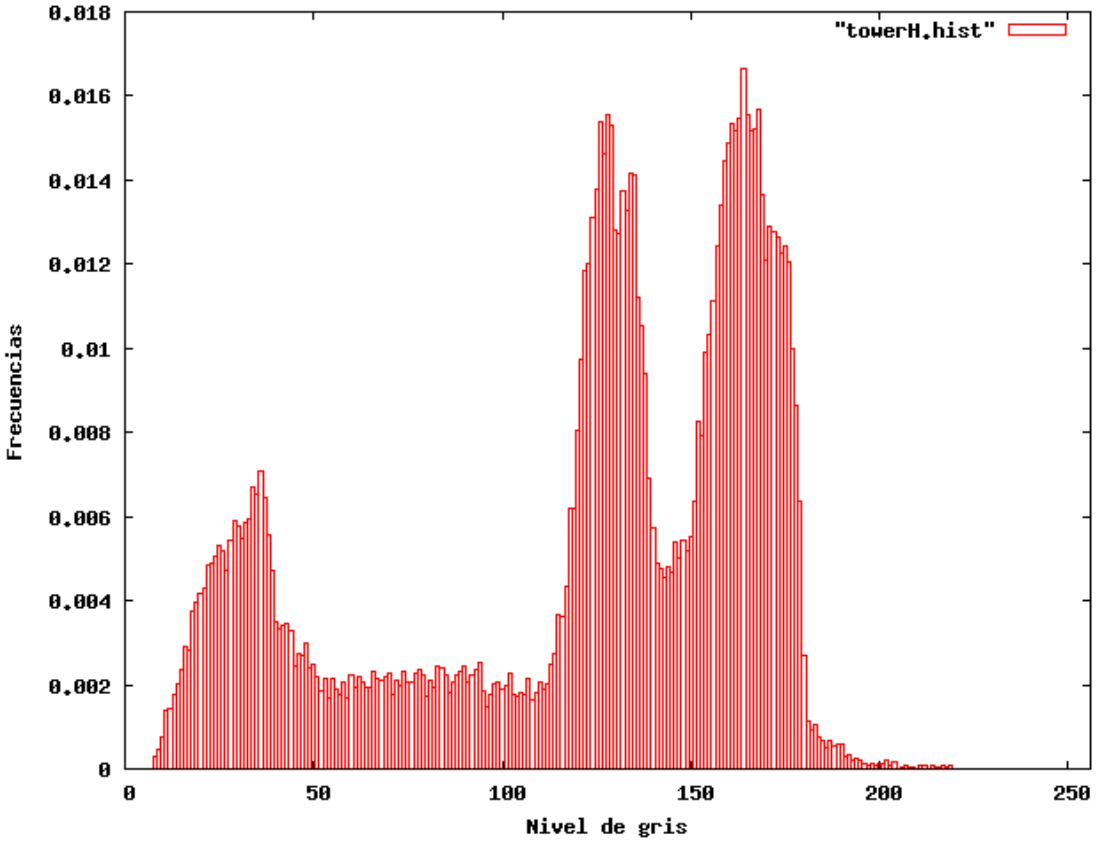
Figura 26: Prueba 7 – Canny tools 85-95

5.8 PRUEBA 8

En esta imagen existen zonas de sombra y zonas iluminadas, con un umbral bajo del 5% y uno alto del 65%, se puede observar que la parte superior se encuentra llena de ruido, los contornos están definidos.

Imagen original

Filtro Canny



Histograma original

Figura 27: Prueba 8 – Canny tower 5-65

Con un umbral bajo del 55% y uno alto del 65%, se puede observar como el ruido es bastante bajo y los contornos están bastante bien definidos.

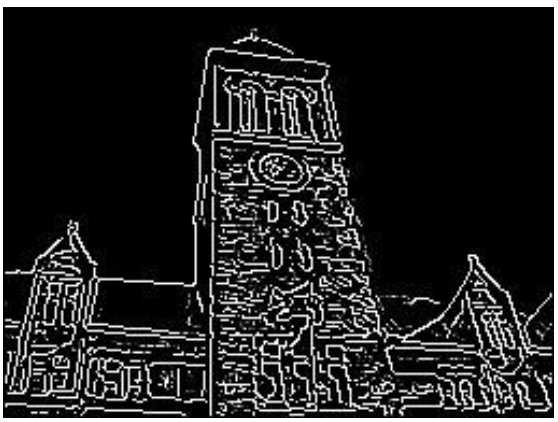


Imagen original Filtro Canny

Figura 28: Prueba 8 – Canny tower 55-65

Por último, con un umbral bajo del 55% y uno alto del 95%, se puede observar como el ruido es bastante bajo pero existe perdida de nitidez de los contornos, ahora son más débiles.

Imagen original

Filtro Canny

Figura 29: Prueba 8 – Canny tower 55-95

5.9 PRUEBA 9

Esta imagen también es bastante buena para el reconocimiento de contornos, ya que tiene un objeto principal de un color con un fondo de otro color; con un umbral bajo del 5% y un umbral alto del 65%, se puede observar que los contornos están definidos pero el ruido es enorme.

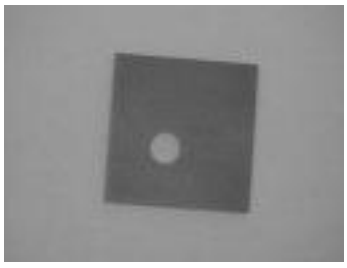
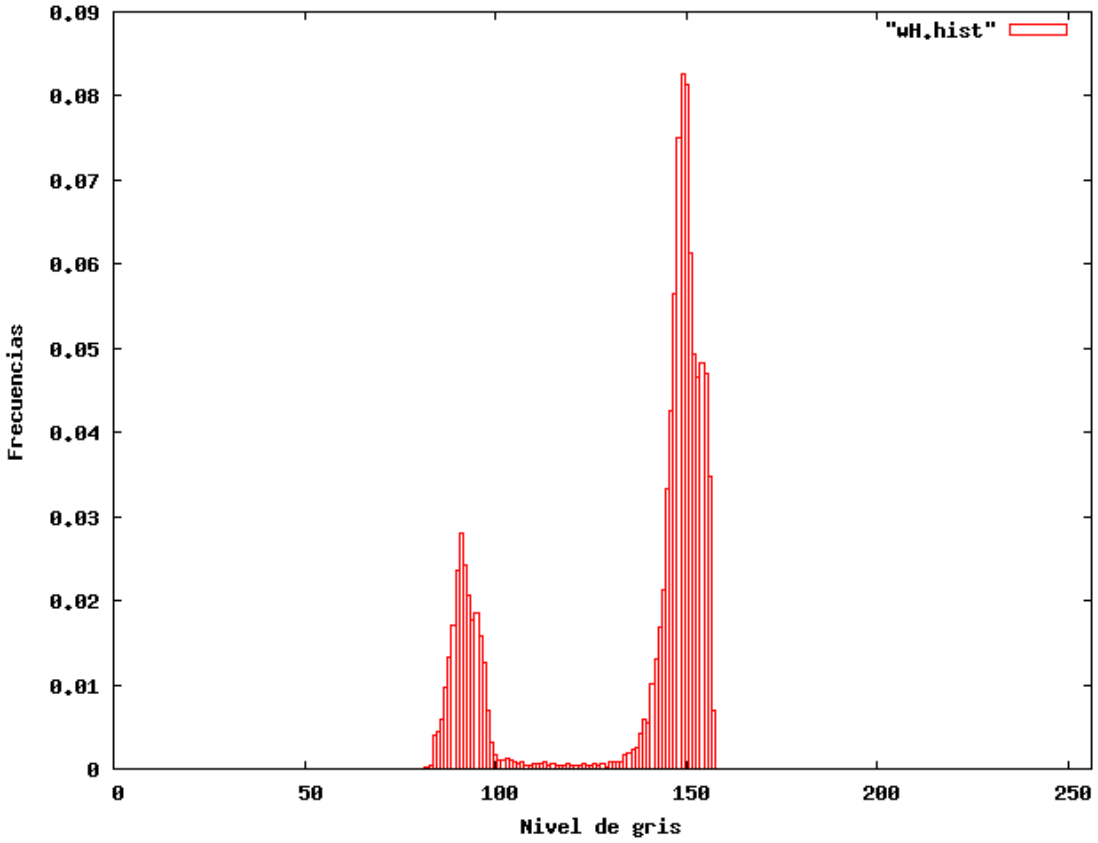


Imagen original

Filtro Canny



Histograma original

Figura 30: Prueba 8 – Canny w 5-65

Con un umbral bajo del 5% y uno alto del 95%, se puede observar como la intensidad de las líneas es menor, el ruido aún es muy alto.

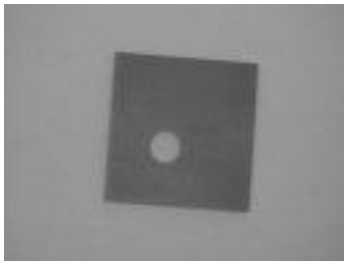
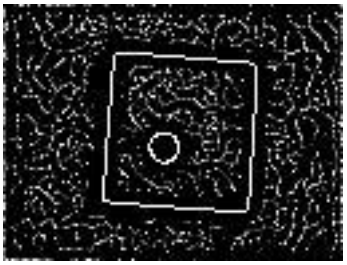


Imagen original



Filtro Canny

Figura 31: Prueba 8 – Canny w 5-95

Por último, con un umbral bajo del 85% y uno alto del 95%, se puede observar como la imagen carece ya de ruido y que el contorno del objeto está claramente definido.

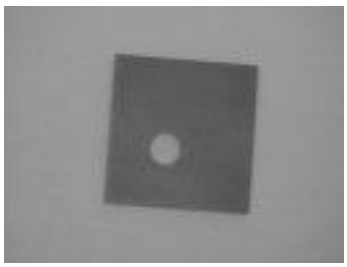
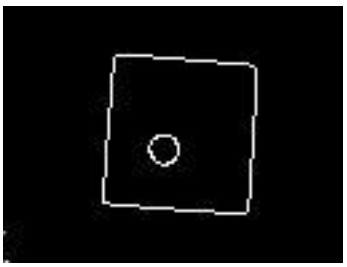


Imagen original



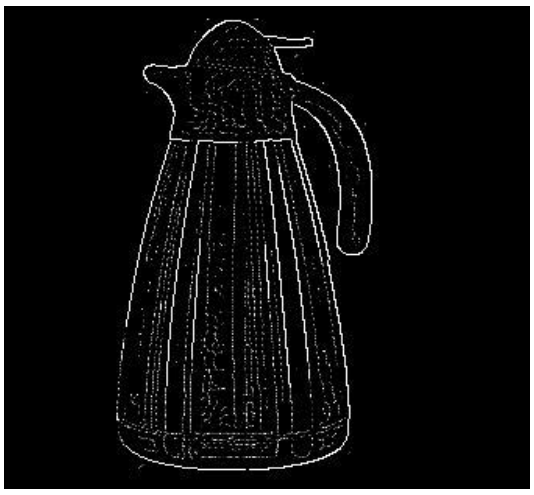
Filtro Canny

Figura 32: Prueba 8 – Canny w 85-95

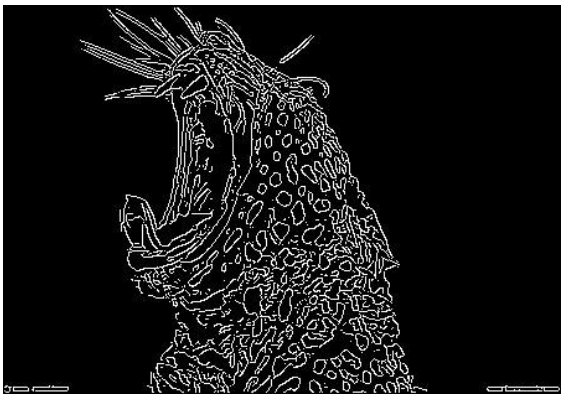
5.10 Diferencias con Otros Algoritmos

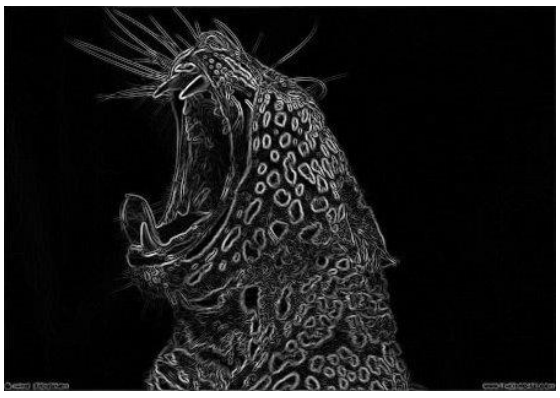
Aquí podemos observar las diferencias entre el algoritmo de Canny y el de Sobel, se podría comparar con algún otro algoritmo, pero ya que, en este caso, digamos que se utiliza uno dentro del otro, es fácil de observar las diferencias, siempre y cuando se hayan elegido unos umbrales correctos.

Canny



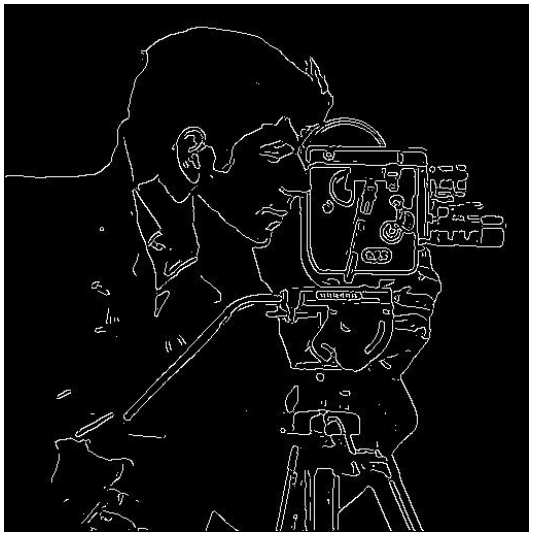
Sobel

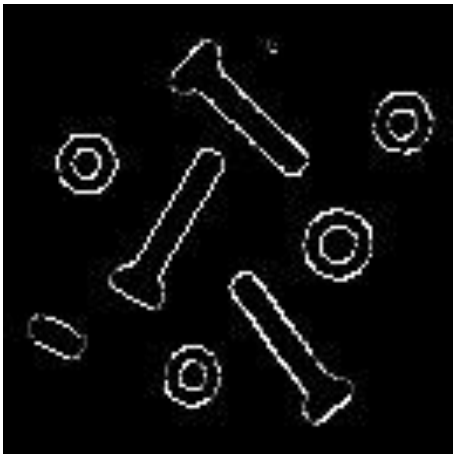


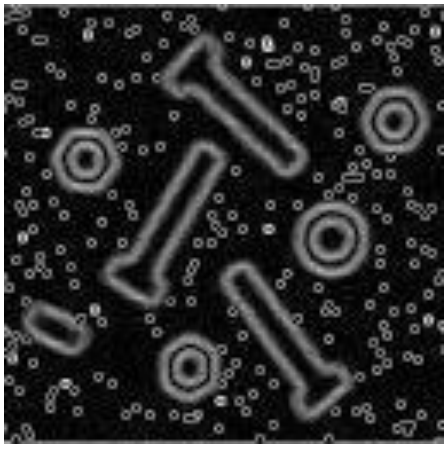



Milka


Milka







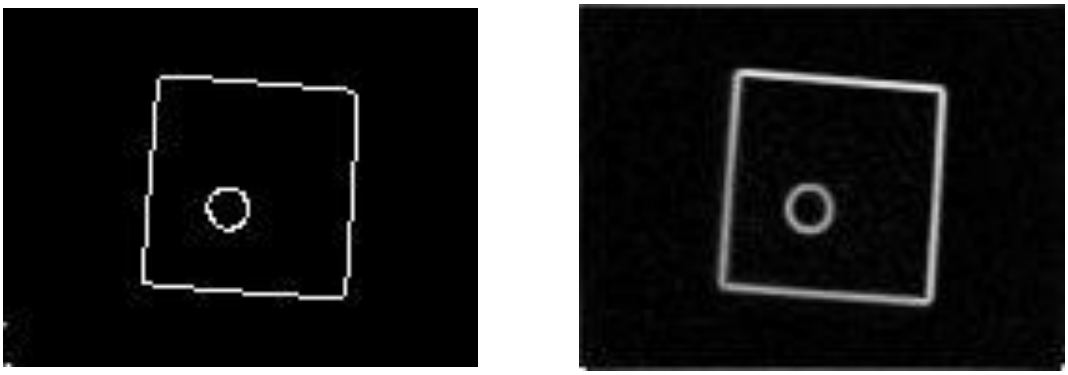


Figura 33: Diferencias Canny - Sobel

6 CONCLUSIONES

Como hemos podido observar a lo largo del documento, Canny es un algoritmo para la detección de bordes bastante bueno, la única pega que tiene es los umbrales; elegir unos umbrales malos puede hacer que la imagen resultante no sea la deseada, en cambio, unos umbrales buenos puede hacer que el reconocimiento sea genial.

Con respecto a las diferencias con otros algoritmos, en este caso la diferencia con Sobel, cada uno tiene sus defectos y sus virtudes según para lo que vayan a ser utilizados. Así, dependiendo de la finalidad que tenga el reconocimiento uno será más útil que otro, y así también con el resto de algoritmos que no se han tratado aquí.

6.1 APORTACIONES O POSIBLES MODIFICACIONES

Como es obvio, este programa es mejorable en todos los aspectos, se podría desde optimizar la memoria hasta la simplificación de código.

También se pueden hacer varias aportaciones como añadir la implementación de más algoritmos para tratamiento de visión digital, crear un programa principal más completo o en su caso para un uso diferente al que se le da aquí, o hacer más pruebas para una mayor fiabilidad del código.

A BIBLIOGRAFÍA

[1.] [**UCO**] Universidad de Córdoba, http://www.uco.es, (Enero, 2009).

[2.] [**UCM**] Plataforma Moodle Universidad de Córdoba, http://www3.uco.es/moodle, (Enero, 2009).

[3.] [**PLE**] Prácticas de laboratorio para estudiantes de ingeniería con FPGA, A. Castillo, J. Vázquez, J. Ortegón y C. Rodríguez, http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol6/vol6issue2June2008/6TLA2_02CastilloAtoche.pdf, (Enero, 2009).

[4.] [**API**] Adquisición y procesamiento de imágenes estereoscópicas y modelado de mundos 3D para su implementación en exploración de ambientes, José Antonio Amador González, http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/amador_g_ja/, (Enero, 2009).

[5.] [**VPC**] VISIÓN POR COMPUTADOR, Universidad Miguel Hernández, en especial el tema 6 de "Detección de bordes en una imagen", http://isa.umh.es/asignaturas/crss/temasvision/, (Enero, 2009).

[6.] [**DEB**] Detección de Bordes: Algoritmo de Canny, Jorge Valverde Rebaza, http://jc-info.blogspot.com/2007/06/deteccin-de-bordes-algoritmo-de-canny.html, (Enero, 2009).

[7.] [**DBC**] Detección de bordes mediante el algoritmo de Canny, Jorge Valverde Rebaza, Universidad Nacional de Trujillo, http://www.seccperu.org/files/Detecci%C3%B3ndeBordes-Canny.pdf, (Enero, 2009).

[8.] [**SRA**] Sistema de Reconocimiento Automático de Formas, Diego A. Macrini y Guillermo Baruh, en especial la página 6, http://www.cs.toronto.edu/~dmac/images/ProjectFiles/sraf/srafdoc/sraf.html, (Enero, 2009).

[9.] [**PDV**] Utilización de FPGAs para el procesamiento digital de video, J. Javier Martínez, F. Javier Toledo, F. Javier Garrigós, J. Manuel Ferrández, Universidad Politécnica de Cartagena, http://repositorio.bib.upct.es/dspace/bitstream/10317/397/1/2006_AI_18.pdf, (Enero, 2009).

[10.] [**CAT**] Canny Tutorial, Noah Kuntz, http://www.pages.drexel.edu/~nk752/cannyTut2.html#Step%201, (Enero, 2009).

[11.] [**EDT**] Canny Edge Detection Tutorial, Bill Green, http://www.pages.drexel.edu/~weg22/can_tut.html, (Enero, 2009).

[12.] [**CED**] Canny Edge Detection, Simon Strandgaard, http://www.opcoders.com/articles_about_graphics/canny_edge_detection.html#nonmax, (Enero, 2009).