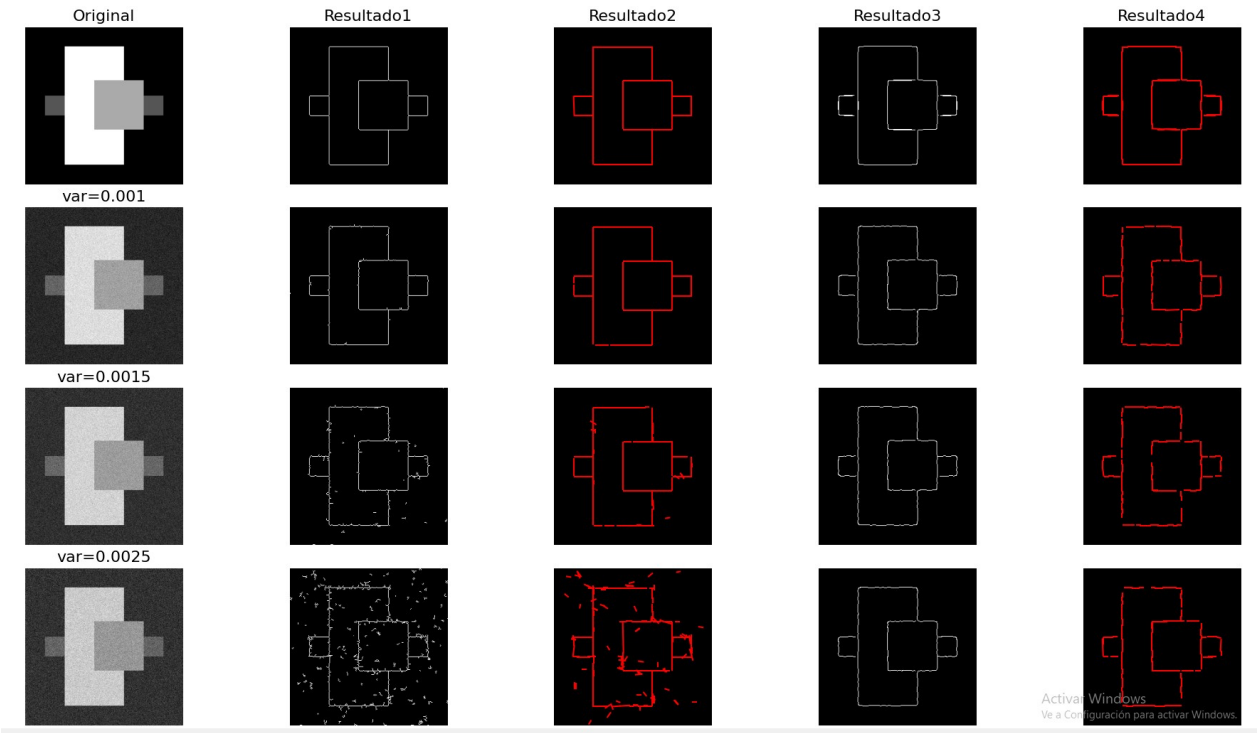
DETECTORES DE BORDES Y CONTORNOS

Parte 1

En este ejercicio hemos generado a partir de la imagen “cuadrados.png” tres imágenes con ruido gaussiano de varianzas 0.001 ,0.0015 y 0.0025.

Para cada una de ellas hemos obtenido su gradiente mediante el operador de Sobel y hemos generado su mapa de bordes con la función *apply_hysteresis_threshold.*Además,hemos refinado el mapa de bordes obtenido con la función *thin* de forma que tenga el grosor de un píxel. A esto le hemos llamado **resultado 1.** A la obtención de los segmentos de sus contornos mediante la transformada de Hough probabilística y progresiva la hemos llamado r**esultado 2.**

A continuación, hemos obtenido el mapa de bordes de cada imagen mediante el detector de *Canny* y lo hemos llamado **resultado 3** y a partir del mapa de bordes hemos obtenido de la misma forma que en el resultado 2 los segmentos correspondietes a sus contornos a lo que hemos llamado **resultado 4**. Finalmente,hemos obtenido lo siguiente:



Ahora explicaremos los parámetros que hemos tenido que indicar en las funciones comentadas anteriormente.

La función *apply_hysteresis_threshold* que es un algoritmo que encuentra la región del gradiente de la imagen donde la imagen es mayor que high o es mayor que low y esa región está conectada a una región mayor que high. Por lo tanto,se le debe pasar como parámetros el gradiente de la imagen,el valor high y el valor low. Nosotros hemos utilizado como valor high el 20% del valor máximo del gradiente y como low el 10% del máximo del gradiente.

La función *canny* es un algoritmo que devuelve el mapa de bordes de una imagen.Recibe como parámetros una imagen y sigma que es la desviación estándar del filtro Gussiano.Nosotros hemos utilizado sigma=2.

La función *probabilistic_hough_line* devuelve las líneas a partir de una transformada de Hough probabilística progresiva. Más concretamente los puntos de inicio y final de los segmentos encontrados.Para ello le tenemos que pasar como parámetros el umbral, la longitud mínima de los segmentos detectados que queremos guardar y la distancia máxima entre píxeles para que aún formen una línea. Nosotros hemos utilizado como umbral 10, como longitud de segmento 5 y como tercer parámetro 3.

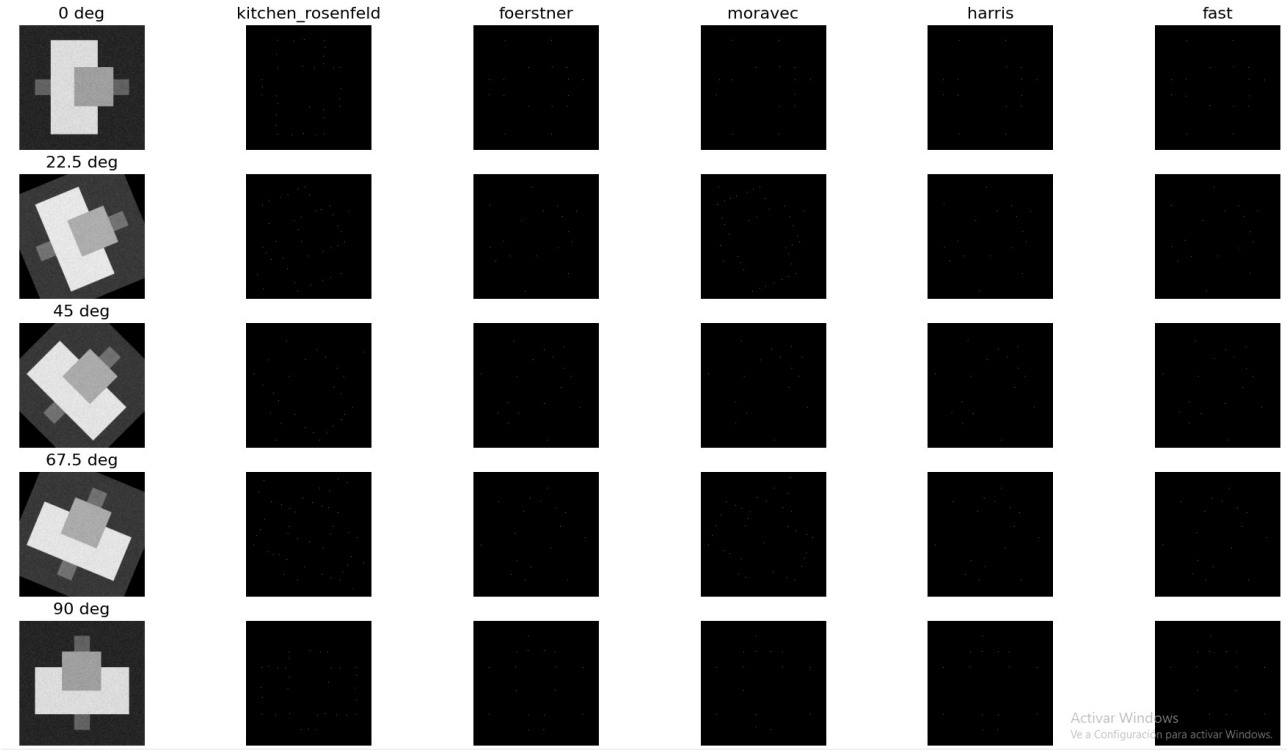
En cuanto a los resultados obtenidos, los contornos obtenidos mediante Sobel son más fieles a la realidad si no hay ruido, pero al introducir ruido aparecen contornos donde no los hay en forma de pequeños segmentos. A diferencia del Sobel, el algoritmo Canny presenta unos resultados buenos a la hora de detectar los contornos sin ruido, y con ruido su rendimiendo no cambia significativamente. Por tanto, a la hora de elegir un algoritmo de detección de bordes en imágenes con ruido eligiría el algoritmo Canny.

Parte 2

En este ejercicio hemos aplicado ruido gaussiano con una variación de 0.001 a la imagen *cuadros.png*. A continuación, hemos aplicado sobre ella los detectores de esquinas siguientes: **Kitchen&Rosenfeld, Foerstner, Moravec, Harris y Fast**. En cada caso hemos utilizado un umbral diferente para que con todos los detectores se pueda detectar correctamente las esquinas de la imagen ruidosa. Estos umbrales son los siguientes:

- Para Kitchen&Rosenfeld: 0.3
- Para Foerstner: 0.08
- Para Moravec: 0.08
- Para Harris: 0.01
- Para Fast : 0.1

Después de encontrar los umbrales más adecuados para cada caso,hemos girado la imagen varios ángulos para ver si la detección de esquinas se ve afectada por el giro de la imagen original. Los resultados obtenidos son los siguientes:



Como conclusión, la detección de bordes se ve afectada al girar la imagen original. Especialmente en los algoritmos Kitchen & Rosenfeld y Fast. En cambio, los demás han sufrido pequeñas variaciones en la detección de esquinas al ser rotada la imagen , de una o dos esquinas no detectadas pero sin detectar esquinas donde no las hay.

Parte 3

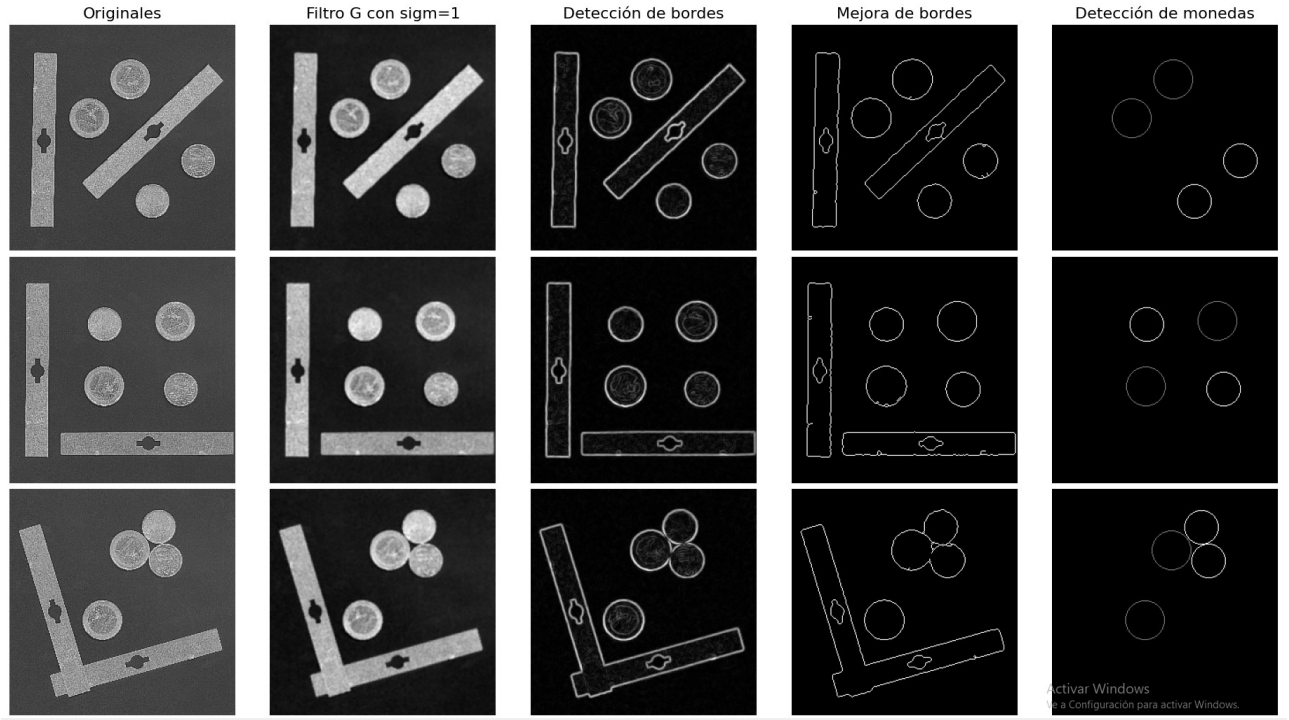
El objetivo de este ejercicio es detectar varías monedas de 10 céntimos y de 1 euro e identificar cual es cual a partir de su radio. Para realizar dicha detección y clasificación,en primer lugar hemos aplicado un filtro gaussiano a todas las imagenes que contienen monedas para eliminar el ruido de ellas.A continuación, para detectar su contorno hemos utilizado el filtro de Sobel para encontrar sus gradientes y después hemos generado su mapa de bordes utilizando la función *apply_hysteresis_threshold* con un valor high=50% de su máximo valor del gradiente y un valor min= 20% del valor máximo del gradiente. Cabe destacar que este borde lo hemos refinado de forma que tenga un grosor de un píxel mediante la función *thin.*

Una vez detectados sus bordes, hemos calculado los píxeles a los que equivaldría el radio de ambas monedas.Al saber que se ha utilizado para las imagenes una resolución de 50 píxeles por pulgada,y el radio de la moneda de un euro es de 23/2 mm y la de 10 céntimos es de 19.5/2 mm, aplicando la siguiente fórmula hemos obtenido los píxeles a los que equivaldría ambos radios:

radio en píxeles=((diametro en mm*píxeles_por_pulgada)/25.4)/2

Con esta fórmula, el radio de la moneda de 10 céntimos es aproximadamente 19 píxeles y la de 1 euro de aproximadamente 22 píxeles.

Finalmente, hemos utilizado la transformada de Hough para la detección de círculos, a la que le hemos pasado como parámetros los radios comprendidos entre 22 y 19. Esta función nos devuelve el centro de los círculos detectados y su radio. A partir de su radio, podemos clasificarlas en monedas de 10 centimos y un euro , y mediante las coordenadas de su centro podemos dibujar círculos donde realmente se encuentran las monedas en la imagen. El resultado final obtenido es el siguiente:



Como podemos observar, se clasifican correctamente las monedas en todas las imágenes.