Reto 4 - Detección Canny Edge y Detección Via Templates

(Miércoles 14 Febrero)

4.1. Detección Canny Edge

Teoria Canny Edge

El algoritmo de Canny es un popular método de detección de bordes, desarrollado por John F. Canny en 1986. El algoritmo consta de múltiples etapas, cada una de las cuales se explicarán a continuación.

1. **Reducción de ruido**

Debido a que la detección de bordes es un proceso susceptible al ruido en la imagen, el primer paso es eliminarlo o reducirlo lo más que se pueda. Para esto, el algoritmo utiliza un filtro Gaussiano 5×5.

2. **Encontrar gradiente de intensidad de la imagen**

Una vez que la imagen ha sido alisada con el filtro Gaussiano, se calcula el gradiente* de la misma. Para esto la imagen se filtra nuevamente, esta vez utilizando un kernel Sobel en la dirección horizontal (Gx) y la dirección vertical (Gy). A partir de estas dos imágenes, resultado de aplicar ambos kernels Gx y Gy, se pueden encontrar los bordes y la dirección del gradiente en cada píxel de la imagen original.

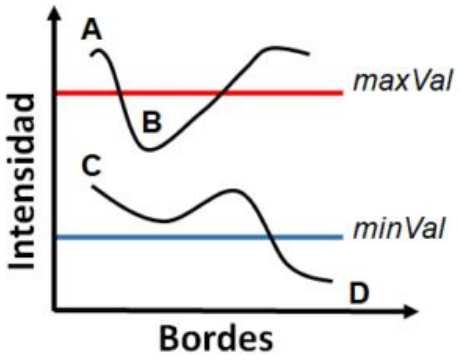
*Gradiente de la imagen: es un cambio direccional en la intensidad o el color de una imagen.

3. **Supresión de falsos máximos**

Esta técnica es utilizada para afinar los bordes encontrados en el paso anterior. Básicamente, consiste en escanear la imagen para eliminar los píxeles que no formen parte de los bordes. Para esto se compara el valor de cada píxel con sus vecinos cercanos en la dirección del gradiente (perpendicular al borde). Si el valor del píxel en cuestión es mayor que sus píxeles vecinos, entonces este es considerado un máximo local y el algoritmo lo acepta. De lo contrario, si el píxel resulta no ser un máximo local, entonces es suprimido. El resultado final será una imagen con bordes muy finos.

4. Umbral de histéresis

El procedimiento anterior logra determinar los píxeles que conforman los bordes con bastante precisión. Sin embargo, aún pueden quedar algunos píxeles provenientes del ruido o de variaciones en los colores de la imagen. En esta cuarta etapa se decide cuáles píxeles pertenecen realmente a bordes y cuáles no. Para ello, se deben fijar dos valores de umbral, *minVal* y *maxVal*. Los píxeles con gradientes de intensidad mayores que *maxVal* serán aceptados como pertenecientes a los bordes, mientras que los menores que *minVal* serán descartados. Los píxeles correspondientes a bordes con valores de gradientes que se encuentren entre estos dos umbrales son etiquetados como píxeles débiles. Estos últimos serán o no aceptados, dependiendo de su conectividad. Si están conectados a píxeles "fuertes", se consideran parte de los bordes; de lo contrario, también se descartan. Para entender mejor este procedimiento, veamos el siguiente ejemplo:



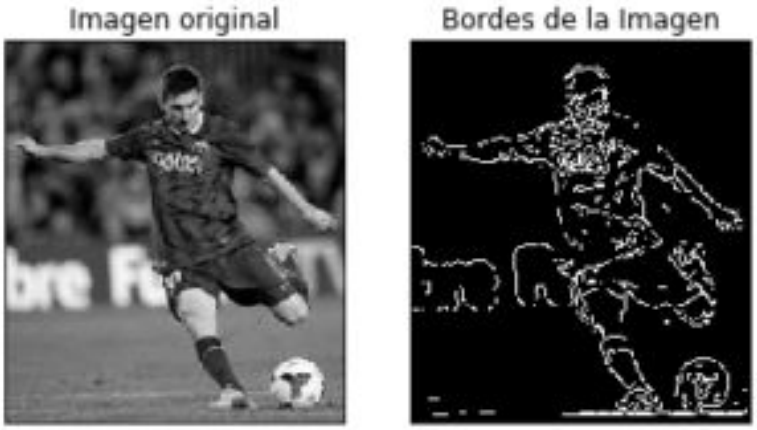
El gráfico muestra el valor de la intensidad de los píxeles que conforman los bordes. En este caso, el píxel A será aceptado como parte del borde dado que su valor supera el umbral *maxVal,* mientras que el píxel D será descartado por tener un valor inferior a *minVal*. Por otra parte, los píxeles B y C se considerados débiles por encontrarse entre los dos valores umbrales. Sin embargo, B será aceptado como parte de un borde, mientras que C no. La razón de esto, es que B está conectado a A, que es un píxel fuerte, pero C sólo está conectado a píxeles débiles o descartados.

Canny Edge en OpenCV

Todos los pasos del algoritmo de Canny, hasta aquí descritos, están contenidos en una simple función en OpenCV: **cv2.Canny()**. Veamos cómo usarla. El primer argumento es la imagen de entrada, mientras que el segundo y tercer argumento son *minVal* y *maxVal*, respectivamente. El cuarto argumento es *aperture_size*, que no es más que el tamaño del kernel Sobel utilizado para buscar gradientes de imagen (por defecto es 3). El último argumento es *L2gradient* que especifica la ecuación para encontrar la magnitud del gradiente.

```
cv2.Canny(img,180,260)
```

Resultado:



Bibliografía

https://www.aprenderpython.net/algoritmo-de-canny/

4.2 Detección Via Plantilla

La detección via templates es un método para buscar y encontrar la ubicación de una imagen plantilla sobre una imagen base.

OpenCV incluye la función cv2.matchTemplate() para este propósito. Simplemente desliza la imagen de la plantilla sobre la imagen de entrada (como en la convolución 2D) y compara la plantilla y parchea la imagen de entrada sobre debajo la imagen de la plantilla. Esta función devuelve una imagen en escala de grises, donde cada píxel indica cuánto coincide el entorno de ese píxel con la plantilla.

Si la imagen de entrada es de tamaño (WxH) y la imagen de la plantilla es de tamaño (wxh), la imagen de salida tendrá un tamaño de (W-w + 1, H-h + 1). Una vez que obtenga el resultado, puede usar la función cv2.minMaxLoc () para encontrar dónde está el valor máximo / mínimo. Para saber la esquina superior izquierda del rectángulo y toma (w, h) como ancho y alto del rectángulo. Ese rectángulo es tu región de plantilla.

Bibliografía

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html