Campus Puebla

Materia

Fundamentación de Robótica TE3001B

Actividad:

Operaciones básicas con imágenes

Alumno:

José Jezarel Sánchez Mijares A01735226

Abril 20 2023

Instrucciones

Ejercicio 1

- Descarga diez imágenes del conjunto de datos Road Sign DetectionLinks to an external site., y elabora un programa que muestre las imágenes a color y en escala de grises.
- Binariza las imágenes del apartado anterior aplicando un nivel de umbral sobre los valores de grises, y muéstralas en tu programa. Viendo las imágenes a color, en grises y en forma binaria, ¿qué imágenes utilizarías para detectar el tipo de señal de tránsito? ¿Qué información consideras importante que puedes recuperar de cada tipo de imagen?
- Binariza las imágenes, pero ahora aplicando un nivel de umbral sobre alguno de los canales de color (cualquiera de los 3). ¿Estas imágenes resultan más informativas que las anteriores?
- A las imágenes en escala de grises, aplica ruido Gaussiano de diferentes niveles, y trata de filtrar dicho ruido con las técnicas de suavizado vistas en clase. ¿Qué técnica te parece adecuada para estos casos?

Con el objetivo de cumplir los objetivos solicitados para la primera actividad desarrolle un código en python usando un notebook, ya que este me permite tener una mejor visualización del código implementado, con eso logre tener una mejor visualización de las funciones implementadas.

Para el punto 1 use la función `cv2.cvtColor()`, lo que hay detrás del código es que implementa una transformación matemática usando una matriz perteneciente a la matriz original, esta matriz mapea los valores de cada píxel en su espacio de color correspondiente, En este caso, se requiere el color gris, por lo que la función combina los valores de los canales de color (BGR o RGB) mediante una fórmula matemática para producir una intensidad de escala de grises. Finalmente, se pueden mostrar las imágenes en escala de grises y originales


Para el punto dos use una función que había creado en clase durante uno de los ejercicios, solo la acondiciona para que funcionara correctamente como lo requería, esta funciona de manera bastante sencilla ya que dado un umbral va a buscar en cada pixel de la imagen dada y si encuentra uno debajo del umbral lo pasará a 0 mientras que si encuentra uno por arriba pondrá el valor de ese pixel en 255, claro que cambiando el umbral se obtendrán resultados diferentes ya que puede ser que se puedan

visualizar más o menos objetos dentro de la imagen.


contestando a *¿qué imágenes utilizarías para detectar el tipo de señal de tránsito? ¿Qué información consideras importante que puedes recuperar de cada tipo de imagen?* Considero que para que se pueda detectar el objeto de manera digital la mejor opción sería la de binaría ya que permite ver el borde y tamaño de los objetos lo que nos permitiría discriminarlos de una mejor manera así detectaremos una mejor el tipo de señal que se observa, datos como el tamaño y limitaciones del objeto son muy útiles

En el tercer punto solo se especificó el canal que quiero cambiar dentro en la imagen, en este caso use el rojo y obtuve el siguiente resultado

Respondiendo a la pregunta *¿Estas imágenes resultan más informativas que las anteriores?* no ya que se observa que se pierden detalles de los edificios como la composición de las ventanas, sin embargo si quisiera pensar en un caso más en específico de solo querer obtener el objeto perse, sería una buena opción

para esta última imagen debido a que tenía que especificar el pixel a usar preferí usar la función `cv2.threshold()`

Por último en el punto 4 use también una función de la librería opencv para el filtro Gaussiano, que es otra forma para referirnos al efecto blur o desenfoque, en este caso lo que sucede es que se toma un pixel en específico y se hace un promedio con los pixeles vecinos que tiene, pero ¿qué pasa cuando estamos en una esquina y faltan vecinos? pues en clase incorporamos otra fila de pixels dummy que completa las parte de la imagen que falta, con esto conseguimos que el efecto se aplique correctamente

Ahora para “limpiar” un poco este efecto de blur tenemos que aplicar un suavizado, la opción que implementamos fue el filtro de mediana que es un tipo de filtro no lineal que se utiliza para eliminar el ruido en las imágenes. Este filtro reemplaza el valor de cada píxel con la mediana de los valores de los píxeles en una ventana de tamaño definido alrededor de ese píxel. El tamaño de la ventana se puede ajustar para lograr un equilibrio entre la eliminación del ruido y la conservación de los detalles de la imagen.

Para aplicar el filtro de mediana a las imágenes generadas por la función del ruido gaussiano, se puede utilizar la función `cv2.medianBlur()` de OpenCV. Esta función toma la imagen de entrada y el tamaño de la ventana del filtro y devuelve la imagen filtrada.

Su precisión aumenta de acuerdo a la precisión del cálculo y tamaño de ventana dado

Ejercicio 2

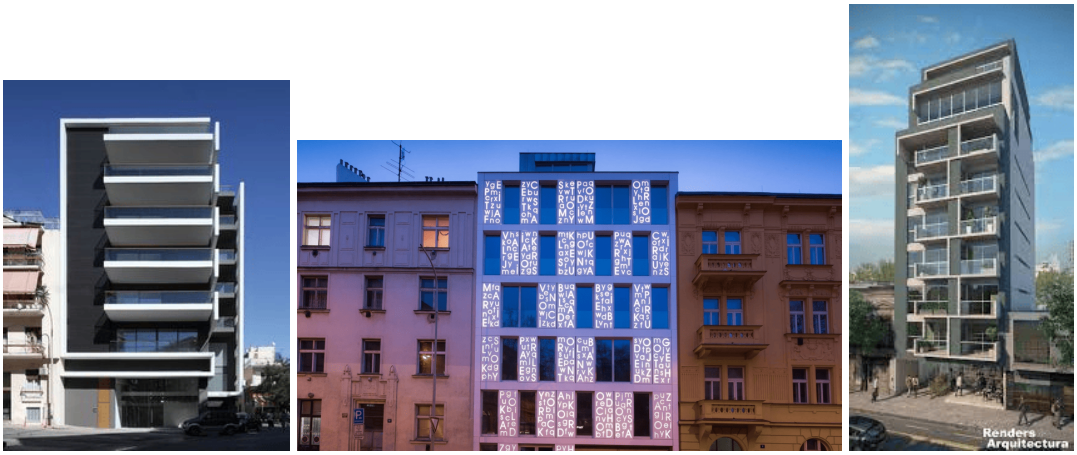
- Busca en internet 5 fotografías de edificios, y aplica los algoritmos vistos en clase de detección de bordes. ¿En qué casos consideramos que los algoritmos tienen problemas en encontrar bordes en las imágenes?
- Prueba con las imágenes de bordes todas las operaciones morfológicas que se hayan visto en clase, e indica qué operaciones mejoran la presentación de los bordes.

En este caso me decante por el algoritmo de Canny ya que es un algoritmo que utiliza múltiples etapas para detectar bordes. Primero, suaviza la imagen para reducir el ruido, luego calcula el gradiente de la imagen y realiza una supresión no máxima para eliminar los bordes falsos. Finalmente, utiliza histéresis para unir los bordes que pertenecen a los mismos objetos.

Se puede mejorar la detección de bordes utilizando la operación morfológica de erosión, ya que esta técnica nos permite erosionar el borde de la imagen y así obtener una mejor percepción del tamaño de los objetos que estamos observando.

Ahora respondiendo a *¿En qué casos consideramos que los algoritmos tienen problemas en encontrar bordes en las imágenes?* observe que cuando las imágenes son oscuras se pueden a llegar a perder los bordes, ya que el algoritmo no llega a detectarlos de manera correcta

Use la función de `cv2.Canny(image, 100, 200)`



Ejercicio 3

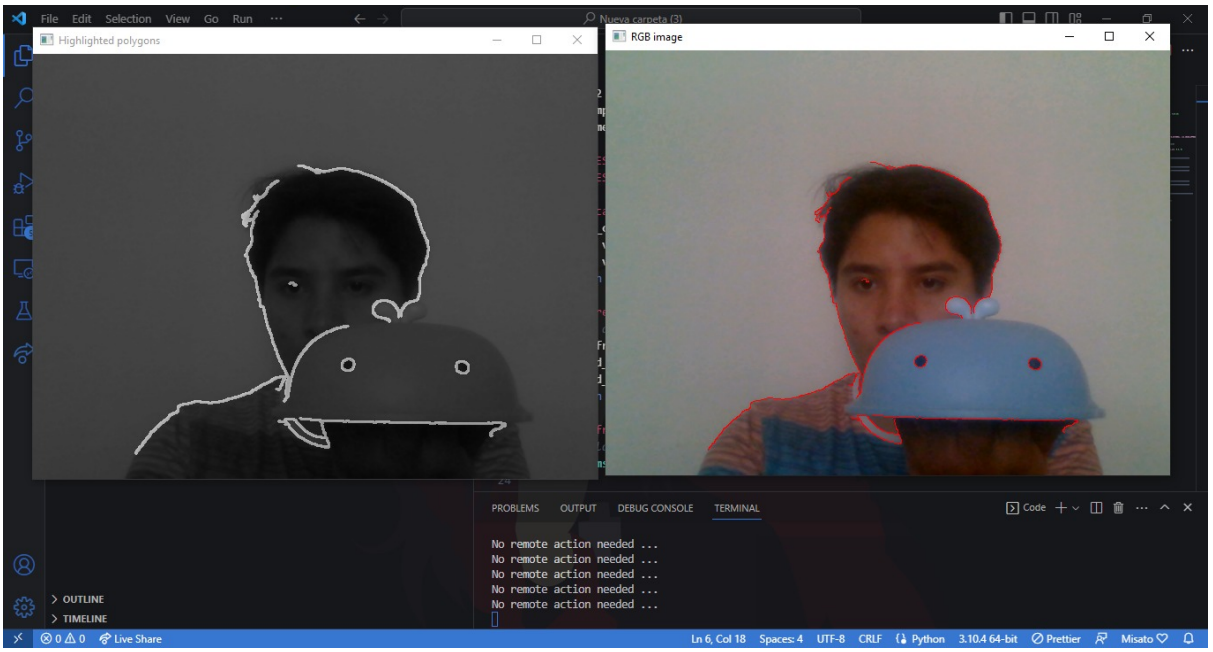
Escribe un programa que muestre en pantalla el video capturado por la cámara conectada a tu computadora, así como el mismo video en niveles de gris, pero con los bordes superpuestos con un color llamativo. En la misma aplicación, muestra un tercer video en niveles de gris, pero que resalta las líneas, círculos y otros polígonos que se detecten. ¿Consideras que las operaciones que estás realizando sobre las imágenes se pueden hacer en un tiempo razonable como para mostrar un video sin retraso?

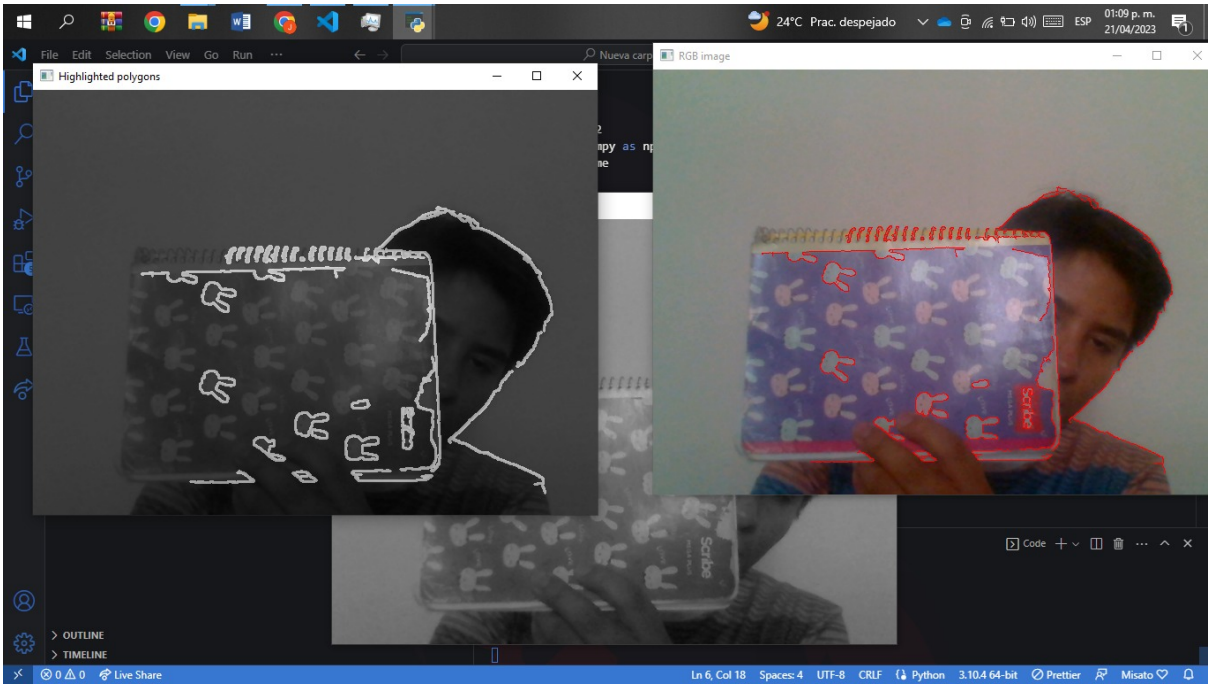
Para el tercer código utilice uno brindado por el profesor el código funciona de la siguiente manera Comenzamos con la función highlight_polygons() aplica el algoritmo Canny para detectar bordes en la imagen en escala de grises. A continuación, encuentra los contornos de los objetos en la imagen y los dibuja en una nueva imagen negra. Luego, superpone la imagen de contornos en la imagen en escala de grises original para resaltar los polígonos detectados. En cada iteración, se captura una nueva imagen de vídeo y se procesa utilizando la función highlight_polygons().

A continuación, se muestran varias versiones de la imagen procesada, incluyendo la imagen original en formato BGR, la imagen en escala de grises y la imagen en escala de grises con los polígonos detectados resaltados.

¿Consideras que las operaciones que estás realizando sobre las imágenes se pueden hacer en un tiempo razonable como para mostrar un video sin retraso?

Si pero esto dependerá de la calidad de nuestro algoritmo, esto se puede solucionar mediante librerías que no retrasan demasiado el tiempo de ejecución, sin embargo también puede depender de la potencia del ordenador, quizás con un sistema embebido mejore todavía más





Para estos códigos tuve la ayuda de mi compañero Antonio Silva Martínez, en especial en la parte de tomar las imágenes ya que me ayudó a que me fuera más óptimo poder tomar las imágenes de las carpetas y así el algoritmo genera todas las imágenes que pedía

Referencias:

opencv-python. (2023). Retrieved 21 April 2023, from https://pypi.org/project/opencv-python/