Tecnológico de Monterrey

Campus Puebla

Materia

Fundamentación de Robótica TE3001B

Tema

Actividad: Operaciones básicas con imágenes

Alumno

Antonio Silva Martínez A01173663

Fecha

Abril 20 2023

Instrucciones

Ejercicio 1 (30 puntos)

Descarga diez imágenes del conjunto de datos Road Sign Detection

Links 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?

Ejercicio 2 (30 puntos)

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.

Ejercicio 3 (40 puntos)

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 resalte 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 desarrollo de esta actividad se han realizado 3 códigos en el primero de estos se tiene que descargar una serie de imágenes para el procesamiento de estas mismas, la primera aparte de esta actividad consistía en convertir a una escala de grises las imágenes que obtuvimos del repositorio

Para mostrar las imágenes en escala de grises y en forma binaria, se puede utilizar la función cv2.cvtColor() para convertir las imágenes a escala de grises, el funcionamiento de la librería es por medio de una transformación matemática la cual de la matriz de la imagen original se utiliza una otra matriz para transformar a los colores que necesitamos, siendo que en este caso es el color gris. Esta transformación se basa en una matriz de conversión llamada matriz de transformación de color o matriz de coeficientes de conversión, que se utiliza para mapear los valores de cada pixel en el espacio de color original a los valores correspondientes en el espacio de color de destino, la función utiliza una fórmula matemática que combina los valores de los canales de color (BGR o RGB) en una sola intensidad de escala de grises.

Los resultados que podemos obtener son los siguientes:

Para convertir las imágenes a binario fue necesario el obtener primero la imagen a blanco y negro para poder procesar posteriormente a binario, la función utilizada fue cv2.threshold() ya que para binarizar las imágenes se utiliza un nivel de umbral sobre los valores de grises. El funcionamiento de la función es el siguiente:
La imagen de entrada se convierte a una imagen binaria. Para esto, se utiliza un umbral, que es un valor que separa los píxeles en dos grupos: aquellos que cumplen una condición dada (por ejemplo, que su valor de intensidad sea mayor que el umbral) y aquellos que no cumplen esa condición.

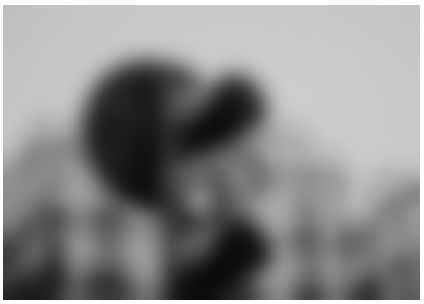
Dependiendo del valor del umbral y de cómo se ha especificado el tipo de umbral (por ejemplo, si es un umbral binario, adaptativo, etc.), la función cv2.threshold() asigna valores de blanco o negro a los píxeles de la imagen binaria resultante, en el caso de las siguientes imagenes el umbral utilizado fue de 100.


El siguiente apartado tiene un funcionamiento similar al anterior, siendo que lo que cambia es que nosotros elegíamos el canal al que nosotros elegimos, el canal elegido en este caso fue el rojo, los resultados obtenidos son los siguientes:


Por último, el último punto del apartado uno se nos pedía aplicar un filtro gaussiano en diferentes escalas, se utilizó la función cv2.GaussianBlur() la cual es una operación de filtrado en imágenes que se utiliza para reducir el ruido o las irregularidades en una imagen. Esta función aplica un filtro Gaussiano a la imagen, que es un tipo de filtro lineal que se utiliza para suavizar la imagen.

El filtro Gaussiano se aplica mediante la convolución de la imagen con una matriz de kernel Gaussiana. Esta matriz es simétrica y tiene valores que se ajustan a una distribución Gaussiana o normal. El kernel se aplica a cada píxel de la imagen y suaviza la imagen en función de los valores de los píxeles circundantes. Esta función toma tres argumentos: la imagen a suavizar, el tamaño del kernel del filtro y la desviación estándar del filtro Gaussiano en ambas direcciones x e y. El tamaño del kernel especifica el tamaño de la ventana utilizada para la convolución, y la desviación estándar controla la cantidad de suavizado que se aplicará a la imagen.

En términos matemáticos, se realiza una operación de convolución entre la imagen de entrada y un kernel Gaussiano, los resultados obtenidos fueron los siguientes:

Preguntas:

¿qué imágenes utilizarías para detectar el tipo de señal de tránsito? ¿Qué información consideras importante que puedes recuperar de cada tipo de imagen?

Puedo considerar que para detectar de manera eficaz una señal de tránsito sería necesario el color rojo u amarollo debido a que generalmente casi todos los semáforos son de una tonalidad amarilla y por ende eso nos facilita el poder identificar si se trata de un semáforo el que estamos viendo, por otra parte podemos decir que el color rojo nos ayudaría a poder identificar el estado en el que se encuentra nuestro semáforo ya que podemos percibir ese color y por ende podríamos detenernos si es que se está controlando un vehículo de manera autónoma.

Lo importante de cada imagen es la nitidez o la forma que logramos obtener ya que esto nos servirá para poder identificar las señales de tránsito u algún semáforo.

¿Estas imágenes resultan más informativas que las anteriores?

Si me parecen muy informativas ya que estas nos permiten identificar los semáforos de manera eficiente y por ende podemos utilizarlas para la detección de algún peligro o de alguna señal de alto

¿Qué técnica te parece adecuada para estos casos?

Una de las técnicas que me parecen más efectivas para poder esclarecer imágenes es la erosión, ya que si una imagen se ve muy borrosa podemos identificar de manera más precisa la silueta del objeto en cuestión

Para el segundo punto lo que hicimos fue utilizar algoritmos vistos en clase de detección de bordes, siendo que para poder lograr esto es necesario el convertir a binario nuestras imágenes y posteriormente aplicar algún algoritmo, siendo que los algoritmos que se utilizaron fueron erosión y ruido (Algoritmo de Canny), este algoritmo es un método de procesamiento de imágenes utilizado en la visión por computadora para la detección de bordes en imágenes digitales. Es un método muy popular debido a su precisión, bajo error y capacidad para suprimir el ruido de la imagen.
El algoritmo de Canny se basa en una serie de etapas para detectar los bordes de la imagen. Primero, se suaviza la imagen utilizando un filtro Gaussiano para reducir el ruido. Luego, se calcula el gradiente de la imagen para encontrar la dirección y la magnitud de los cambios en la intensidad de la imagen. Después, se realiza la supresión de no máximos para eliminar los píxeles que no forman parte del borde y reducir el ancho del borde. Por último, se utiliza un umbral de histéresis para clasificar los píxeles en bordes y no bordes, seleccionando los píxeles con una intensidad de gradiente mayor que un umbral alto y conectando los píxeles cercanos con una intensidad de gradiente mayor que un umbral bajo.

La detección de los bordes se puede detectar de mejor manera con la operación morfológica de erosión ya que nos permite desgastar lo que es el borde de la imagen y así podríamos obtener una mejor percepción del tamaño de un objeto que estemos viendo

Preguntas:
¿En qué casos consideramos que los algoritmos tienen problemas en encontrar bordes en las imágenes?
Los algoritmos llegaron a tener problemas al momento que las imágenes están muy oscuras o no se puede lograr detectar bien el borde por decir un ejemplo, cuando algo es del mismo color y es casi imperceptible detectarlo con la computadora

Para el tercer punto se utilizó como base el utilizado por el profesor, este código funciona de la siguiente manera:
La función init_camera() se encarga de inicializar la cámara y configurar su resolución, la función acquire_image() adquiere una imagen de vídeo de la cámara y devuelve dos versiones de ella: una versión de tamaño completo en formato BGR (una variante de RGB) y

una versión reducida en tamaño en formato RGB. 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.

El bucle principal del código se encarga de repetir el proceso de percepción-acción mientras el programa está en ejecución. 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.

Además, se ha incluido un mecanismo de comunicación que imprime "No remote action needed" a intervalos regulares.

Pregunta:

¿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, es posible en parte gracias a la librería ya que nos facilita el uso de cálculos innecesarios, sin embargo al realizar el procesamiento de manera manual es posible que lleguemos a utilizar cálculos innecesarios los cuales perjudiquen el performance de nuestro código y así se alente lo que es nuestro procesamiento de imágenes, sin embargo actualmente con las máquinas que tenemos es posible el realizar este tipo de operaciones ya que son potentes y nos permite el poder realizarlas en un tiempo razonable y sin ningún tipo de problema

Resultados obtenidos

Github:

https://github.com/AntonioSilva-bot/-Actividad-Operaciones-b-sicas-con-im-genes.git

Referencias:

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