Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Curso Académico 2023/2024

Práctica 1

DETECCIÓN DE SEÑALES VIALES

Cristian Fernando Calva Troya

Luis Ovejero Martín

Jaime Rueda Carpintero

Índice de figuras

Índice

1. Introducción

2. Objetivos y metodología

2.1. Descripción del problema

2.2. Objetivos

3 pag.

2.3. Tecnologías

3. Creación de la aplicación

30 pag.

3.1. Normalización

3.1.1. Corrección de color

3.1.2. Corrección de ángulo

3.2. Detección de regiones de alto contraste

3.2.1. Aplicar el filtro de azul

Antes de empezar a detectar regiones, vamos a aplicar el filtro de azul sobre las imágenes normalizadas. Esto lo hacemos con la idea de mejorar la detección de los carteles, ya que al aplicar el filtro de azul podemos determinar de forma más eficaz la zona de imagen que puede pertenecer a un cartel.

3.2.2. MSER

Una vez hemos aplicado este filtro a las imágenes ya podemos empezar a detectar regiones sobre las máscaras que hemos generado con el paso anterior.
Para detectar las regiones de alto contraste definimos la función *mser*. Esta recibirá la lista de imágenes normalizadas y la lista de mascaras que hemos creado al aplicar el filtro de azul.

Crearemos dos listas que nos servirán para guardar aquellas imágenes sobre las que vamos a dibujar las regiones que detectemos y la otra para guardar las regiones detectadas.
La función va a iterar sobre la lista de imágenes y creará una copia de la imagen normalizada para dibujas las regiones y una copia de la máscara sobre la que aplicará *MSER*. Antes de aplicar el algoritmo dilatamos la mascara y restamos la máscara original a la máscara dilatada para obtener una imagen de bordes. Tras esto creamos un *MSER* y ajustamos los parámetros para filtrar un poco las regiones que pueda detectar, por ejemplo, haciendo que el área mínima de las regiones que detecte sea 900 y que la máxima sea 5000.
Las regiones detectadas en una imagen se guardaran en la variable llamada *polygons*. Sobre esa lista iteraremos a continuación y para cada elemento de la lista aplicaremos la función *convexHull* de *openCV*, con la finalidad de hallar el casco convexo de un conjunto de puntos. Esto quiere decir que la usaremos para determinar el perímetro de la región que hemos detectado. Guardaremos estos perímetros en la lista *hulls*, que usaremos más adelante para poder dibujar las regiones en la imagen. También guardamos la lista de regiones detectadas en *detected_regions* y finalmente dibujaremos las regiones sobre la copia de la imagen y la guardaremos en la lista que hemos creado al principio para ello. Por último mostramos la primera y última imagen con las regiones dibujadas y devolvemos las listas de las imágenes con las regiones dibujadas y la lista de las regiones detectadas.
En la siguiente celda del *notebook* vemos un bloque de código con un bucle *for* que está comentado, lo podemos descomentar y ejecutar para ver el resultado de nuestra función sobre el resto de imágenes de test. Para pasar las fotos simplemente vamos pulsando cualquier tecla. En caso de querer parar tenemos que pulsar el cuadrado de arriba a la izquierda y cerrar la ventana para terminar la ejecución.

3.2.3. Extraer los píxeles de la región en un rectángulo

Para este paso vamos a definir la función *rectangle_of_regions* que tiene como objetivo dibujar, usando *cv2.boundingRect*, un rectángulo alrededor de las regiones detectadas en el paso anterior. La función recibirá la lista de imágenes normalizadas y la lista de las regiones detectadas en el paso anterior. Se creará una lista en la que se añadirán las imágenes con los rectángulos dibujados. Iteraremos sobre las imágenes de test y crearemos una copia de cada imagen. Ahora iteraremos sobre las regiones de la imagen sobre la que estamos trabajando y usaremos *cv2.boundingRect*, para obtener el punto superior izquierdo de la región y el ancho y alto de la

misma y con ello dibujaremos el rectángulo con *cv2.rectangle*.
Tras dibujar las regiones añadimos la imagen a la lista, imprimimos la primera y la última de estas imágenes y devolvemos la lista donde las hemos guardado.
Al igual que en la región anterior, tenemos una celda de código comentado que podremos usar para poder ver el resultado en el resto de imágenes.

3.2.4. Filtrar rectángulos por relación de aspecto

En este apartado se pide que filtremos los rectángulos de las regiones, que hemos dibujado en el paso anterior, por su relación de aspecto con la finalidad de eliminar aquellos que disten mucho de la relación de aspecto de los carteles que se pretenden detectar.
Además de esto hemos hecho que la función encargada de realizar esto también elimine los rectángulos contenidos dentro de otros. Esto lo hacemos porque queremos solo detectar la región del panel azul, no los subpaneles. La idea es encontrar los subpaneles más adelante, pero solo sobre la imagen del cartel en lugar de hacerlo en la imagen completa, así podemos eliminar detecciones basura que pueda llegar a hacer en la imagen completa, pero de esto ya hablaremos próximamente.
Crearemos dos listas, una para las imágenes con las regiones filtradas dibujadas y otra para las regiones filtradas.
Luego iteraremos sobre las imágenes, para cada una de estas vamos a crear una lista para guardar las regiones filtradas de esta imagen y copiaremos la imagen para poder dibujar las regiones en la copia así como hemos anteriormente.
Ahora iteraremos sobre la lista de regiones que pertenecen a esta imagen. Para cada región que hay en la lista vamos a usar *cv2.boundingRect* para obtener el ancho y el alto, con esto podremos calcular la relación de aspecto de las regiones. Nos vamos a quedar con las regiones que tengan una relación de aspecto (ancho/alto) entre 0.6 y 6.5.
Con la regiones que cumplan esto vamos a hacer la comprobación de si están dentro de otra, en caso de que se cumpla y esté dentro de otra, no nos vamos a quedar con ella, solo queremos quedarnos con las regiones más exteriores con la idea de quedarnos con la región del cartel completo, ya que como hemos dicho antes los subpaneles los buscaremos luego.
Iremos agregando las regiones que cumplan ambas condiciones a la lista *filtered_regions* y dibujando las mismas en sus imágenes correspondientes, estas las añadiremos también a *images_filtered_rectangles*.

Finalmente mostraremos la primera ya la última imagen ya que en estas vemos como se eliminan algunas regiones que no cumplen con las exigencias que queremos. Por último vamos a devolver la lista de imágenes y de regiones filtradas.
Al igual que en los pasos anteriores tenemos una celda comentada para poder ver el resultado en el resto de imágenes.

3.2.5. Expandir regiones filtradas

3.3. Correlación por máscaras

3.3.1. Mascara ideal

Para la correcta comprobación de que si una imagen ya filtrada por color es un panel deseado necesitamos un panel de referencia, para poder indicar como debería ser un panel detectado de forma aproximada.
Para la correcta comprobación de que si una imagen ya filtrada por color es un panel deseado necesitamos un panel de referencia, para poder indicar como debería ser un panel detectado de forma aproximada.

3.4. Eliminación de regiones repetidas

3.5. Validación

3.6. ...

4. Conclusiones y trabajos futuros

3 pag.

5. Referencias

1 pag.