Ingeniería Informática (Universidad Rey Juan Carlos)

Asignatura: “Visión Artificial”
Curso 2023/24

Práctica Obligatoria 1: “Detección y normalización de paneles de señalización vertical azul en autovías”

Normas generales:

- La práctica deberá realizarse en Python usando el entorno Jupyter Notebook, Visual Studio Code o similar (en combinación con las librerías correspondientes).

- Habrá que entregar: 1) los códigos desarrollados/usados para la realización de la práctica y 2) una breve memoria explicativa, que incluirá: la descripción de la solución, los resultados conseguidos junto con su análisis y las conclusiones del trabajo. La entrega se realizará a través de una ´”carpeta de entregas” que se habilitará para tal fin en el Aula Virtual de la asignatura.

- La práctica se realizará **exclusivamente en grupos de dos o tres alumnos** (como máximo). Para ello, será obligatorio inscribir el grupo de prácticas a través del Aula Virtual (hasta el día 24 de marzo de 2024, incluido).

- El plazo límite de entrega será de la práctica será el día **3 de mayo de 2024** (incluido).

1 Copias de código o de la memoria

El código desarrollado en las prácticas debe de ser original. La copia (total o parcial) de prácticas será calificada con una nota de 0 puntos en dicha prueba. En estos casos, en la siguiente convocatoria el grupo tendrá que **defender oralmente la solución de su práctica**. **Las sanciones derivadas de la copia, afectarán tanto al grupo que copia como al grupo copiado.**

Para evitar que **cuando se usa código de terceros** sea considerado como una copia, se debe **citar siempre la procedencia** de cada parte de código no desarrollada por el propio alumno (con comentarios en el propio código y con mención expresa en la memoria).

2 Detección de paneles de señalización vertical azul en autovías

Se desea construir un sistema para la detección automática de paneles informativos de tráfico de color azul, en imágenes realistas que han sido tomadas desde un coche. Los paneles se han diseñado para que sean fácilmente distinguibles del entorno en cualquier condición de luz, color de fondo y climatología. La Figura 1 muestra algunos ejemplos de estos paneles azules que deben ser detectados:

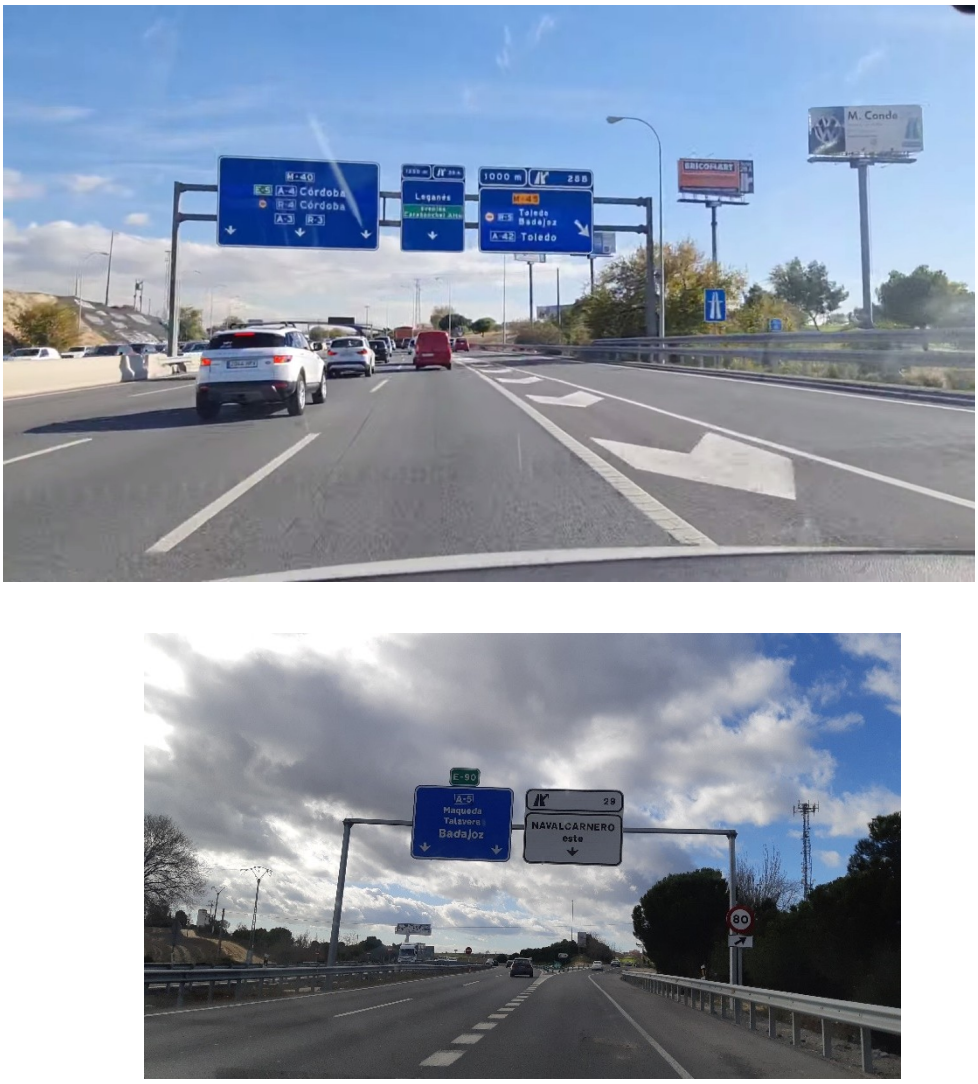

Figura 1. Dos ejemplos de imágenes conteniendo señalización vertical azul a detectar.

Por tanto, tienen unas formas geométricas regulares, unas dimensiones estandarizadas, colores muy vivos y son reflectantes, como puede verse en la Figura 2.

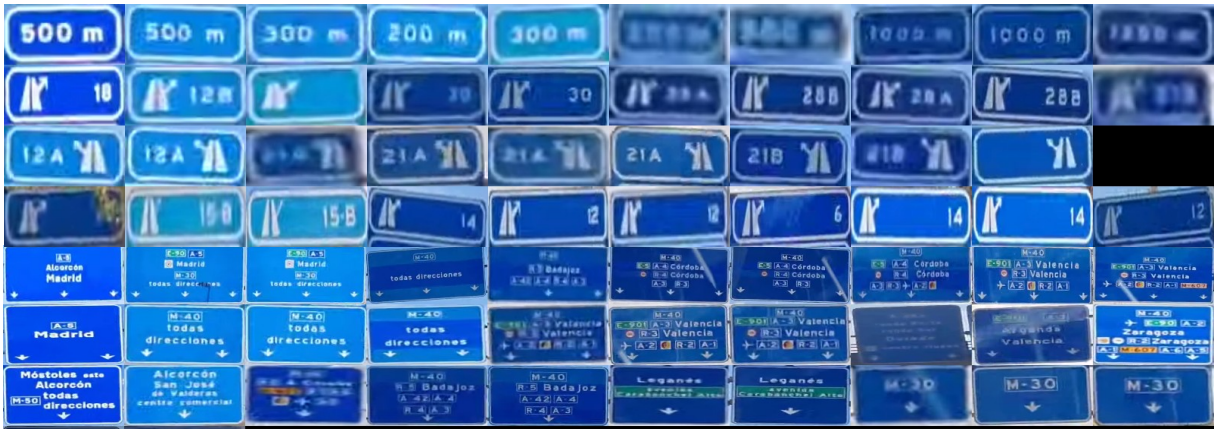

Figura 2: Algunos ejemplos de paneles de tráfico a detectar

Observando la Figura 2 podemos hacernos una idea sobre el tipo de información que nos permitirá distinguir un tipo de otro de panel, e incluso localizarlos en una imagen de carretera. En particular, se distinguen por su forma geométrica rectangular, por las zonas en las que se divide (borde claro y fondo oscuro) o por el color del mismo (azul y blanco). Es decir, para detectar este tipo de paneles podríamos utilizar:

- Un detector de regiones de alto contraste (detectaríamos la parte interna del panel).
- Un algoritmo que detectase rectángulos (p.ej., combinando detección de líneas con detectores de esquinas).
- Un algoritmo para descubrir qué píxeles son de color azul en la imagen junto con su distribución espacial.
- Detectar ciertas formas que aparecen en los paneles en localizaciones especiales (símbolo de salida de la autopista por la izquierda o derecha, diversos tipos de flechas, etc.).
- Cualquier otra técnica que tenga en cuenta color y forma.

2.1 Desarrollo de la práctica

El objetivo de esta práctica es desarrollar un detector básico de sub-paneles de información en autovías. Hablamos de sub-paneles porque la idea es detectar cada una de las regiones rectangulares azules enmarcadas con un borde blanco en los paneles de la carretera (ver imagen superior de la Figura 1).

Para desarrollar el algoritmo de detección, como en muchos otros problemas de este tipo, se ofrece una colección de imágenes de test tomadas desde un coche.

Para detectar los sub-paneles de tráfico se pide seguir los siguientes pasos generales:

1. Utilizar MSER como detector de regiones de alto contraste (*mser.detectRegions*):
    - Pasar la imagen a niveles de gris (y, posiblemente, mejorar su contraste con las técnicas vistas en clase).
    - Hay que tener en cuenta que los parámetros de MSER se pueden ajustar en el constructor de la clase (*cv2.MSER_create*). El ajustar los parámetros puede suponer eliminar muchas detecciones incorrectas (http://stackoverflow.com/questionshttps://stackoverflow.com/questions/53317592/reading-pascal-voc-annotations-in-python/17647500/exact-meaning-of-the-parameters-given-to-initialize-mser-in-opencv-2-4-x).
    - Extraer en un rectángulo los píxeles de la región detectada (*cv2.boundingRect*).
    - Se pueden eliminar las regiones con una relación de aspecto (ancho/alto) muy diferente de la que tienen los sub-paneles. Si las regiones son demasiado alargadas en horizontal o vertical se podrán eliminar.
    - En los sub-paneles se podría detectar originalmente la región azul interna al borde blanco. Por tanto, habrá que agrandar un poco el rectángulo detectado para que la imagen recortada contenga el sub-panel completo (incluyendo el borde blanco, y no sólo su parte interna). Si no se hace esto podemos tener una tasa de detección peor de la que realmente tenemos dado que las detecciones no solaparán completamente con las anotaciones.

2 Utilizar el espacio de color HSV para localizar los píxeles que sean del color azul (característico del panel) y que estén muy saturados (tiene que ser un azul bastante puro):
    - Para ello, se construye un *np.array* de tamaño fijo (p.ej. 40x80) en el que los píxeles azules y muy saturados tengan valor 1, y 0 el resto. Esta matriz será la máscara de color azul saturado del sub-panel informativo general. Indicaría en qué píxeles debería tener color azul saturado el subpanel.
3 Detección mediante correlación de máscaras. Los pasos deberían de ser los siguientes:
    - Recortar cada ventana detectada por MSER en una imagen de entrada y cambiar su tamaño a uno fijo (p.ej. 40x80 píxeles) con *cv2.resize*.
    - Extraer la máscara de color azul saturado *M* de la ventana redimensionada.
    - Correlar M (multiplicar elemento a elemento y sumar), con máscara de color azul saturado ideal que debería tener un sub-panel de carretera (p.ej. casi todos los píxeles a 1). Se puede usar el valor de correlación para saber cuál es la proporción de píxeles de color azul saturado en el sub-panel evaluado. Si se pone un umbral también se pueden rechazar ventanas como "no panel" cuando tienen una correlación muy baja (es decir, las que "tienen poco azul").
    - Si al correlar se pasa el umbral, establecer el valor de correlación como la puntuación, o *score*, que daremos a esa ventana detectada. Ello nos indicará "cómo de parecido es lo detectado a un sub-panel" esa parte de la imagen (0 no es sub-panel y 1 sí lo es).

2.2 Evaluación de los resultados de detección

Deberá quedar reflejada en la memoria de la práctica los resultados obtenidos sobre las imágenes de test proporcionadas. Se deberán mostrar resultados *cualitativos* y *cuantitativos* sobre las imágenes de test entregadas.

Para la evaluación de resultados se deberán seguir los siguientes pasos:

1) Seleccionar un subconjunto de, al menos, 20 imágenes del conjunto de imágenes de test proporcionado. Estas imágenes deben de ser lo suficientemente representativas como para contener todas las variabilidades presentes en el *dataset* de imágenes (p.ej., diferentes condiciones meteorológicas, diferentes tamaños y perspectivas de los sub-paneles en las escenas, etc).

2) Anotar (o marcar) las posiciones de los sub-paneles presentes en cada una una de las imágenes seleccionadas. Dichas anotaciones constituyen el *ground truth* del conjunto de imágenes a evaluar, y contra ellas se compararán nuestras detecciones de sub-paneles para evaluar cuantitativamente los resultados conseguidos por nuestro sistema.

Para la anotación de imágenes podría utilizarse cualquier aplicación disponible para este propósito. Por ejemplo, *VGG Image Annotator* o *LabelImg*.

Un ejemplo de anotación se muestra sobre un fragmento de una imagen de prueba:

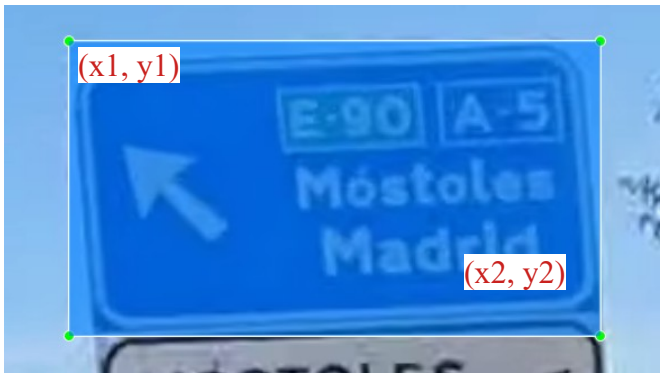

Figura 3: Ejemplo de anotación de sub-panel

Al anotar un sub-panel contenido en una imagen, el programa de anotación guardaría por cada objeto considerado una información del tipo:

<nombre_fichero>;<x1>;<y1>;<x2>;<y2>

donde: (x1, y1) y (x2, y2) son, respectivamente, las coordenadas de la esquina superior izquierda e

inferior derecha del sub-panel en la imagen dada por <nombre_fichero>. Por ejemplo:

00000.png;1491;339;1724;468

Los programas de anotación también guardan la clase de objeto anotado (en este caso, solo tenemos la clase “sub-panel”).

3) Los *resultados cualitativos* consisten en pintar (por ejemplo, en color rojo) para el subconjunto de imágenes de test seleccionado, los rectángulos donde se han detectado los sub-paneles en cada una de las imágenes consideradas.

4) Para calcular los *resultados cuantitativos*, se utilizarán los *ground truth* (o anotaciones) del subconjunto de imágenes de test, que se compararán con las correspondientes detecciones de sub-paneles en las mismas imágenes. Para ello, se calculará la métrica del *Intersection over Union* (IoU) entre cada detección en dicho subconjunto de imágenes y su correspondiente anotación. Esta IoU consiste en el grado de solapamiento entre el rectángulo que contiene el objeto detectado y el correspondiente a su anotación.

Se podría considerar un objeto sub-panel como “correctamente detectado” (*True Positivo*) cuando el valor de IoU para dicho objeto supera el 0.5 (es decir, al menos un 50% de solapamiento).

5) Por último, en la memoria hay que comentar las dificultades y/o problemas que ha presentado la implementación del método propuesto. Indicar también posibles ideas de mejora (u otros tipos de algoritmos propuestos) para resolver este problema.

3 Eliminación de detecciones repetidas

MSER detecta la parte interna del sub-panel (p.ej. la parte azul dentro del borde blanco) varias veces dependiendo de los parámetros que le pasemos y además la parte externa (el borde), con lo que muy probablemente tendremos más de una detección correspondiente a cada sub-panel.

En esta parte de la práctica se pide diseñar un algoritmo para la que las ventanas repetidas se queden reducidas a una sola detección. Este algoritmo puede utilizar alguna de las siguientes ideas:

- Definir un criterio de solapamiento de ventanas (p.ej. el área de la intersección dividido por el área de la unión de dos ventanas).
- Elegir la ventana que contenga a otras de las que se solapen.
- Elegir la ventana promedio de las que solapen.
- Otros criterios propuestos por los alumnos.

4 Normalización de paneles detectados

Muchos paneles detectados en las imágenes pueden aparecer ligeramente rotados (debido al efecto de la perspectiva) y/o también afectados por unas condiciones de iluminación desfavorable. Ello puede dificultar el posterior reconocimiento de la información contenida en los mismos. La Fig. 4 ilustra un ejemplo de esta situación. En este apartado se pide desarrollar un código que aplique la normalización indicada con respecto a la posición del cartel en la imagen y/o la corrección del efecto de la iluminación.

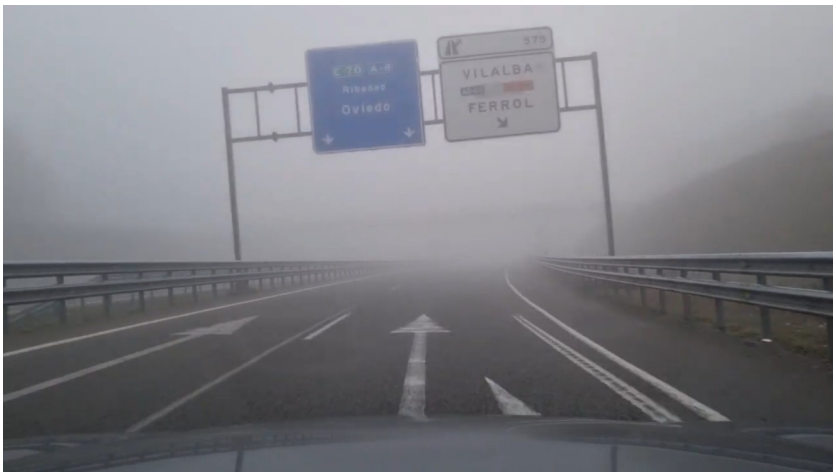

Figura 4: Ejemplo de un panel no normalizado

6 Normas de presentación

La presentación seguirá las siguientes normas:

- Se deberá entregar un único fichero ZIP que contendrá el código fuente y un fichero PDF con la descripción de la solución desarrollada (memoria de la práctica).
- Dicho fichero PDF incluirá una explicación con el algoritmo desarrollado, los métodos de OpenCV utilizados, copias de las pantallas correspondientes a la ejecución del programa y unas estadísticas correspondientes al resultado de la ejecución del programa sobre la muestra de test.
- Se puede presentar, de forma alternativa, el código fuente solicitado usando el *notebook* que se adjunta con el enunciado de la práctica.

La puntuación de esta práctica corresponde al 30% del peso de la asignatura. La práctica se valorará sobre 10, y para poder hacer media se necesita al menos un 4,5 de nota mínima. Cada parte de la práctica tendrá la siguiente puntuación:

- Ejercicio 1 (sección 2 - algoritmo de detección propuesto): **6 puntos.**
- Ejercicio 2 (sección 3 – eliminar detecciones repetidas): **1 punto.**
- Ejercicio 3 (sección 4 – normalización de las imágenes detectadas)**: 3 puntos.**

Se valorarán los siguientes aspectos:

- Implementar la solución de la manera explicada en este enunciado.
- La limpieza y organización del código en clases, con un Diseño Orientado a Objetos razonable.
- El funcionamiento correcto del código en las imágenes de test.
- Las ideas propias o técnicas pensadas por los alumnos para mejorar lo que se propone en este enunciado.

5 Referencias

1. Detector de Regiones de Interés MSER:

   http://stackoverflow.com/questions/17647500/exact-meaning-of-the-parameters-given-to-initialize-mser-in-opencv-2-4-x

2. Ejemplo de uso de MSER:

   https://github.com/opencv/opencv/blob/master/samples/python/mser.py

3. Documento informativo sobre señalización vertical de carreteras en España (Norma 8.1-IC):

   https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiW0NHrzer9AhVW_rsIHSm4CIQQFnoECA0QAQ&url=https%3A%2F%2Fwww.fomento.gob.es%2Faz.bbmf.web%2Fdocumentacion%2Fpdf%2Fre3723.pdf&usg=AOvVaw0fiTv9E-tx5peATLWOjwx4

4. Intersection over Union (IoU):

   https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef

5. Herramienta de anotación *VGG Image Annotator*:

   https://www.robots.ox.ac.uk/~vgg/software/via/

6. Herramienta de anotación LabelImg:

   https://viso.ai/computer-vision/labelimg-for-image-annotation/