Práctica obligatoria 1: Detección de objetos

El enunciado corresponde a la primera práctica calificable de la asignatura: **entrega día del examen de mayo.**

Importante: En esta práctica **s**olamente se podrán utilizar las técnicas de procesamiento de imagen vistas en clase. No está permitido utilizar clasificadores (módulo ml de OpenCV) ni otras técnicas parecidas (p.ej. paquetes de python como sklearn, torch, etc) que se verán más adelante.

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 sancionada, al menos, con el SUSPENSO global de la asignatura en la convocatoria correspondiente. En estos casos, en la siguiente convocatoria supondrá el tener que **resolver nuevas pruebas (distintas de las de mayo) y la defensa de las mismas de forma oral**. **Las sanciones derivadas de la copia, afectarán tanto al alumno que copia como al alumno copiado.**

Para evitar que **cuando se usa código de terceros** sea considerado 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). El plagio o copia de terceros (p.ej. una página web) ya sea en el código a desarrollar en las prácticas y/o de parte de la memoria de las prácticas, sin la cita correspondiente, acarrearán las mismas sanciones que en la copia prácticas de otros alumnos.

2 Detección de paneles de información de autopistas

Se desea construir un sistema para la detección de ciertos paneles informativos de tráfico en imágenes realistas (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. Por tanto, tienen unas formas geométricas regulares, colores muy vivos y son reflectantes (ver Figura 1).

Figura 1: Ejemplo de paneles de tráfico a detectar

Observando la Figura 1 podemos hacernos una idea sobre el tipo de información que nos permitirá distinguir un tipo de otro e incluso localizarlas en una imagen de la carretera o la calle. En particular se distinguen por la forma geométrica rectangular, por las zonas en las que se divide (borde claro y fondo oscuro) o por el color (azul y blanco). Es decir, para detectar 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 la práctica 1 es desarrollar un detector básico de paneles de información en autopista. Vamos a definir panel como cada una de las regiones rectangulares azules enmarcadas con un borde blanco en los paneles más grandes que se ven en autopistas españolas (Se pueden ver ejemplos en la Figura 1).

Para desarrollar el algoritmo de detección, como en muchos otros problemas de detección, se ofrecen imágenes de entrenamiento y de prueba tomadas desde un coche. Además, en un fichero de texto *gt.txt* aparecen las coordenadas de ventanas ("Bounding Boxes") que enmarcan cada panel. El fichero de texto con las anotaciones, una por línea, sigue el siguiente formato:

```
<nombre_fichero>;<x1>;<y1>;<x2>;<y2>;<tipo>;<score>
```

donde (x1, y1) son las coordenadas de la esquina superior izquierda de la ventana que enmarca el panel en la imagen dada por <nombre_fichero> y (x2, y2) son las coordenadas de la esquina inferior derecha de la ventana. El tipo de sub-panel en este caso siempre está anotado con la etiqueta 1 y el score también con un 1. En la Figura 2 podemos ver un ejemplo de anotación.

Figura 2: Ejemplo de anotación de un panel

Un ejemplo de anotación en el fichero es:

```
00000.png;1491;339;1724;468;1;1
```

Para detectar los 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 el contraste con las técnicas vistas en clase).
   - Tened 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).
   - Habrá que pasar a 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 paneles. Si las regiones son demasiado alargados en horizontal o vertical se podrán eliminar.
   - En los paneles se detecta la región azul interna al borde blanco. Por tanto, habrá que agrandar un poco el rectángulo detectado para que el panel completo, y no sólo la parte interna, esté dentro de la región detectada. Si no se hace esto podemos tener una tasa de detección peor que 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 de color azul y que estén muy saturados (tiene que ser un azul bastante puro):
   - Construir 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 panel informativo general. Indicaría en qué píxeles debería tener color azul saturado.
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 panel de carretera (p.ej. casi todos los píxeles a 1). Se puede usar el valor de correlación para saber cuál es la proporcion de píxeles de color azul saturado en el panel evaluado. Si se pone un umbral también se pueden rechazar ventanas como “no panel” cuando tienen una correlación muy baja (tiene poco azul).

- Si pasa el umbral, establecer el valor de correlación como la puntuación, o *score*, que daremos a esa ventana detectada que nos dirá "cómo de parecido es a un panel" esa parte de la imagen (0 no es panel y 1 lo es). Se recomienda calcular un *score* que tenga en cuenta el número de píxeles del color adecuado en el lugar correcto (*recall*) pero también cuántos de los píxeles con el color adecuado en el lugar incorrecto se han detectado (*precision*).

2.2 Evaluación de los resultados de detección

Deberá quedar reflejada en la memoria de la práctica los resultados obtenidos sobre las escenas de test suministradas. Para ello se volcará el fichero *resultado.txt* con el mismo formato que el fichero gt.txt de las anotaciones y se comparará entre el resultado obtenido en la detección de paneles con los resultados de la implementación de los profesores (ficheros proporcionados en el Aula Virtual). Se utilizará el script proporcionado *evaluar_resultados.py.* A este script se le pasa el path al directorio de *test_detection* y se le llama desde el directorio donde se encuentran los ficheros *resultado.txt* generado y el fichero "resultado_jmbuena_road_panels.txt" proporcionado en el Aula Virtual. Este script dibujará la curva precision-recall del detector que generó el fichero "resultado.txt" en comparación con los resultados de la implementación de los profesores de la asignatura para la práctica (Ver Figura 3). El *recall* es la proporción de paneles que se han detectado de todos los presentes en las imágenes (un 0.8 es detectar el 80% de todos los sub-paneles), *y precision* es la proporción de paneles encontrados entre todas las detecciones realizadas (un 0.5 significa que entre todas las detecciones únicamente el 50% eran paneles). Cuanto más alta la curva (más cercana a 1.0 en todos los valores de *recall*), mejor será el detector. Ejemplo de uso:

python3 evaluar_resultados.py –test_path ./test_detection

El script también permite enseñar el resultado de la detección imagen a imagen pasando el parámetro "*--show_detections True"* a *evaluar_resultados.py*.

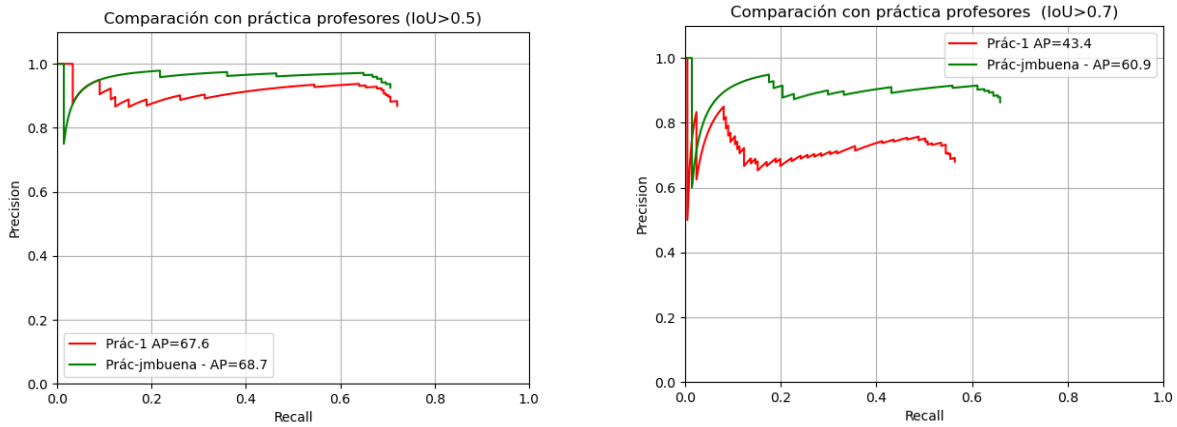

Figura 3: Curvas "precision-recall" de un método de detección vs las de la implementación de los profesores. AP es el valor de "Average Precision" para cada curva y una métrica muy utilizada en detección.

El valor de *Intersection over Union (o IoU)* de una detección con el rectángulo anotado mide el el % de solapamiento entre los dos rectángulos. El script genera dos curvas, una con un umbral de solapamiento de 0.5, que sirve para saber si más o menos todas las detecciones están sobre anotaciones, y otra con un umbral de 0.7 que sirve para saber si realmente las detecciones son precisas. En la memoria de la práctica es obligatorio mostrar las dos figuras que genera el script y comentar los resultados.

3 Eliminación de detecciones repetidas

MSER detecta la parte interna del 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 tendremos más de una detección en cada panel.

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

- Definir un criterio de solapamiento de ventanas (p.ej. área de la intersección dividido por el área de la unión de dos ventanas).
- Elegir la ventana con mayor *score* de las que solapen.
- Elegir la ventana promedio de las que solapen.

4 Otros filtros o ideas para detectar paneles

Se valorarán especialmente las prácticas que, además de utilizar el algoritmo establecido en el enunciado, implementen otros detectores que utilicen ideas nuevas o diferentes para detectar paneles (p.ej., transformada Hough para detectar líneas, detectores de Puntos de Interés tipo esquina, votación a la Hough de descriptores, etc).

Importante: Solamente se podrán utilizar técnicas de procesamiento de imagen como las vistas en clase u otras parecidas aunque no las hayamos explicado. En esta práctica no está permitido utilizar clasificadores (módulo ml de OpenCV) ni otras técnicas parecidas (p.ej. paquetes de python como sklearn, torch, etc).

5 Datos de entrenamiento, de test y formato de salida

Los **datos de entrenamiento** proporcionados son imágenes .png con escenas de autopista en las que aparecen paneles de carretera y el correspondiente fichero gt.txt (con el formato explicado en el apartado 2.1).

Para las **imágenes de test** se proporciona también un fichero de anotaciones gt.txt para poder establecer la tasa de acierto en la detección. Las imágenes se encuentra disponibles también en formato .png.

Todas las prácticas entregadas deberán:

- Utilizar el script python *main.py* que se proporciona junto con este enunciado.
- La llamada a *main.py* tiene la siguiente estructura:

  python main.py –train_path /home/usuario/train --test_path /home/usuario/test –detector detector

  - **El primer parámetro** es el directorio donde están las imágenes de entrenamiento (incluyendo el fichero "gt.txt").
  - **El segundo parámetr**o es el directorio donde están las imágenes de test y su correspondiente fichero "gt.txt".
  - **El tercer parámetro** es un string con el nombre del detector a ejecutar (por si el alumno implementa más de uno).
- El script main.py deberá crear un directorio "resultado_imgs" desde donde ejecute y guardar en el mismo las imágenes de test procesadas con los rectángulos de las detecciones en rojo y el score en texto amarillo.
- El resultado de *main.py* también deberá ser un fichero "resultado.txt", con todas las detecciones para todas las imágenes de test en el mismo, con el mismo formato explicado en la sección 2.1. La diferencia es que ahora añadiremos el *score* al final de cada línea calculado por nuestro detector para cada panel. El identificador de clase será siempre 1.

  Por tanto, un ejemplo de panel detectado en la imagen 0000.png, con *score* 0.6, será:

  `00000.png;774;411;815;446;1;0.6`

6 Normas de presentación

La presentación seguirá las siguientes normas:

- Su presentación se realizará a través del Aula Virtual en la fecha del examen**.**
- Para presentarla se deberá entregar un único fichero ZIP que contendrá el código fuente, el ejecutable y un fichero PDF con la descripción del sistema desarrollado.
- 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. En particular se pide utilizar el script proporcionado (la curva *precision-recall*).
- Se permitirán grupos de hasta 3 alumnos.

La puntuación de esta práctica corresponde al 25% de la asignatura. La práctica se valorará sobre 10, para poder hacer media se necesita al menos un 4,5, y cada parte tendrá la siguiente puntuación:

- Ejercicio 1 (sección 2 - algoritmo de detección básico): **6 puntos.**
- Ejercicio 3 (sección 4 – eliminar detecciones repetidas): **1 punto.**
- Ejercicio 4 (sección 5 – otras ideas originales): **3 puntos.**

Para obtener la máxima nota en cada apartado, se valorará:

- Implementar el *main.py* de la manera como se pide en la sección 5.
- La limpieza y organización del código en clases con un Diseño Orientado a Objetos razonable.
- El funcionamiento del código en las imágenes de test.
- Las ideas propias o técnicas pensadas por los alumnos para mejorar lo que se propone en el enunciado.

7 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