Práctica obligatoria 1

Detección de objetos

El enunciado corresponde a la primera práctica puntuable 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, además, no regirá la liberación de partes de la asignatura (habrá que volver a presentarse al examen) y podrá significar, en la siguiente convocatoria y a discreción del profesor, el tener que **resolver nuevas pruebas 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 de las prácticas). 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 señales de tráfico

Se desea construir un sistema que permita la detección de ciertas señales de tráfico en imágenes realistas (tomadas en la calle desde un coche). Las señales de tráfico 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 señales 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,

las que hemos seleccionado, se distinguen por la forma geométrica (triángulo, octógono o círculo), por las zonas en las que se divide (borde oscuro y fondo claro o fondo oscuro) o por el color (rojo, blanco, negro). Es decir, para detectar señales de tráfico podríamos utilizar:

- Un detector de regiones de alto contraste (detectaríamos la parte interna de la señal).
- Un algoritmo que detectase círculos (p.ej. la transformada Hough), triángulos u octógonos (p.ej. combinando detección de líneas con detectores de esquinas).
- Un algoritmo para descubrir qué píxeles son de color rojo en la imagen junto con su distribución espacial (p.ej. circular en el caso de las señales de peligro).
- Cualquier otra técnica que tenga en cuenta color y forma.

2.1 Desarrollo de la práctica

El objetivo de la práctica 1 a entregar es desarrollar un detector muy básico de señales de tráfico para los tres tipos principales: prohibición (con fondo blanco), peligro y stop.

Para desarrollar el algoritmo de detección, como en muchos otros problemas de detección, se ofrecen imágenes de entrenamiento tomadas desde un coche. Además, en un fichero de texto adjunto aparecen las coordenadas de ventanas ("Bounding Boxes") donde aparecen las señales y su tipo. El fichero de texto con las señales de tráfico anotadas, una por línea, sigue el siguiente formato:

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

donde (x1, y1) son las coordenadas de la esquina superior izquierda de la señal de tráfico en la imagen dada por <nombre_fichero> y (x2, y2) son las coordenadas de la esquina inferior derecha de la ventana de la señal de tráfico.

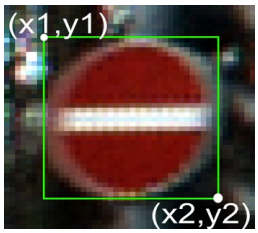

Figura 2: Ejemplo de anotación de señales

Un ejemplo de anotación en el fichero es:

```
00000.ppm;774;411;815;446;11
```

En el fichero de texto "gt.txt" contiene las anotaciones de señales y permite distinguir en multitud de tipos de señales de tráfico (p.ej. Obligación, velocidades máximas, diferentes peligros, etc). Los tres tipos que queremos detectar en la práctica aparecen en el fichero de anotaciones con los siguientes índices:

- Prohibición: [0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 15, 16]
- Peligro: [11, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]
- Stop: [14]

Para detectar las señales de tráfico se sugiere seguir los siguientes pasos:

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).
   - 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/questions/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 1.0.
   - En las señales de prohibición se detecta la región blanca de la misma, en las de peligro lo mismo y en las de stop, se detecta la región roja (suelen tener un borde blanco). Por tanto, habrá que agrandar el cuadrado detectado para que la señal completa, y no solo la parte interna, esté dentro de la región detectada.

2 Utilizar el espacio de color HSV para localizar los píxeles que sean de color rojo:

Establecer rango de valores de H y S que permitan detectar los píxeles rojos (ver Figura 3).

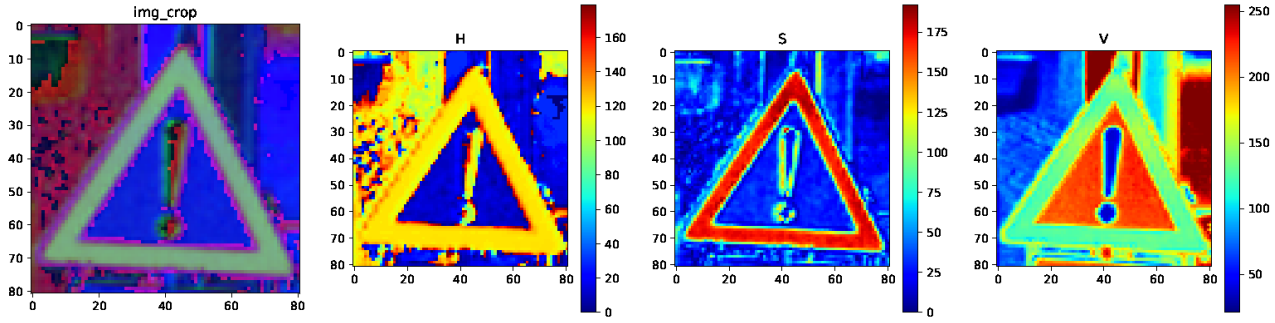

Figura 3: Espacio de color HSV para una señal de peligro.

   - Construir un np.array de tamaño fijo (p.ej. 25x25) en el que los píxeles rojos tengan valor 255 y 0 el resto. Esta matriz será la máscara de color rojo y habrá una diferente para cada clase de señal de tráfico.
   - Se puede construir la máscara media o también la máscara a partir de la imagen media de todas las señales de tráfico de entrenamiento (ver Figura 4).

3 Detección mediante correlación de máscaras:

3.1 Recortar cada ventana detectada por MSER en una imagen de entrada y cambiar su tamaño a uno fijo (p.ej. 25x25 píxeles) con *cv2.resize*.

3.2 Extraer la máscara de color rojo, M, de la ventana redimensionada.

3.3 Correlar M (multiplicar elemento a elemento y sumar), con cada una de las tres posibles máscaras de color rojo de las clases de señal de tráfico. Se puede usar el valor de correlación con cada tipo de señal, para saber cuál es la que mejor encaja con la ventana procesada. Si se pone un umbral también se pueden rechazar ventanas como "no señal" cuando tienen una correlación muy baja con las tres máscaras de señales.

3.4 Si pasa el umbral, establecer el valor de mayor correlación como la clase de señal detectada. Esta será la puntuación, o *score*, que daremos a esa ventana detectada que nos dirá "cómo de parecido es a una señal de tráfico" esa parte de la imagen.

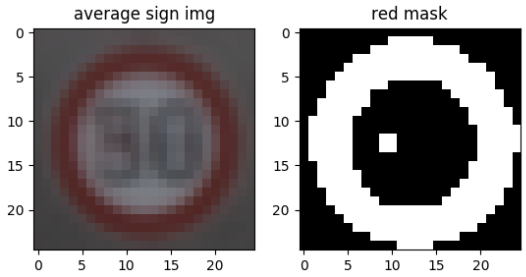

Figura 4: Ejemplo de máscara de píxeles de color rojo (derecha) para señales de prohibición y de imagen media de un tipo de señal (izquierda).

3 Eliminación de detecciones repetidas

A veces en las detecciones de señales con MSER se detecta la parte interna de la señal (p.ej. la parte blanca en las de prohibición) y además la parte externa (el borde), con lo que tendremos más de una detección en cada señal.

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 correlación de las que solapen.
- Elegir la ventana promedio de las que solapen.
- Si hay varias ventanas con clase diferente elegir la clase con mayor correlación.

4 Otros filtros o ideas para detectar señales

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 señales (p.ej. bordes, transformada Hough, detectores de Puntos de Interés tipo esquina, etc).

Importante: Solamente se podrán utilizar técnicas de procesamiento de imagen como las vistas en clase. 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, datos de test y formato de salida

Los **datos de entrenamiento** proporcionados son imágenes .jpg con escenas urbanas en las que aparecen señales de tráfico y el correspondiente fichero gt.txt (con el formato explicado en el apartado 2.1).

Las **imágenes de test**, no disponen de fichero de anotaciones, y se proporcionan en formato .jpg.

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.
  - **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 él las imágenes de test procesadas con los rectángulos de las detecciones en rojo.
- 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. El identificador de clase será siempre 1 en este caso.

  Por tanto, un ejemplo de señal de stop, detectada en la imagen 0000.jpg, con score 80, será:

  ```
  00000.ppm;774;411;815;446;1;80
  ```

- Opcionalmente *main.py*, y si se quiere optar a la máxima nota en cada apartado, deberá escribir otro fichero "resultado_por_tipo.txt", con todas las detecciones para todas las imágenes de test, con el mismo formato explicado en la sección 2.1. La diferencia es que ahora añadiremos el *score* al final de cada línea. Los identificadores de clase serán:
    - 1 para señales de prohibición.
    - 2 para señales de peligro.
    - 3 para señales de stop.

    Por tanto, un ejemplo de señal de stop, tipo 3, detectada en la imagen 0000.jpg, con score 80, será:

    `00000.ppm;774;411;815;446;3;80`

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 (p.ej. Contar cuantas señales de cada tipo ha detectado en cada imagen y cuántas hay).
- Se permitirán grupos de hasta 3 alumnos.

La puntuación de esta práctica corresponde al 35% 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 2 (sección 3 – eliminar detecciones repetidas): **1 punto.**
- Ejercicio 3 (sección 4 – 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