Práctica obligatoria 2

Reconocimiento de objetos

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

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 Reconocimiento de señales de tráfico

En esta práctica vamos a mejorar el detector de señales de tráfico de la práctica 1 utilizando clasificadores sencillos como los vistos en clase. Para ello generaremos un clasificador binario (señal / no señal) para cada uno de los tipos de señales de la práctica 1 (ver Figura 1). Estos clasificadores binarios partirán de una detección MSER como la de la práctica anterior y se aplicarán a cada una de las ventanas propuestas por MSER para descartar o no la ventana como uno de los tipos de señal de tráfico buscados.

Figura 1: Ejemplo de señales de tráfico a detectar

La práctica consistirá en tres pasos:

- **Generar los datos de entrenamiento**. Para ello se necesitará construir un conjunto de imágenes recortadas negativas (sin señal de tráfico) y otro con señales recortadas positivas (con el tipo de señal de tráfico buscada). Para ello se utilizará el detector MSER de la práctica 1 sobre las imágenes de "train" y se comprobará si la ventana detectada solapa o no con una señal de tráfico (usando la información del fichero "gt.txt"). Es importante usar las ventanas provenientes de la detección MSER para que el clasificador "vea" en entrenamiento el mismo tipo de regiones que en ejecución.

- **Entrenar un clasificador binario por señal.** Utilizando las imágenes recortadas y escaladas a un tamaño fijo correspondientes a las detecciones MSER, se entrenará un clasificador binario (señal/no señal) para cada uno de los 6 tipos de señales de la práctica 1.

- **Detección de señales.** En la fase de ejecución se recibirán los 6 clasificadores entrenados y se pasará MSER a la imagen dada. Para todas las ventanas detectadas por MSER se devolverá el tipo de señal con mayor probabilidad o "no señal" si la probabilidad de los 6 clasificadores binarios está por debajo de un umbral.

En la Figura 2 mostramos el resultado del detector de la práctica 1 y la mejora producida con la versión a implementar en esta práctica 2 (MSER + HOG + LDA + Bayes).

Figura 2: A la izquierda, detecciones con 1MSER+COLOR (Práctica 1). A la derecha, detecciones con MSER+HOG+LDA+Bayes (práctica 2)

2.1 Generación del conjunto de datos de entrenamiento.

En el sistema a implementar en esta práctica 2, queremos mejorar el "filtrado" que hicimos en la práctica 1 para las ventanas propuestas por el detector MSER. Para ello, utilizaremos 6 clasificadores binarios, uno por cada tipo de señal. Con estos clasificadores distinguiremos cada tipo de señal de tráfico de las ventanas MSER que pertenezcan a otro tipo de partes de la escena (las que no sean señales).

El primer trabajo a realizar en esta práctica consistirá en identificar el conjunto de ventanas MSER que no son señales de tráfico en las imágenes completas (no las recortadas) en la carpeta "train". Para ello el alumno seguirá los siguientes pasos:

1. Cargar la información sobre ventanas que son señales de tráfico en el fichero *gt.txt*.

2. Pasar el detector MSER sobre las imágenes de train. En esta práctica el primer filtrado debe de consistir también en quedarse con las ventanas casi cuadradas (eliminando las muy alargadas).

3. Identificar las ventanas detectadas por MSER sin solapamiento (o con solapamiento pequeño) con las ventanas sobre las señales de tráfico en *gt.txt*. Estas ventanas serán de la clase negativa para el entrenamiento de los clasificadores binarios. El solapamiento se puede calcular con la medida "Intersection over Union" (IoU), que consiste en el ratio entre el área de la intersección de dos rectángulos sobre el área de la unión de los mismos (1 solapamiento máximo y 0 solapamiento mínimo).

4. Identificar las ventanas con señales de tráfico en *gt.txt*. Estas serán las de la clase positiva para cada uno de los 6 clasificadores binarios a entrenar.

Llegados a este punto caben dos posibilidades: a) recortar las regiones de imagen delimitadas por las ventanas positivas y negativas y guardar en disco para su uso posterior, b) realizar el cálculo de los vectores HoG (ver sección 2.2) y guardar estos descriptores en disco en lugar de las imágenes (tanto los positivos como los negativos) junto con la clase a la que pertenecen. Se recomienda utilizar el 0 como identificador para la clase "no señal" y los números de 1 a 6 para los tipos de señal que vamos a detectar.

2.2 Ejercicio 1. Implementación y pruebas del sistema básico de reconocimiento

La Figura 3 muestra un sistema de reconocimiento básico que funciona para el problema de clasificación señal/no señal de tráfico. Este funcionamiento básico es el que el alumno debe de seguir para construir una aplicación que mejore el detector de señales de tráfico de la práctica 1 (ver resultados en la Figura 2).

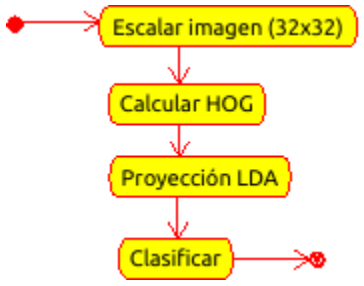

Figura 3: Esquema de funcionamiento del sistema de reconocimiento básico a implementar.

En este ejercicio el alumno deberá implementar el sistema de reconocimiento básico en el que podemos distinguir 3 partes bien diferenciadas:

1. Extracción del vector de características: escalado de la imagen y extracción de HOG (Histograma de Orientación de Gradientes). Usar para ello la clase: cv2.HOGDescriptor y el método "compute" de esa clase para calcular el vector de características HOG de una imagen (ver referencias que explican el HOG al final del enunciado).

2. Reducción de la dimensionalidad con LDA (usar la clase LinearDiscriminanAnalysis del paquete sklearn.discriminant_analysis).

3. Entrenamiento del clasificador Bayesiano con Gaussianas. Hay dos posibilidades en este caso:

    - cv2.ml.NormalBayesClassifier de OpenCV.

    - sklearn.discriminant_analysis.LinearDiscriminanAnalysis ofrece la reducción de dimensionalidad de LDA (con los métodos fit y transform) y además proyección LDA + clasificación con un Bayesiano con Gaussianas (fit y predict o predict_proba).

En este primer ejercicio el alumno deberá utilizar el sistema de reconocimiento básico para entrenar 6 clasificadores binarios:

1. Señales de prohibición vs no señal.

2. Señales de peligro vs no señal.

3. Señal de stop vs no señal.

4. Señal de dirección prohibida vs no señal

5. Señal de ceda el paso vs no señal

6. Señal de dirección obligatoria (la que tenemos identificada en la Figura 1) vs no señal.

Es decir, habrá que recortar los trozos de imagen correspondientes a las ventanas negativas detectadas por MSER (ver sección 2.1) y los correspondientes a ventanas positivas en las imágenes de *train*. Una vez recortados habrá que redimensionar a 32x32 píxeles, calcular su descriptor HOG y llamar al método *fit* de la clase LinearDiscriminanAnalysis. Este proceso se repetirá para cada uno de los 6 clasificadores binarios a entrenar.

Se construirá un clasificador multi-clase (6 señales de tráfico y la clase "no señal") a partir de los 6 clasificadores binarios. Para ello se ejecutará el método *predict_proba* de cada clasificador binario y se dará como la clase señal ganadora a aquella con mayor probabilidad siempre y cuando esta pase un umbral (por ejemplo 0.5). Si la probabilidad de la clase señal ganadora no pasa el umbral se declarará como "no señal".

En este ejercicio **habrá que evaluar el clasificador multi-clase** generado. Para ello se deberá reservar un subconjunto de los datos de entrenamiento como validación (p.ej. un 10% de negativos y un 10% de cada tipo de señal). La evaluación consistirá en calcular las matrices de confusión y otras medidas de rendimiento sobre el conjunto de validación.

2.3 Ejercicio 2. Otras alternativas para el reconocimiento.

El ejercicio 2 consiste en comparar los resultados que ofrece el sistema de reconocimiento con otras alternativas. Entre las posibles alternativas se proponen:

- Otros vectores de características:
  - Los niveles de gris directamente como un vector.
  - Los valores RGB como un vector.
  - Otras medidas disponibles en OpenCV o en skimage.feature (http://scikit-image.org/docs/dev/api/skimage.feature.html). Un vector de características típico en el caso de detección de objetos es son las Haar-like-features.
- Otros algoritmos de reducción de dimensionalidad[1]:
  - Por ejemplo, PCA (Principal Component Analysis) disponible en sklearn (http://scikit-learn.org/stable/modules/decomposition.html).
- Utilizar Otros clasificadores[2]:
  - Euclídeo, KNN o cualquier otro clasificador (supervisado) disponible en OpenCV y/o sklearn. Es importante hacer notar aquí que el clasificador en este caso podría ser multi-clase de partida (p.ej. un KNN) y entrenarlo con 7 clases (la negativa y los 6 tipos de señales).

El código desarrollado para esta parte de la práctica deberá permitir elegir cada uno de los diferentes clasificadores mediante el parámetro correspondiente. Además, la comparativa entre el sistema básico y la alternativa implementada en esta sección quedará reflejada en la memoria de la práctica. La claridad y adecuación en la exposición de las medidas de rendimiento (tasas de acierto, matrices de confusión, gráficos, curvas ROC o de Precisión/Recall, etc.) será valorada en la nota.

Para obtener la máxima puntuación habría que probar con relativo éxito **una alternativa** (entendiendo como alternativa el cambio de cualquiera de los componentes del sistema básico – extracción de características, algoritmo de reducción de dimensionalidad o clasificador).

2.4 Ejercicio 3. Detección de señales de tráfico en imágenes

El ejercicio 3 consiste en unir el sistema de reconocimiento diseñado en el apartado anterior con el detector MSER desarrollado en la práctica obligatoria 1. De esta manera tendremos una mejora en la detección y reconocimiento de las señales que hay en una escena de carretera**.**

---

1. Al usar LDA o PCA como reducción de dimensionalidad con otros clasificadores (distinto del que proporciona la clase LDA de sklearn), implicará que hay que usar el método *transform* para bajar la dimensionalidad antes de entrenar y también al ejecutar los clasificadores.

2. Al usar PCA como reducción de dimensionalidad, implicará que hay que usar el método *transform* para bajar la dimensionalidad antes de entrenar y ejecutar los clasificadores.

Se utilizará el mejor clasificador multi-clase desarrollado en las secciones 2.2 y 2.3 sobre las detecciones MSER pero esta vez **entrenado sobre todos los ejemplos** (sin retirar el 10% de cada clase para validación). De esta manera se clasificará cada ventana MSER en cada uno de los 6 tipos de señales o en "no señal"[3]. Habrá que definir de nuevo el *score* para la detección y en este caso se podrá utilizar la probabilidad del detector binario con mayor probabilidad.

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 en la práctica 1 y se comparará entre el resultado obtenido en la práctica 1, esta práctica 2 y los resultados del sistema implementado por 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_jpg* desde el directorio donde se encuentran los ficheros *resultado.txt* generado y los ficheros "resultado_práctica1_jmbuena.txt" y "resultado_práctica2_jmbuena.txt" proporcionados 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 las prácticas 1 y 2 (Ver Figura 4). Ejemplo de uso:

python3 evaluar_resultados.py –test_path ./test_alumnos_jpg

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

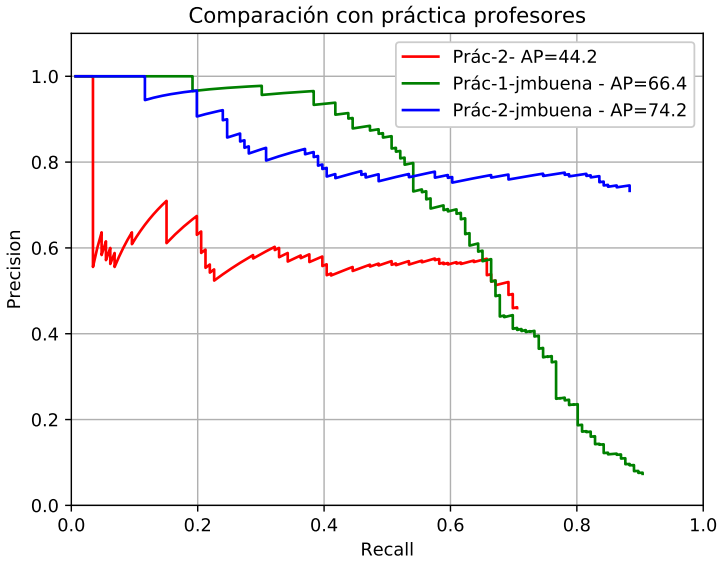

Figura 4: Curva "precision-recall" de un método de detección vs las de la implementación de los profesores (de la práctica 1 y práctica 2). AP es el valor de "Average Precision" para cada curva y una métrica muy utilizada en detección.

3 Se recomienda guardar los clasificadores entrenados en disco utilizando el paquete *pickle* (entrenamos una vez y para ejecutar cargamos de disco los clasificadores entrenados).

3 Datos de entrenamiento, 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). Estos ficheros son los mismos que los de la práctica 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 .jpg. Estos ficheros son los mismos que los de la práctica 1.

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 /un/ejemplo --test_path /un/ejemplo [–classifier HOG_LDA_BAYES]

  - **El primer parámetro** es el directorio donde están los subdirectorios de entrenamiento.
  - **El segundo parámetr**o es el directorio donde están las imágenes a reconocer. Si una imagen tiene un tamaño igual o inferior a 100x100 píxeles se supondrá el caso base (una señal recortada), si el tamaño es mayor a 100x100 (en cualquiera de las dos dimensiones) se supondrá que es una escena real y deberá llamar a la función que detecta señales de tráfico. En ambos casos se procesarán todas las imágenes del directorio de test.
  - **El tercer parámetro** es el tipo de clasificador (con sus alternativas del ejercicio 2) que se implementen: HOG-LDA-BAYES, HOG-PCA-KNN, etc.
- El resultado de *main.py* deberá ser un fichero “resultado.txt”, con una línea por cada resultado de reconocimiento que sea una señal de tráfico. Dicha línea seguirá el siguiente formato:

  ```
  00000.jpg;774;411;815;446;1;0.6
  00008.jpg;0;0;90;90;3;0.5
  ```

  Obsérvese que en el caso de imágenes de señales recortadas solo se genera una línea por fichero si se detecta una señal. Sin embargo, en el en caso de imágenes de escenas reales pueden generarse más de una línea por fichero. Además, si la imagen es de una señal recortada (tamaño igual o inferior a 100x100 píxeles) la ventana de detección devuelta es la imagen completa.

4 Normas de presentación

La presentación seguirá las siguientes normas:

- Su presentación se realizará a través del Aula Virtual**.**
- Para presentarla se deberá entregar un único fichero ZIP que contendrá el código fuente en python y un fichero PDF con la descripción del sistema desarrollado.

- Dicho fichero PDF incluirá una explicación con el algoritmo desarrollado, los métodos de OpenCV y/o sklearn utilizados, copias de las pantallas correspondientes a la ejecución del programa y unas estadísticas correspondientes al resultado de la ejecución del programa sobre las muestras de test. Si se han probado diferentes sistemas de clasificación deberán reflejarse los diferentes resultados que se hayan obtenido.

- Se permitirán grupos de 3 alumnos.

La puntuación de esta práctica corresponde al 35 % de la asignatura. La práctica se valorará sobre 10 y cada parte tendrá la siguiente puntuación:

- Ejercicio 1: **6 puntos.**

- Ejercicio 2: **1 puntos.**

- Ejercicio 3: **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 3.

- 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.

5 Referencias

1. Medida de solapamiento con *"Intersection over Union":*

   https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

2. Explicación del vector de características HoG:

   http://www.learnopencv.com/histogram-of-oriented-gradients/

3. Documentación de del descriptor HOG en OpenCV:

   http://docs.opencv.org/trunk/d5/d33/structcv_1_1HOGDescriptor.html

4. Ejemplo de uso de LDA:

   http://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html