

Tarea 4 EL7008 – Primavera 2020

Detección de personas usando Características tipo HOG

Profesor: Javier Ruiz del Solar

Auxiliar: Patricio Loncomilla

Ayudantes: Juan Pablo Cáceres, Hans Starke, Javier Smith, José Villagrán

Fecha enunciado: Jueves 5 de Noviembre de 2020

Fecha entrega: Miércoles 19 de Noviembre de 2020

(si la semana del 16/11 es de receso, entonces Miércoles 26 de Noviembre de 2020)

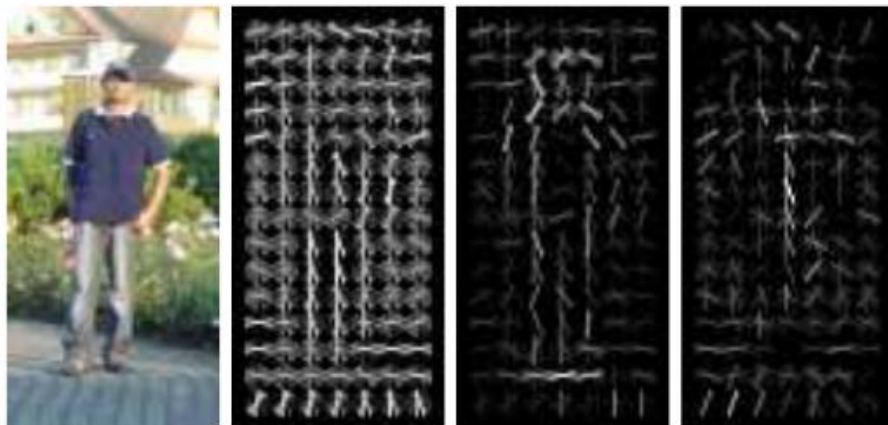
El objetivo de esta tarea es diseñar y construir un sistema de detección de personas, que utilice características tipo *HOG* (Histograms of Oriented Gradients) y clasificadores *SVM* y *Random Forest*. Se sugiere utilizar las librerías OpenCV, numpy y scikit-learn, pues contienen varias de las funcionalidades requeridas en la tarea.

Preparación de Conjuntos de Entrenamiento y Test

Para las tareas de entrenamiento y validación se debe utilizar las imágenes de la base de datos subida a U-Cursos (370 imágenes), la cual incluye 185 imágenes con personas y 185 imágenes sin personas (sillas y autos). Esta base de datos debe ser separada en 70% para entrenamiento (70% de personas, sillas y autos), y 30% para realizar la evaluación.

Extracción de Características

Histograms of Oriented Gradients (HOG)¹: El método procesa imágenes (recortadas) de un tamaño fijo (ejemplo: 64x128). Se calculan gradientes en la imagen y se dividen espacialmente usando 8x16 celdas. En cada celda se calcula un histograma de orientación del gradiente con 9 componentes (orientaciones). Finalmente se hace una normalización por bloques, y se genera el histograma final concatenando los histogramas. Un ejemplo de histogramas de gradientes orientados se muestra en la figura 1.



¹ Ver paper: Navneet Dalal, Bill Triggs. "Histograms of Oriented Gradients for Human Detection".

Figura 1: Imagen de prueba, HOG, HOG ponderado por los pesos positivos del SVM, y HOG ponderado por los pesos negativos del SVM (figura extraída del paper original)

Clasificación

Se debe entrenar un clasificador SVM y un clasificador *Random Forest* para diferenciar entre personas y no-personas utilizando las características HOGs. Scikit-learn tiene una implementación de SVM y de *Random Forest*.

Se pide:

1. Implementar una función en python que transforme la imagen a escala de grises, al tipo `np.float32`. Además debe redimensionar la imagen a tamaño 64x128
2. Implementar en python una función que reciba una imagen y calcule sus gradientes (se puede reutilizar código)
3. Implementar en python una función que, a partir de los gradientes, calcule las características HOG usando 8x16 celdas (la salida debe ser un arreglo de numpy de dimensión 8x16x9)
4. Implementar en python el *block normalization* para el histograma. Para implementar esto, se deben formar bloques de 2x2 celdas (con traslape). Cada bloque se debe transformar en un vector de 1x36, el cual se debe normalizar. El vector de características final (de tamaño 1x3780) se obtiene concatenando los vectores normalizados de cada bloque.
5. Extraer características HOG de cada imagen del conjunto de entrenamiento. Se recomienda aplicar un `StandardScaler` (usando scikit-learn) a las características para normalizarlas. El `StandardScaler` se debe entrenar usando sólo las características de entrenamiento.
6. Entrenar un SVM (usando scikit-learn) con las características extraídas a cada imagen redimensionada del conjunto de entrenamiento. Se debe elegir un kernel, y usar grid search.
7. Entrenar un clasificador Random Forest (usando scikit-learn) con los parámetros por defecto (no es necesario especificar los parámetros) usando el mismo conjunto de entrenamiento.
8. Realizar las pruebas correspondientes con el conjunto de evaluación. Calcular la matriz de confusión del clasificador SVM y del Random Forest. ¿Qué tan bien funciona cada clasificador? ¿Cuál funciona mejor? ¿Cómo se pueden mejorar los resultados?
9. Documentar cada uno de los pasos anteriores en el informe

El código entregado debe ejecutar el entrenamiento, usando tanto SVM como Random Forest, y calcular las matrices de confusión correspondientes usando el conjunto de evaluación. Se debe usar un jupyter notebook (idealmente usando collaboratory).

Los informes, los códigos y el archivo README.txt deben ser subidos a U-Cursos hasta las 23:59 horas del día miércoles 19 de noviembre..

Importante: La evaluación de esta tarea considerará el correcto funcionamiento del código, la calidad de los experimentos realizados y de su análisis, las conclusiones, así como la prolijidad y calidad del informe entregado.

Nota: En collaboratory, se recomienda subir el .zip con las imágenes y ejecutar:

```
!unzip archivo.zip
```

Nota: El informe de la tarea debe ser subido a turnitin

Nota extra: Dado que la ejecución del código de HoG puede ser lenta en Python, se recomienda usar numba en las funciones que hacen operaciones píxel a píxel:

```
from numba import jit

@jit(nopython=True)
def computeHog(gradx, grady):
    # Cuerpo de la función en Python normal
```