# Extracción de características en imágenes

**DEADLINE: 02/04/17**

## **Práctica 1** – Descriptores

1. Implementación de un descriptor, LBP o HOG.
2. Cálculo de características.
3. Entrenamiento y clasificación.
4. Entrega.

Dudas:

Juan G. Serra: jgserra@decsai.ugr.es

### 1. Implementación de un descriptor.

Elegid uno de los dos descriptores explicados en clase para implementar.

#### A – Local Binary Patterns (LBP)

Se pide la implementación de las siguientes funciones, respetando nombres y argumentos de entrada y salida:

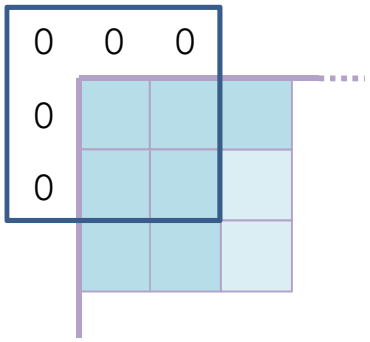
```
function lbp = LBPu(im)

    % im:      imagen de niveles de gris

    % lbp:     imagen lbp de códigos uniformes (R=1, N=8)
                % códigos: 0-58
```

Esta primera función se encarga de calcular los códigos LBP uniformes [0,58] correspondientes a cada uno de los píxeles. Devuelve una matriz del mismo tamaño que la de entrada. Debe aceptar cualquier tamaño de imagen de entrada.

Para resolver el problema en los bordes en la imagen, podemos utilizar *zero-padding*, es decir, suponer que nuestra imagen viene rodeada por un marco de un píxel de anchura con valor cero. Se ilustra el problema en la figura.



```
function x = lbp_features(patch)

    % patch:  imagen lbp de un parche/ventana

    % x:      vector de rasgos de la ventana (histograma por bloques)
               % tamaño de bloque 16x16
               % solapamiento 8 píxeles
```

Esta función recibe una matriz con los LBP de una ventana de tamaño 128x64 y se encarga de calcular el descriptor según el proceso visto en clase, respetando tamaño de los bloques y el solapamiento indicado. Para la concatenación de los histogramas normalizados (de forma que todos sumen uno), se recorrerá la ventana de izquierda a derecha y de arriba abajo.

- Recordad que el vector resultante deberá tener 6195 componentes.
- Se puede añadir algún parámetro de entrada a las funciones si se cree necesario.

Las siguientes funciones pueden resultar de utilidad:

```
help imread             % comando para mostrar ayuda
doc imread              % ayuda más extensa
str = dec2bin(d,n)      % decimal a binario
d = bin2dec(binarystr)  % binario a decimal
A = zeros(m,n);         % matriz de ceros mxn

im(:,:,1)               % banda R
```

## B – Histogram of Oriented Gradients (HOG)

Si se opta por la implementación del descriptor de Histogramas de Gradientes Orientados (HOG), se deberá implementar una función principal con la siguiente cabecera:

```
function hog = hog_features(patch)

    % patch:      parche de tamaño 128 x 64
    % hog:        cector de rasgos hog del parche dado
                  % Tamaño de  celda: 8x8
                  % N° de intervalos: 9
                  % Tamaño de bloque: 2x2
```

Nótese que para simplificar la implementación se han fijado todos los parámetros: tamaño del parche, tamaño de celda, número de intervalos y tamaño de bloque.

Como vimos en la clase de teoría, el cálculo del descriptor se puede dividir en 3 subtareas:

1- Cálculo del gradiente
2- Cálculo de los histogramas
3- Normalización de bloque

Para la práctica se deberá implementar cada una de estas subtareas en 3 funciones distintas, que serán llamadas por "hog_features" y que tendrán las siguientes cabeceras:

```
function [magnitud,orientacion] = gradiente(patch)
      % patch:    parche de tamaño 128 x 64
      % magnitud: matriz de tamaño 128 x 64 con la magnitud del
                  gradiente de cada pixel
      % orientacion: matriz de tamaño 128 x 64 con las orientaciones
                     del gradiente de cada pixel (PISTA: usar función
                     artan2 de MATLAB)
```

```
function histogramas = calcula_histogramas(magnitud,orientacion)
      % magnitud: matriz de tamaño 128 x 64 con la magnitud del
                  gradiente de cada pixel
      % orientacion: matriz de tamaño 128 x 64 con las orientaciones
                     del gradiente de cada pixel

      % histogramas: Matriz 3D de tamaño "nº de celdas en la dirección
      %              vertical x nº de celdas en la dirección
      %              horizontal x nº de intervalos"
```

```
function hog = norm_bloque(histogramas)
      % histogramas: Matriz 3D de tamaño "nº de celdas en la dirección
      %              vertical x nº de celdas en la dirección
      %              horizontal x nº de intervalos"
      % hog: es un vector con del descriptor final calculado de
      %      dimensión ???
```

## 2. Cálculo de características.

Se deberá aplicar el descriptor implementado sobre una base de datos de imágenes. Estas imágenes son una selección de la base de datos pública *INRIA Person Dataset*. Podéis descargaros las imágenes para la práctica en la carpeta del tema 2 del curso.

Las imágenes se encuentran convenientemente divididas en carpetas separadas para las muestras que se usarán para entrenamiento y las que se usarán para la evaluación

- data
  - train
    - background [2390]
    - pedestrians [1916]
  - test
    - background [600]
    - pedestrians [500]

del clasificador. La estructura de carpetas es la siguiente:

- Se muestran entre corchetes el número de imágenes en cada carpeta.

Las líneas de cálculo que se presentan a continuación muestran un ejemplo para la lectura secuencial de las imágenes de un directorio mediante Matlab:

```
path = '.train/pedestrians/';
lista = dir([path '*.png']);

for k = 1:numel(pos)
    im = imread([path lista(k).name]);
end
```

### 3. Entrenamiento y clasificación.

Se utilizarán las 4306 imágenes de la carpeta *train* para entrenar un clasificador a elección del alumno. Recordad que tendréis que añadir manualmente las etiquetas de clase.

Una vez entrenado el modelo se evaluará tanto sobre el propio conjunto de datos de entrenamiento como sobre el de evaluación (1100 imágenes).

- Este proceso puede realizarse en cualquier software.

### 4. Entrega.

Se deberán subir a *prado2* un fichero de la forma ApellidosNombrePr1.zip que contenga:

- Archivos .m de las funciones que implementen el descriptor escogido.
- Script que, utilizando las funciones anteriores, lea las imágenes y calcule sus características, creando las matrices con los datos y etiquetas que se utilizarán para el entrenamiento y evaluación del clasificador.
- Breve documento comentando el clasificador elegido y presentando los resultados obtenidos, por ejemplo, las matrices de confusión en *train* y *test*.