Actividad de laboratorio #2

Estudiante: Rodrigo Muñoz
22 de marzo de 2016

1. Introducción

Este documento presenta la implementación y análisis de algoritmos de clasificación de personas usando información periocular. En particular se implementa un clasificador usando como características histogramas LBP.

2. Algoritmo de análisis de textura LBP

El Código 1 muestra la implementación del algoritmo LBP usado para obtener las imágenes mostradas en la Figura 1

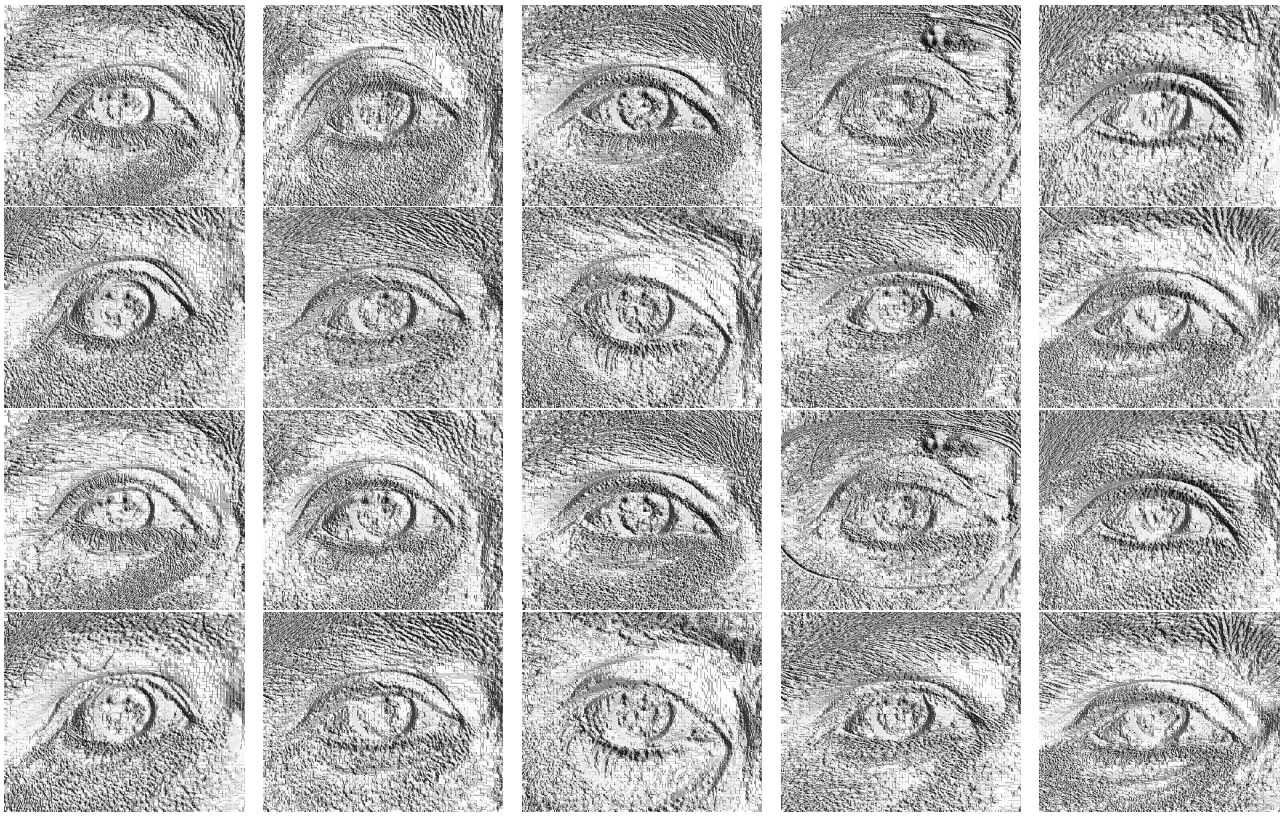

Figura 1: Imágenes con transformada LBP.

La matriz empleada en el algoritmo correspondió a la matriz:

$$\begin{matrix} 1 & 2 & 4 \\ 128 & 0 & 8 \\ 64 & 32 & 16 \end{matrix} \tag{1}$$

Las imágenes obtenidas con este algoritmo corresponden a imágenes en escala de grises donde los cambios intensidad corresponden a los cambios de textura de la imagen original. De esta forma se resaltan características como los bordes del iris y de los ojos. Si el cambio de textura es suave o nulo, la imagen LBP se mantiene blanca.

3. Extracción de carácteristicas

Las características empleadas se obtienen concatenando histogramas de distintas ventanas de la imagen LBP, en este caso se consideraron 30 ventanas de 100x100 píxeles, donde cada una ellas aporta un histograma de 59 puntos. Esto genera un vector de 1770 características por imagen.

Para obtener las ventanas se utilizo la función de `mat2cell` e `hist` para la obtención del histograma.

4. Medidas de distancia

Para obtener el *match* entre un elemento de prueba y la base de datos se emplearon medidas de distancia. En particular se implementaron dos medidas:

$$\begin{aligned} d_1(x,y) &= \sum_{i=0}^{n} |x_i - y_i| \\ d_2(x,y) &= \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2} \end{aligned}$$

5. Construcción de la base datos

Para construir la base datos incorporan las cada uno de los vectores de características a una matriz, dando como resultado una matriz 10x1770, para un set de 10 imágenes.

6. Reconocimiento con distancia d_1

Para el reconocimiento se considera la distancia que del vector de características de la imagen de prueba con cada vector almacenado en la base de datos, el *match* se asocia con el elemento con menor distancia. La Tabla 1 muestra el resultado de aplicar este método usando la distancia d_1.

Como se muestra en la Tabla 1 solo existe un error de clasificación, por lo que el clasificador tiene un desempeño del 90 %.

Cuadro 1: Resultados de clasificación con distancia d_1

Clase	Resultado
1	1
2	2
3	3
4	4
5	5
6	6
7	4
8	8
9	9
10	10

Cuadro 2: Resultados de clasificación con distancia d_2

Clase	Resultado
1	1
2	2
3	3
4	4
5	5
6	6
7	4
8	8
9	9
10	10

7. Reconocimiento con distancia d_2

La Tabla 2 muestra los resultados obtenidos al implementar el mismo clasificador usando, esta vez, la distancia d_2.

Notamos que los resultados son los mismos, de echo el clasificador erra en la misma imagen, lo que permite concluir que las figuras 4 y 7 no presentan una diferencia significativa en textura lo que imposibilita su correcta clasificación usando transformada LBP.

8. Manejo de errores

Hasta el momento el clasificador siempre entrega una respuesta, pues siempre existe un vector con distancia menor. Por ejemplo, esto podría ocasionar graves problemas en un sistema de control de acceso usando esta implementación.

Para resolverlo se plantea el uso de un umbral mínimo de distancia, tal que si un *match* que obtenga un valor menor a este será clasificado como inexistente en la base de datos.

9. Anexos: Implementación de algoritmos en MATLAB

Listing 1: Obtención de imagen LBP

```
function [ result ] = lbp(image_adress)
    % Load image
    raw_image = imread(image_adress);
    % Check gray scale
    if(length(size(raw_image)) ~= 2)
        raw_image = rgb2gray(raw_image);
    end
    % Get size
    [M,N] = size(raw_image);
    lbp_img = zeros(M,N);
    % LBP weight
    weight = [1 2 4; 128 0 8; 64 32 16];
    for i = 2:M-1
        for j = 2:N-1
            mask = zeros(3,3);
            for mi = 1:3
                for mj = 1:3
                    if(raw_image(i-(2-mi),j-(2-mj)) >= raw_image(i,j))
                        mask(mi,mj) = weight(mi,mj);
                    end
                end
            end

            lbp_img(i,j) = sum(sum(mask));
        end
    end
    lbp_img = uint8(lbp_img);
    result = lbp_img;
end
```

Listing 2: Obtención del vector de características

```
function [ out_vector ] = get_vector( lbp_img )
    % Cut image
    sub_img = lbp_img(2:501,2:601);
    % Create windows
    windows = mat2cell(sub_img, [100 100 100 100 100],[100 100 100 100 100 100]);
    out_vector = [];
    for i = 1:5
        for j = 1:6
            % Calc histogram
            temp_hist = hist(double(windows{i,j}(:)),59);
            out_vector = [out_vector temp_hist];
        end
    end
end
```

Listing 3: Creación de base datos

```
function [ db ] = get_db( folder )
    names = get_files(folder);
    disp(names);
    db = [];
    for i=1:length(names)
       lbp_img = lbp(names{i});
       v = get_vector(lbp_img);
       db = [db; v];
    end
end
```

Listing 4: Manejo de archivos

```
function [ file_list ] = get_files( folder )
    dir_data = dir(folder);
    dir_index = [dir_data.isdir];
    file_list = {dir_data(~dir_index).name}';
    for i=1:length(file_list)
        file_list{i} = strcat(folder,'/',file_list{i});
    end
end
```

Listing 5: Script principal

```
%% Options
DISTANCE_TYPE = 1;
%% Create DBs
disp('Extracting features...');
tic
db_gal = get_db('ojos_gal');
db_test = get_db('ojos_test');
toc
disp('[OK]');
%% Compare imgs
d = [10,10];
for n=1:10
    for m=1:10
        d(n,m) = vector_d(db_gal(n,:),db_test(m,:),2);
    end
end
[~,res]=min(d);
res = [1:10;res]';
disp('Results:');
disp('Class  | Result');
disp(res);
```