Tarea 3 EL7008 – Primavera 2018

Clasificación de Raza

El objetivo de esta tarea es diseñar y construir un sistema de clasificación de raza, que utilice características tipo *Histogramas LBP* y clasificador estadístico *SVM* o *Random Forest.* Se sugiere utilizar la librería OpenCV, pues contiene 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 (1000 imágenes), la cual incluye 200 imágenes con caras de personas correspondientes a cinco distintas razas (Asian/Black/Indian/White/Others). La base de datos entregada es un subconjunto de UTKFaces. La base de datos entregada debe ser separada en 70% para entrenamiento y 30% para realizar la evaluación.

**Extracción de Características**

**Histogramas LBP[[1]](#footnote-1)**: El método extrae las características utilizando histogramas de imágenes LBP. Se definen tres niveles diferentes de localidad: a nivel de píxel, a nivel regional y a nivel global. Los dos primeros niveles de la localidad se realizan dividiendo la imagen LBP (imagen original con trasformada LBP) del rostro en pequeñas regiones; se extraen histogramas (características a ser utilizadas). Estos histogramas se utilizan para una representación eficiente de la información de textura (ver Figura 1). El nivel global de localidad, es decir, descripción del rostro, se obtiene concatenando histogramas LBP locales. En este caso deben utilizar las características (ver paper) y dividir en 4 regiones/cuadrantes las caras.

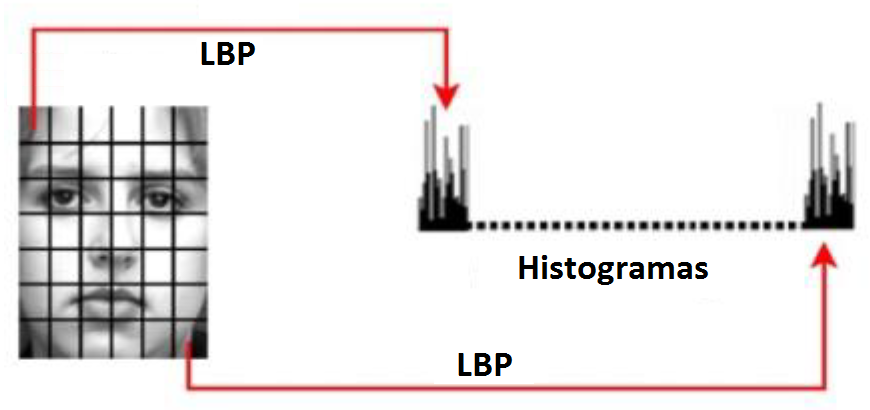


Figura 1: Histogramas LBP.

**Clasificación**

Se debe entrenar un clasificador SVM que pueda diferenciar entre hombres y mujeres utilizando los Histogramas LBP. Además, se debe entrenar un *Random Forest* con los parámetros por defecto. OpenCV tiene una implementación de SVM y una de *Random Fores*t. La interfaz de programación va a depender de la versión de OpenCV que se use (OpenCV 2 o OpenCV 3).

Se pide:

1. Implementar en C++ una función que reciba una imagen, y retorne la imagen con la transformada LBP uniforme . Mostrar en el informe un ejemplo de una cara a la que se le aplique la transformada implementada.
2. Implementar en C++ Histogramas LBP (ver el paper para más detalles), utilizar las características LBP uniformes y dividir las caras en 4 regiones (división de 2 x 2 regiones).
3. Extraer los Histogramas LBP uniformes de cada cara del conjunto de entrenamiento, considerando dos clases: “Asian” y “Black”.
4. Entrenar un SVM (*cv::CvSVM* en opencv2*, cv::ml::SVM* en opencv3) con las características extraídas a cada cara del conjunto de entrenamiento. Elegir un kernel y probar diferentes configuraciones de parámetros.
5. Entrenar un clasificador *Random Forest* (*cv::CvRTrees* en opencv2, *cv::ml::Rtrees* en opencv3) con los parámetros por defecto usando el mismo conjunto de entrenamiento.
6. Realizar las pruebas correspondientes con el conjunto de evaluación. Calcular la tasa de error del clasificador SVM y del *Random Forest* (suma de la diagonal). ¿Qué tan bien funciona cada clasificador? ¿Cómo se pueden mejorar los resultados?
7. Analizar los resultados obtenidos con los clasificadores entrenados.
8. Repetir los puntos 4-7, usando las cinco clases: “Asian”, “Black”, “Indian”, “Others”, “White”.
9. Documentar cada uno de los pasos anteriores en el informe

El código entregado debe ejecutar un entrenamiento y mostrar evaluación del clasificador entrenado. Se debe entregar un único archivo llamado main.cpp. Debe entregar las instrucciones para su ejecución en un archivo README.txt.

Los informes, los códigos y el archivo README.txt deben ser subidos a U-Cursos hasta las 23:59 horas del día lunes 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 esta tarea, el problema corresponde a clasificación multiclase. OpenCV es capaz de resolver este tipo de problemas, pero se recomienda usar los siguientes tipos de datos:

Matriz con características de entrenamiento: CV\_32FC1

Matriz con características de prueba: CV\_32FC1

Matriz con etiquetas de entrenamiento (1 columna): CV\_32SC1

La función predict() toma como entrada una fila conteniendo características de tipo CV\_32FC1, y devuelve un número correspondiente a la clase. En el caso de *random forests*, se le debe entregar sólo una fila a la vez.

Nota: Es necesario usar CV\_32SC1 para la matriz de etiquetas de entrenamiento para que OpenCV detecte que el problema es multiclase. Las características deben ser CV\_32FC1

1. Ver paper: T. Ahonen, A. Hadid, M. Pietikainen, “Face recognition with local binary patterns” [↑](#footnote-ref-1)