Informe Textones: Visión por Computador

Emilio Botero
Universidad de los Andes
e.botero10@uniandes.edu.co

1.. Introducción

La textura es un fenómeno omnipresente, fácil de reconocer pero difícil de definir. Típicamente, definir un componente de una imagen como textura o no depende de la escala en la cual se mira. Una hoja, por ejemplo, puede verse como un objeto si es el único elemento en la imagen; pero las hojas de un árbol se reconocen como textura. La textura es una parte importante del reconocimiento de objetos porque esta da información acerca de propiedades del material [1]: podemos distinguir madera de metal porque la madera se ve arrugada, mientras que el metal se ve liso. Si pudiéramos reconocer textura en imágenes, podríamos decir lo que está contenido en ellas. ¿Cómo, entonces representamos la textura en un objeto? Una posible manera de lograr esto es cuantizando vectores de respuesta a un banco de filtros [2]. Podemos pensar que la textura consiste principalmente en elementos básicos repetidos (la textura de una piedra, por ejemplo, puede pensarse como muchos puntos circulares grises). Necesitamos entonces una representación que pueda distinguir entre texturas de manera significativa, aún si no podemos definir exactamente los elementos básicos repetidos (conocidos como textones). Si asumimos que estos textones se componen de subelementos genéricos como puntos o barras, podemos diseñar filtros similares a estos puntos y barras, de tal manera que la respuesta a un filtro particular sea alta cuando es parecido al subelemento, y baja cuando no. Esto equivale a diseñar un banco de filtros: una serie de filtros consistiendo de barras y puntos de diferentes tamaños y orientaciones [1].

Al filtrar una imagen con este banco de filtros, entonces, obtenemos un arreglo de NxF, donde N es el número de píxeles en la imagen y F es el número de filtros utilizados, de tal manera que el arreglo representa las respuestas de cada píxel a todos los filtros. A este arreglo le podemos aplicar kmeans: los centroides de las respuestas son nuestros textones, y tenemos un diccionario de textones [2]. Así, representar la textura en una nueva imagen resulta simplemente de asignarle a los píxeles en la nueva imagen (filtrados por el mismo banco de filtros) el texton más cercano según una distancia apropiada[1].

En este trabajo el objetivo es clasificar imágenes de prueba según su textura. Para esto, construimos un diccionario de textones, entrenamos dos algoritmos (Nearest Neighbour y Random Forests) con los histogramas de textones en imágenes de entrenamiento y probamos nuestro clasificador en las imágenes de prueba, para reportas así la matriz de confusión para cada clasificador.

En particular, el clasificador Nearest Neighbour parte del diagrama de Voronoi para los datos en el espacio de representación (obtenido por k-means), y dado un ejemplo particular, le asigna la categoría del centroide más cercano en el espacio de representación, según una distancia pertinente.

El clasificador de Random Forests pasa un número aleatorio de descriptores (las respuestas a filtros, en este caso) por nodos en árboles, correspondientes a pruebas organizadas de manera jérarquica. En ese sentido, el objetivo del entrenamiento es encontrar los parámetros óptimos para las "split functions" en cada nodo, de tal manera que explican las etiquetas de los descriptores [3] en la base de datos de entrenamiento.

2.. Metodología de pruebas y construcción de diccionario

Utilizamos una base de datos de textura, en la cual hay 25 clases de textura, con 30 ejemplos en un set de entrenamiento y 10 ejemplos en un set de test para un total de 1000 imágenes. Todas las imágenes están en blanco y negro y tienen un tamaño de 640x480 píxeles [4]. En la base de datos se representan texturas diversas como madera, textiles, vidrio, agua, entre otras.

Teniendo en cuenta que la base de datos comprende imágenes con diferentes texturas, ¿cómo construimos el diccionario de textones? En principio debemos construirlo de tal manera de comprenda todas las texturas. Si concatenamos todas las texturas en una sóla imagen obtenemos un arreglo de $640x(750*480)$, una dimensión absurda sobre la cual trabajar. Podríamos concatenar un número de imágenes menor, digamos 2 o 3 imágenes por textura, pero estaríamos limitando gravemente la información disponible, con lo que podríamos afectar el desempeño de nuestro clasificador de texturas.

El método propuesto es el siguiente: dadas las respuestas

al banco de filtros para todas las imágenes en la base de datos de entrenamiento, submuestreamos 5000 píxeles de cada imagen de manera aleatoria y construimos un arreglo de $(750 * 5000)xfb$, donde fb es el número de filtros el banco. A este arreglo le aplicamos k-means y obtenemos así nuestro diccionario de textones.

¿Cuántos filtros utilizamos, y cómo los diseñamos?¿Cuántos textones? Las respuestas a estas preguntas determinarán el desempeño de nuestra solución. No hay una respuesta única para alguna. Uno podría, sin embargo, diseñar filtros que sean, de alguna manera, subelementos "básicos" (como especificamos en la introducción). Por ejemplo, sería difícil encontrar elementos de textura en una imagen que no tuvieran bordes o puntos [1]. Podríamos entonces diseñar filtros de bordes y puntos orientados. Efectivamente, construimos 32 filtros de esta manera: a partir de 8 orientaciones diferentes, construimos filtros pares e impares simétricos a dos escalas diferentes, donde los filtros pares simétricos son las segundas derivadas de Gaussianas, y los impares simétricos son sus transformadas de Hilbert [2].

Respecto al número de textones, decidimos empíricamente utilizar 50. En últimas, no hay una manera teórica de responder esta pregunta: la idea es que variaciones en respuestas al banco de filtros (dadas por las diferentes orientaciones de los filtros) caigan en el mismo texton, para una misma textura. Esto es porque un texton es finalmente un atributo local, y la textura se repite localmente. Según el tipo de problema y la base de datos, la mejor idea es probar distintos números de textones para ver cuál es el mejor. En este caso, probar 50 textones resultó en matrices de confusiones con un gran número de verdaderos positivos para varias categorías.

Para calcular la distancia entre histogramas, utilizamos una medida de distancia Chi-cuadrado, dada por

$$d(H_1, H_2) = \sum_{i=1}^{I} \frac{(H_1(i) - H_2(i))^2}{H_1(i)}$$

donde I es el número de bins en el histograma y H_1 y H_2 son los histogramas a comparar. Finalmente, el número de árboles utilizado para la clasificación con Random Forests fue de 100, determinado empíricamente por su buen resultado.

3.. Resultados

A continuación se visualizan las matrices de confusión resultantes para ambos métodos de clasificación. Para cada categoría de textura en la base de datos de test había 10 imágenes: las siguientes figuras muestran el número de imágenes correctamente clasificadas según una escala de niveles de gris donde blanco corresponde a 10 imágenes clasificadas correctamente y negro a ninguna.

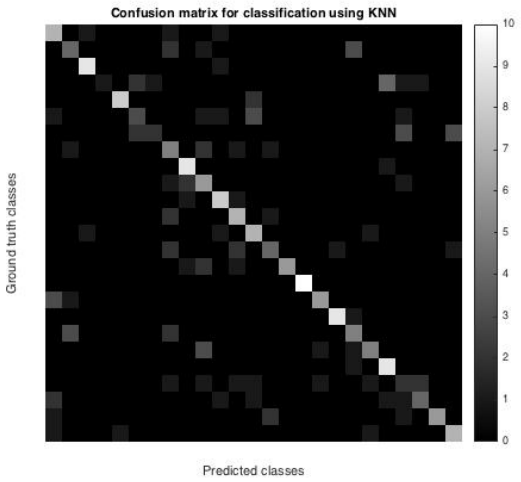

Figura 1. Matriz de confusión para clasificador KNN

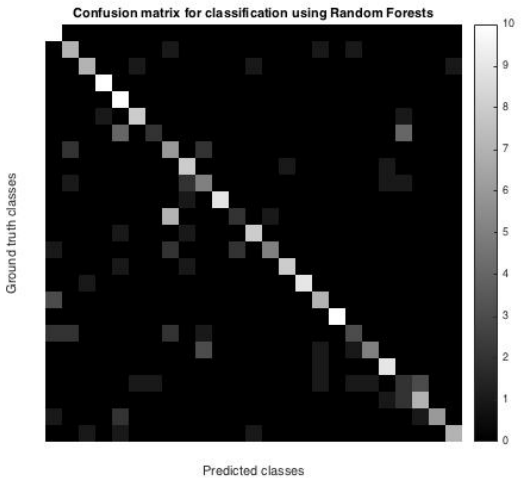

Figura 2. Matriz de confusión para clasificador Random Forests

Adicionalmente, la precisión de los clasificadores es del 59.6 % y 68 % para KNN y Random Forests, respectivamente, definiendo la precisión (de manera algo arbitraria) como TP/P donde TP es el número de imágenes correctamente clasificadas (en la diagonal de la matriz de confusión) y P es el número total de imágenes. Podemos notar que en ambas gráficas hay varias categorías con niveles de gris claros o blancos, con lo que observamos que para esas categorías los clasificadores predijeron correctamente la textura. Para KNN, sin embargo hay también bastantes errores, como se ve en la gran cantidad de puntos (oscuros) a ambos lados de la diagonal. Esto no es tan evidente en el clasificador de Random Forests.

4.. Discusión

Podemos ver, entonces, que el clasificador de Random Forests funciona mejor que KNN. Esto es de esperarse, ya que KNN es un clasificador bastante sencillo, en el cual en realidad no se aprende ningún modelo, sino que se asignan etiquetas según la distancia a centroides. Por lo contrario, los Random Forests aprenden un modelo a partir de los descriptores y optimizan parámetros de clasificación en split functions que explican lo mejor posible las etiquetas de cada descriptor. En general, sin embargo, ambos métodos funcionan bastante bien, clasificando correctamente más de la mitad de las veces. Las categorías 8 y 12 mostraron más confusión para el clasificador de Random Forests, mientras que las categorías 4 y 7 mostraron más confusión en KNN. Esto corresponde a las categorías de granito, piedras, madera 1 y agua, respectivamente. Todas estas categorías tienen en común que su textura es muy uniforme y no presenta respuestas relativamente altas a cambios en orientaciones de los filtros.

El tiempo de clasificación de los modelos es en realidad muy corto: el único punto computacionalmente intensivo del algoritmo es la construcción del diccionario de textones y la asignación de textones a las imágenes de entrenamiento y prueba, que tomó 5 horas. A partir de ahí, entrenar y correr ambos clasificadores no toma más de 10 segundos por clasificador.

Finalmente, el método propuesto tiene una limitación principal en la cantidad de memoria que usa. Una vez se obtuvieron estos resultados, se intentó correr nuevamente con 10000 píxeles aleatoriamente escogidos y 75 textones, lo que agotó la memoria en una máquina virtual compartida de 1.5 TB y 64 GB de RAM. La base de datos tiene una asimismo una gran limitación en tanto todas las texturas comprenden toda la dimensión de la imagen, es decir, las imágenes son muy particulares y limitan el modelo aprendido a únicamente ese tipo de imágenes. En realidad, una imagen cualquiera presenta textura en ciertas regiones, no en todas, por lo que fuera de la base de datos, el modelo aprendido probablemente no sirva de mucho.

Para concluir, entonces, vale la pena pensar sobre cómo mejorar este método para clasificar textura. Una primera idea sería aumentar el número de píxeles submuestrados, y variar el número de textones para encontrar el óptimo. Valdría la pena también evaluar la construcción del diccionario con GMM en vez de k-means, para permitir clusters no esféricos. Otra posibilidad, en una base de datos más real (con regiones con y sin textura) sería considerar los histogramas de textones en ventanas locales. Esto permitiría la identificación de textura y una posible clasificación semántica de la imagen en términos de su contenido. Por último, se podrían utilizar más escalas de filtros, más orientaciones, e incluir filtros independientes a las orientaciones.

Referencias

[1] D. Forsyth and J. Ponce, "Computer vision." Prentice Hall, 2012, pp. 164–174.

[2] J. Malik, S. Belongie, T. Leung, and J. Shi, *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[3] A. Criminisi and J. Shotton, *Decision forests for computer vision and medical image analysis*. Springer, 2013.

[4] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1265–1278, 2005.