TEXTONES Y CLASIFICACIÓN DE IMÁGENES CON NEAREST NEIGHBOUR Y RANDOM FOREST.

Juan Pablo Moreno Ortiz

Maestría en Ingeniería Biomédica

Universidad de los Andes

INTRODUCCIÓN

Dentro de las características que pueden ser extraídas de una imagen y que representan su contenido se encuentra la textura, esta permite diferenciar y asociar imágenes dentro de una colección de varias imágenes. Esta característica puede ser cuantificada y utilizada por los sistemas informáticos a partir del uso de los textones, estos representan los patrones de textura de las imágenes al cuantificar los vectores de respuesta de partes de la imagen contra un banco de filtros con formas, tamaños y orientaciones variables, estos vectores de respuesta se agrupan con k-means, cada grupo que recolecta estas respuestas es un texton. Una imagen puede ser representado de estas unidades representativas y puede ser representada por la respuesta a una serie de textones, de forma gráfica esto se ve como el mapa de textones de la imagen; a partir de esto se generan los histogramas de textones que pueden ser comparados con histogramas de imágenes cuyo contenido es conocido, de esta forma se puede cuantificar el parecido entre las imágenes de entrenamiento y las imágenes de test dentro de un clasificador, de acuerdo a la similitud entre los histogramas de una imagen y los histogramas de la categoría se puede asociar una imagen, cuyo contenido se desconoce, a una categoría de imágenes conocidas.

MATERIALES Y METODOS

***Base de datos.***

La base de datos fue descargada de la página del grupo de investigación en visión por computador y robótica (PONCE) de la universidad de Illinois, esta colección de imágenes se encuentra dividida en un grupo de entrenamiento compuesto por 750 imágenes y otro grupo de test con 250 imágenes, para el entrenamiento se cuenta con 30 imágenes por categoría, para el otro grupo se cuenta con 10. La colección compuesta por imágenes de 25 categorías distintas cuenta para cada imagen con la clasificación en el título. Estas imágenes se encuentran en formato “jpg” en escala de grises con un tamaño de 640x480 pixeles. Esta base de datos incluye un archivo de texto con los nombres de las 25 categorías entre las que se encuentran imágenes de cortezas de árboles, de madera, agua, granito, ladrillos, tela, tapicería entre otras.

***Textones.***

Para generar los textones primero se tiene que generar el banco de filtros, en el script example.m que fue provisto por el docente se utiliza una serie de funciones incluidas en la carpeta lib, para crear el banco de filtros se utiliza la función fbCreate, que generará el banco de filtros, estos serán simétricos pares e impares. Para asegurarse de detectar las texturas a diferentes orientaciones y tamaños de las formas en la imagen los filtros también se aplican variando la orientación de su eje y la escala del tamaño, por defecto los filtros generados contarán con 8 orientaciones y 2 escalas de tamaño. Dentro de la función fbCreate se utiliza la función oeFilter donde se calcula el tamaño del filtro según el valor de sigma y de una constante Support, se calculan los límites de muestreo, la cantidad de muestras y la frecuencia de muestreo, se evalúa la función y se dispone el código para acumular las muestras de cada bin.

Una vez se cuenta con el banco de filtros se definen las imágenes de muestra con las que se generará los textones, en este punto hay que tener cuidado con la cantidad de datos que debe procesar el computador, puesto que si se toma al menos una de las imágenes de la base de datos sin modificación para el cálculo de los textones, el equipo puede demorar días sin generar el diccionario de textones, en este caso se seleccionó la primera imagen de cada categoría en la base de datos de entrenamiento, esta se redujo al 30% pasando de un tamaño original de 640x480 a un tamaño reducido de 192x144 pixeles, lo que reduce el volumen de datos y la cantidad de tiempo para generar el diccionario de textones, las imágenes son convertidas a escala de grises y la intensidad de los pixeles es normalizada a 1 dividiendo el vector por 255 que es el valor máximo (blanco) que puede tomar un pixel dentro de la imagen.

Una vez se cuenta con el banco de filtros y con las imágenes modificadas es posible aplicar la función computeTextons, esta a su vez utiliza la función fbRun que tiene como parámetros de entrada el banco de filtros y la imagen que es el resultado de la suma de las imágenes representativas de cada categoría, la función computeTextons también tiene como parámetro de entrada el número de textones que se desea tener en la aplicación (k), esto define el número de clusters a ser utilizados para clasificar los pixeles de una imagen en un número de textones determinado. La función permite aplicar el banco de filtros sobre la imagen y recolectar la respuesta de la imagen a estos filtros, agrupando esta respuesta con k-means según el número de clusters definido, en este caso son 50 textones, ya que una cantidad menor podría no llegar a representar bien la imagen, así como un número superior de textones podría fragmentar en exceso la imagen perdiéndose información sobre algunas texturas, lo que afectaría el desempeño de los clasificadores.

Con respecto a la respuesta de los textones a los filtros, cabe resaltar que se tiene un total de 32 filtros, los textones son una matriz de 50 por 32 que relaciona la respuesta de cada texton a cada filtro, es así que para saber el comportamiento de los filtros para todos los textones se calculó el promedio de la respuesta de cada texton para cada filtro, lo que resulta en el siguiente gráfico.

Ilustración 1. Promedio de respuesta de los textones a los filtros.

De esta gráfica se destaca que para el filtro 10 y 26 se presenta en promedio la mayor respuesta positiva a los filtros para los 50 textones, por otra parte se tiene para el filtro 9 y 27 la respuesta negativa más pronunciada, se pueden diferenciar dos ondas en los promedios ya que se definieron solo dos escalas para la creación de los filtros, es decir, con respecto a la forma y la orientación de los filtros solo se tienen 16 combinaciones, que resultan en un total de 32 filtros por que se repiten según el número de escalas. A continuación se muestra la forma de estos filtros.

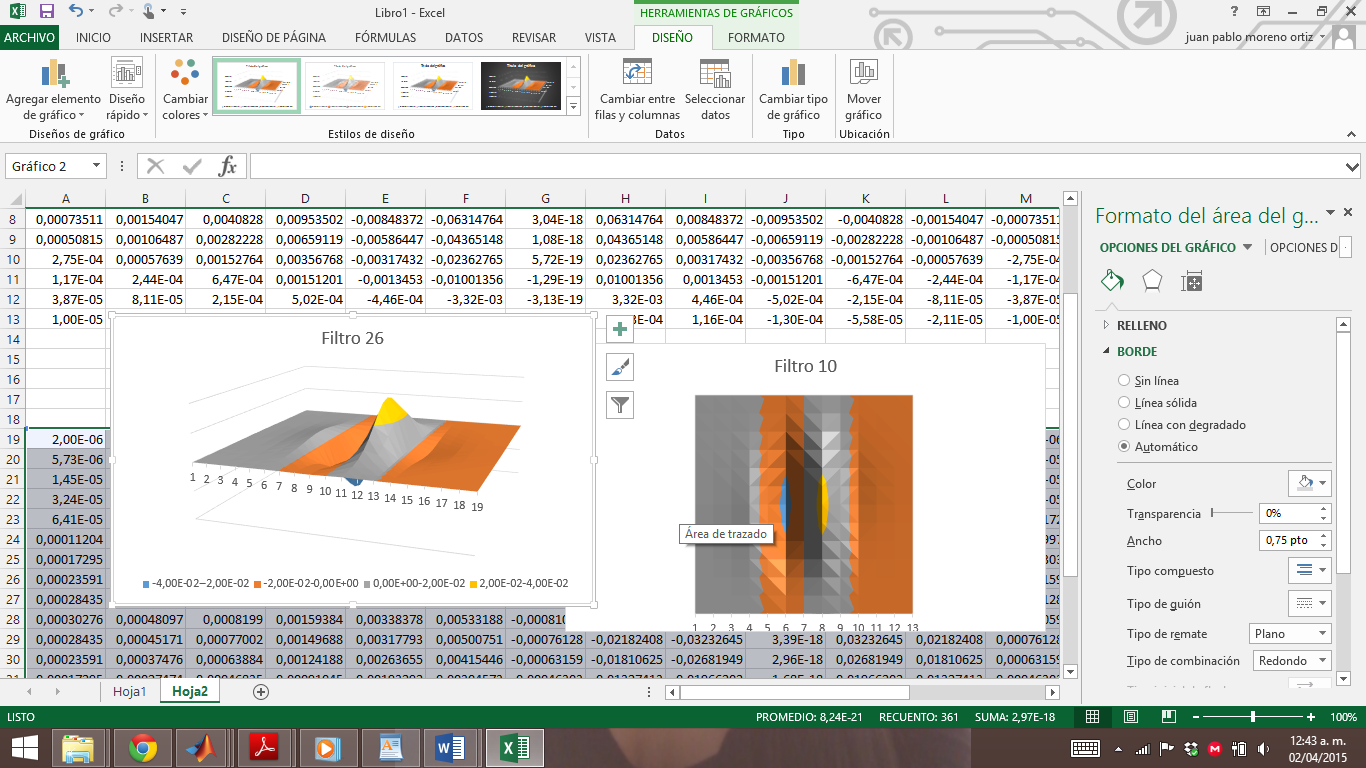
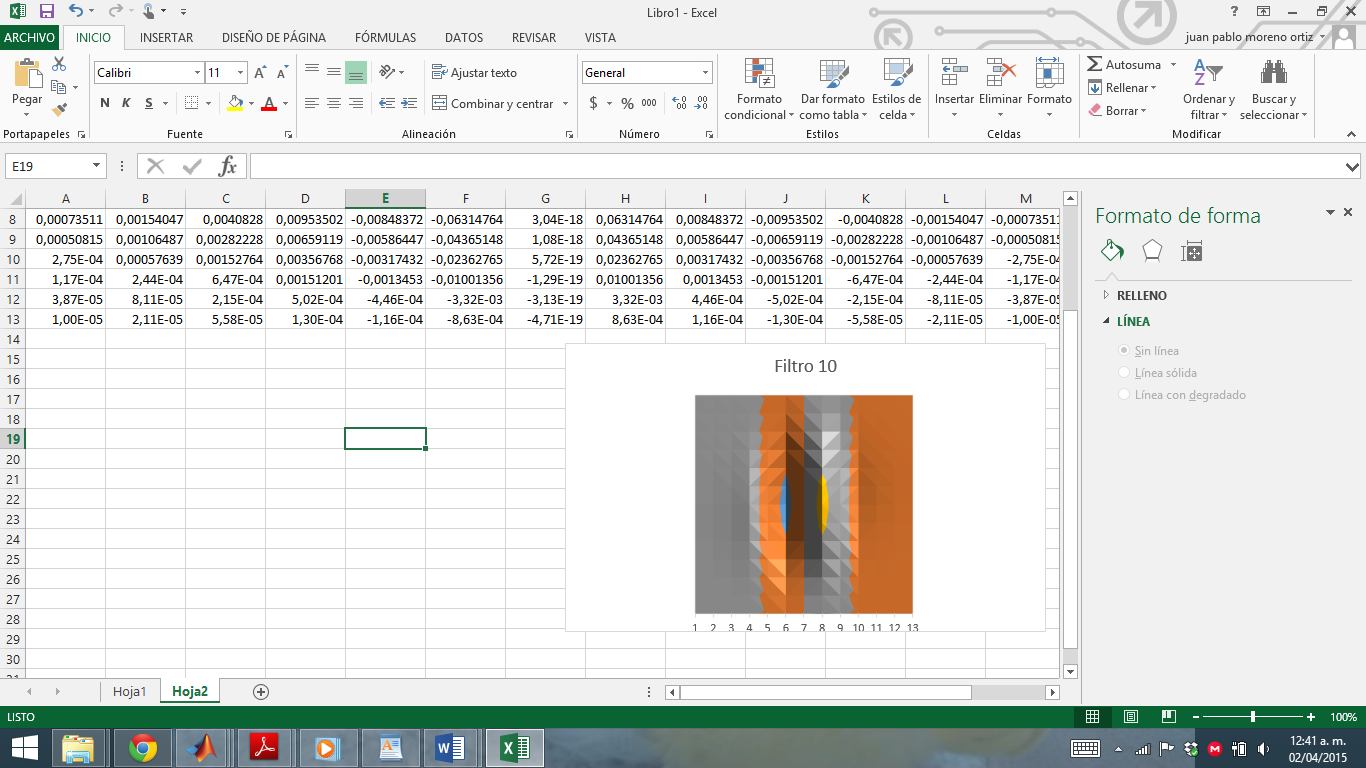


Ilustración 2. Filtros con mayor respuesta positiva.

Tanto el filtro 10 como el 26 tienen la misma forma, la única diferencia entre estos es que el filtro 10 está conformado por una matriz de 13x13, mientras que el filtro 26 está conformado por una matriz de 19x19 datos. Se muestra el filtro 10 desde arriba y el filtro 26 desde otro ángulo para resaltar que tienen dos picos, uno negativo marcado con azul y otro positivo marcado con amarillo. Por otra parte los filtros que dieron las respuestas negativas se muestran a continuación.

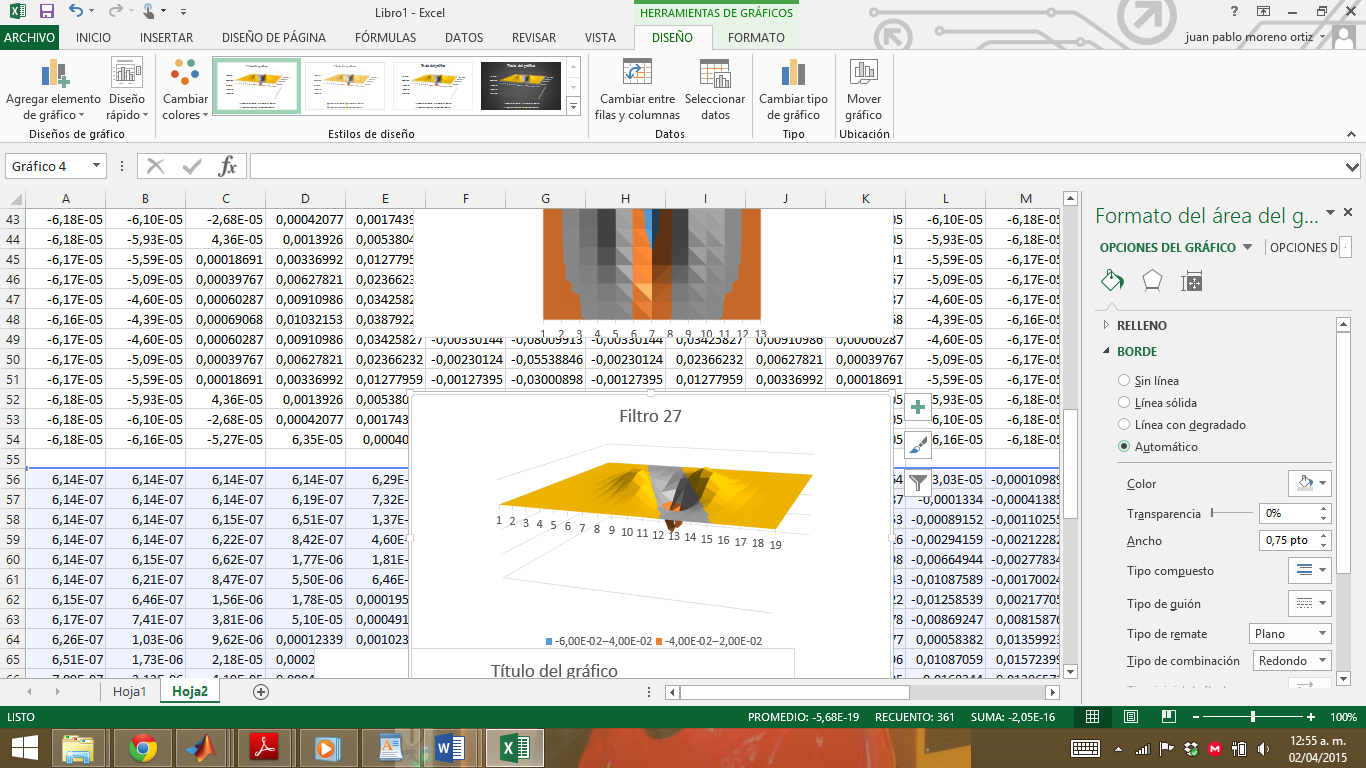
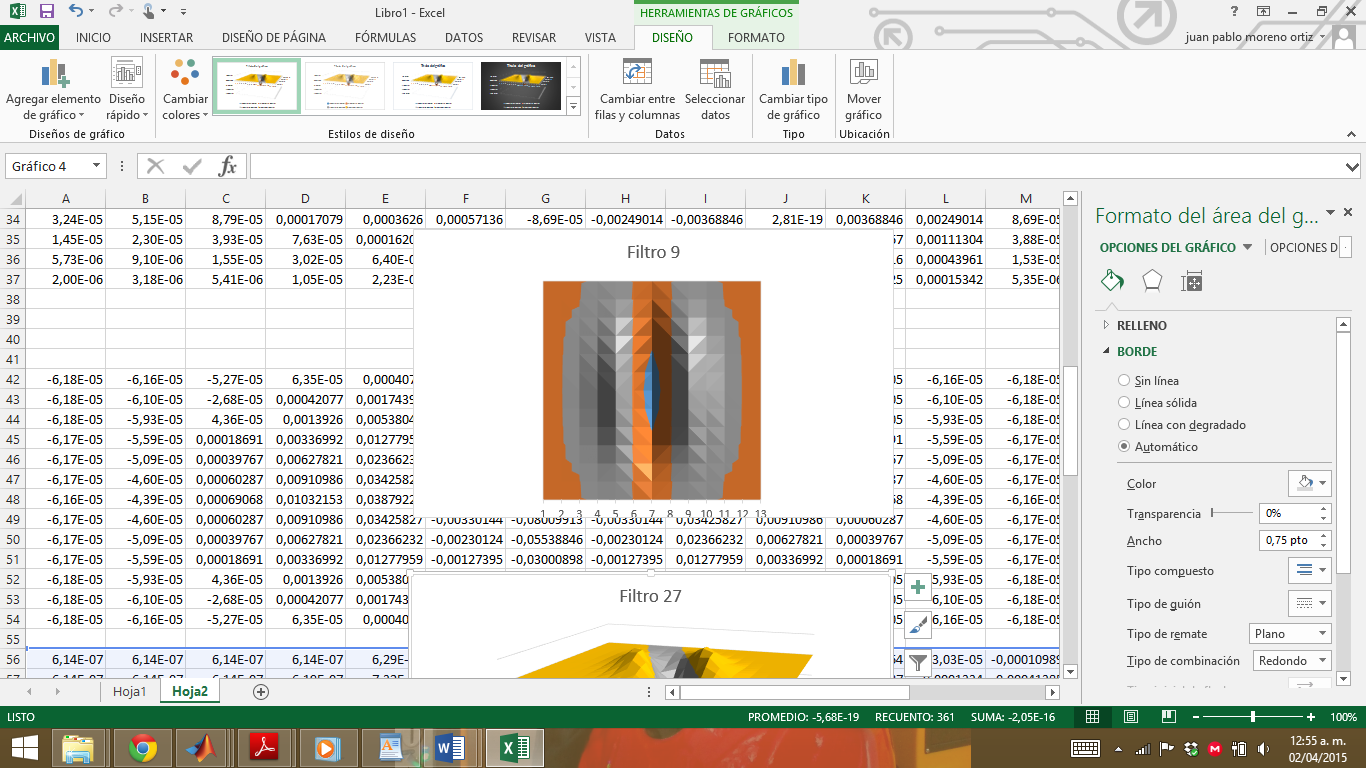


Ilustración 3. Filtros con mayor respuesta negativa.

Los filtros de mayor respuesta negativa no son exactamente iguales entre sí, pues cuentan con una orientación diferente, aunque la forma si es igual, el homólogo del filtro 9 es el filtro 25 que también genera una respuesta negativa, la diferencia más grande con los filtros 10 y 26 es la forma, en estos se tiene un pico negativo en el centro de dos picos positivos no muy pronunciados. De acuerdo con la respuesta de estos filtros con respecto a las imágenes de referencia se tiene que tanto los de alta respuesta positiva como negativa permiten extraer mayor información de la imagen, comparados con los filtros cuya respuesta se acerca a 0, esto será de gran utilidad en el momento de clasificar las imágenes por categorías. Al utilizar los textones para crear el mapa de textones de las imágenes se pueden observar las diferencias en las texturas resaltadas por el cambio de los textones que ocupan ese espacio en la imagen, a continuación se observa el mapa de textones de la imagen de ladrillos, junto con el histograma.

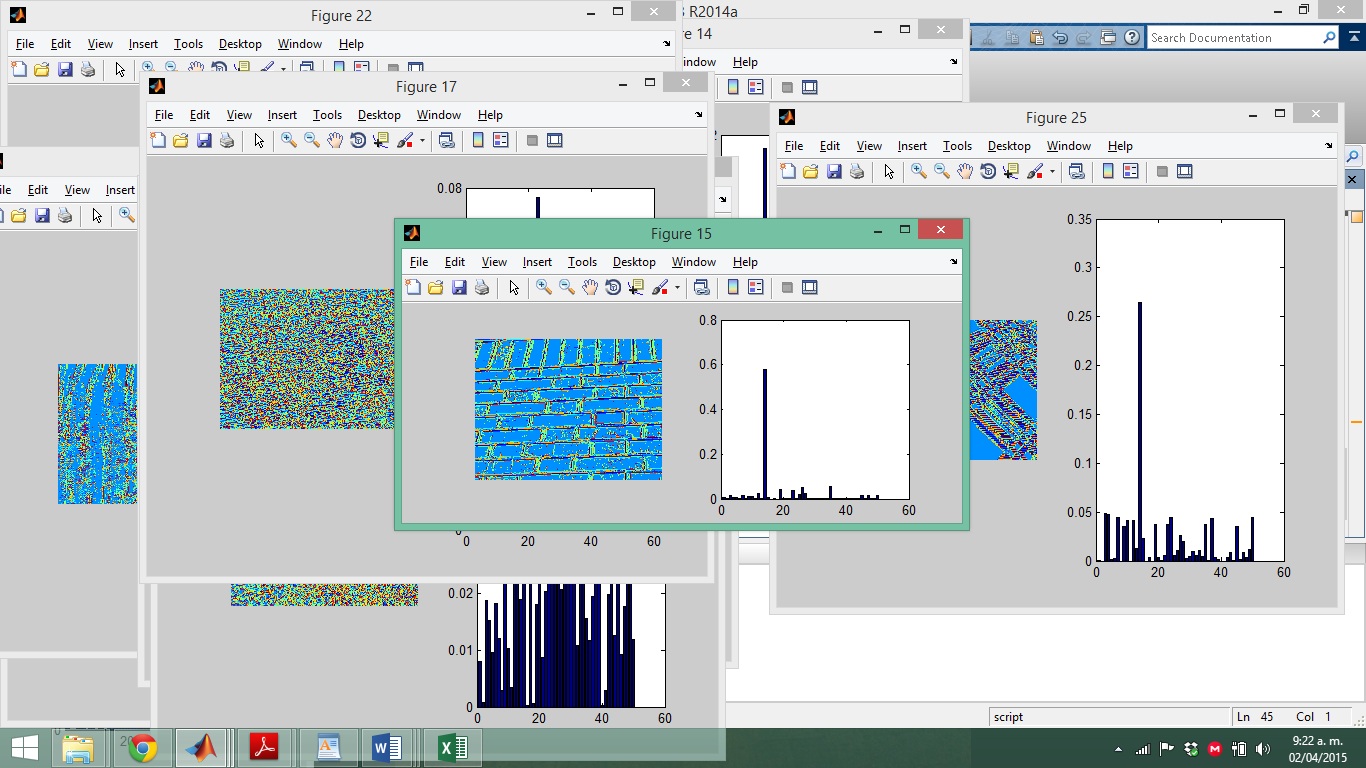
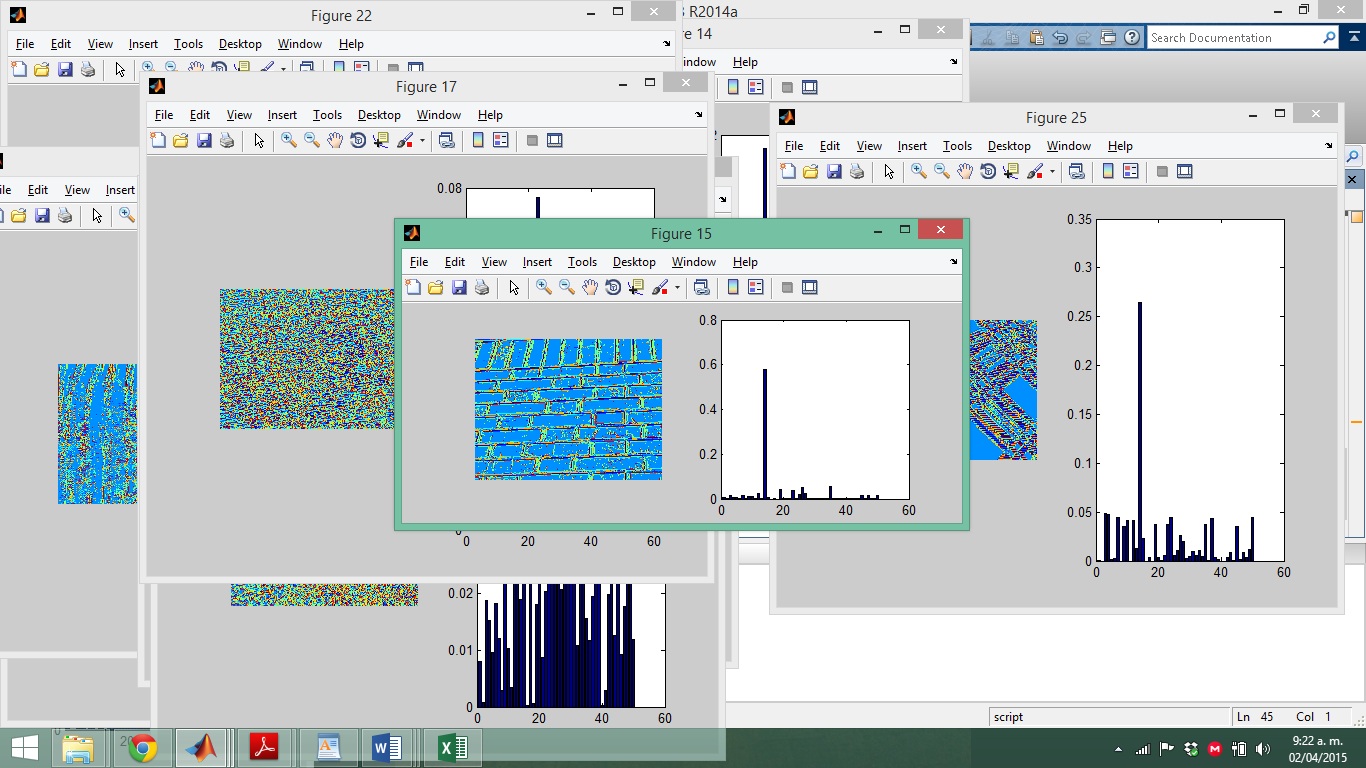
 

Ilustración 4. Imagen, mapa de textones e histograma.

Se observa en la imagen del centro el mapa de textones, donde se diferencian los bordes de los ladrillos, el histograma tiene un pico alto de respuesta al texton 16, esto es característico de esta categoría de imágenes, por lo cual los clasificadores podrán diferenciarlo de las otras categorías con facilidad.

***Clasificadores.***

Para clasificar las 250 imágenes de la base de datos de test en alguna de las 25 categorías se utilizaron dos tipos de clasificadores, el de vecinos cercanos y el de árboles de decisión.

**Nearest Neighbour.** el clasificador de vecino más cercano cuenta en Matlab con funciones específicas que permiten entrenar y aplicar el clasificador a la colección de imágenes, dando como resultado un vector con las categorías identificadas de cada imagen de test, en este caso se utilizó la función knnclassify, esta requiere de los histogramas de los 50 textones para las imágenes de entrenamiento, los histogramas para las imágenes de test y un vector con las etiquetas de cada categoría. Si bien la función simple se escribe: “Clasificador=knnclassify(htestmat,htrainmat,nombrecat);” esta cuenta con otros argumentos que también pueden ser especificados, o que tienen valores por default, en esta función se tiene el argumento k que corresponde al número de vecinos cercanos que serán utilizados en la clasificación, este argumento se mantiene por default en 1, para evaluar el efecto de esta variable se modificó de 1 a 7 y se evaluó la diagonal de la matriz de confusión, de acuerdo a la diferencia entre el máximo de encontrados para cada categoría y el número de encontrados según el valor de k, se tiene que para k igual a 1 y 2 el clasificador responde mejor en general, mientras que para valores altos de k como es el caso de 7 disminuyen los aciertos del clasificador en promedio, esto para distancia euclídea.

También se tiene el argumento de distancia, este especifica la métrica que se tiene en cuenta para comparar los grupos de entrenamiento y los de test, para asociar una imagen de test a una categoría, con respecto a la variación de esta variable se tienen diferencias considerables en cuanto al promedio de clasificación para un k de 1. A continuación se muestra la comparación entre los valores de la diagonal de la matriz de confusión para algunas de las métricas disponibles en la función.

Ilustración 5. Respuesta al cambio de métrica.

En general se tiene que uno de los métodos menos eficientes es el que viene por default programado para la función, medición de distancia euclidea, las métricas que presentaron una mejor respuesta en general para las 25 categorías fueron Cityblock que mide la suma de diferencias absolutas entre el patrón de entrenamiento y la imagen a clasificar, en la misma medida fue superior el método de coseno con respecto a la métrica euclídea, esta representa la resta entre 1 y el ángulo del coseno entre dos puntos tratados como vectores. La correlación es mejor que la distancia euclídea pero no supera las otras dos métricas, esta es el resultado de la resta entre uno y la correlación muestral entre los dos puntos que se están comparando.

Después de este análisis de las opciones en la función knnclassify se tiene que para las pruebas será utilizado un k de 1 y la métrica de cityblock.

**Random Forest:** para implementar el bosque de decisión se utiliza la función TreeBagger, esta función crea un conjunto de árboles de decisión y los ensambla entre sí para formar el bosque. Para este método de clasificación primero se entrena el bosque de decisión y luego se aplica el modelo a las imágenes de test de la base de datos; para la fase de entrenamiento se utiliza la función TreeBagger, que tiene como parámetros de entrada, el número de árboles del bosque, la base de entrenamiento que está representada en este caso por los histogramas de textones de las imágenes de entrenamiento, cada fila de la matriz de entrenamiento corresponde a una categoría o a una imagen, que tiene asociada una etiqueta, esta se extrae del nombre de cada archivo de entrenamiento. Con respecto al número de árboles que se debe utilizar se tiene la posibilidad de incluir dentro de la función el parámetro “oobpred” con valor “on” esto permite almacenar el error de clasificación según el número de árboles utilizados. A partir de esta función se obtiene la siguiente gráfica que demuestra un punto bajo de error al utilizar 200 árboles.

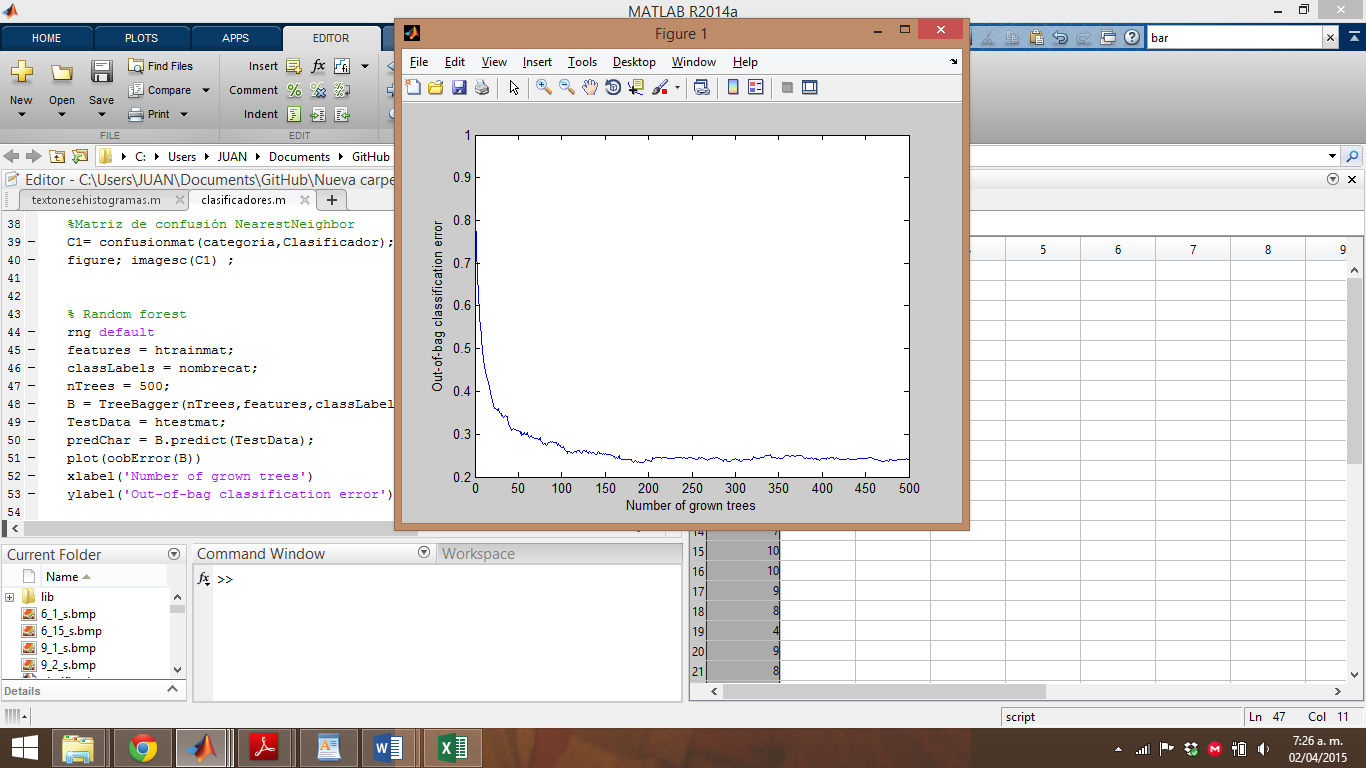


Ilustración 6. Error de clasificación por número de árboles.

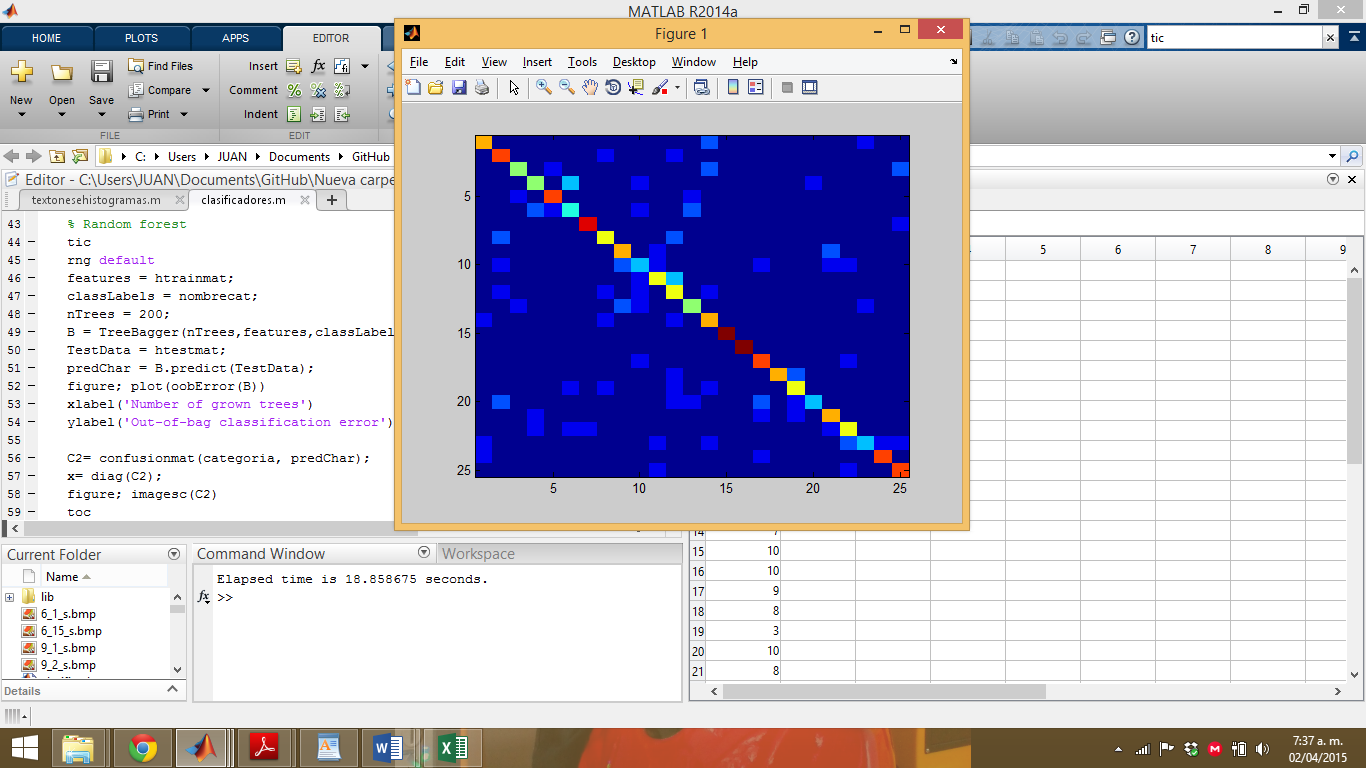
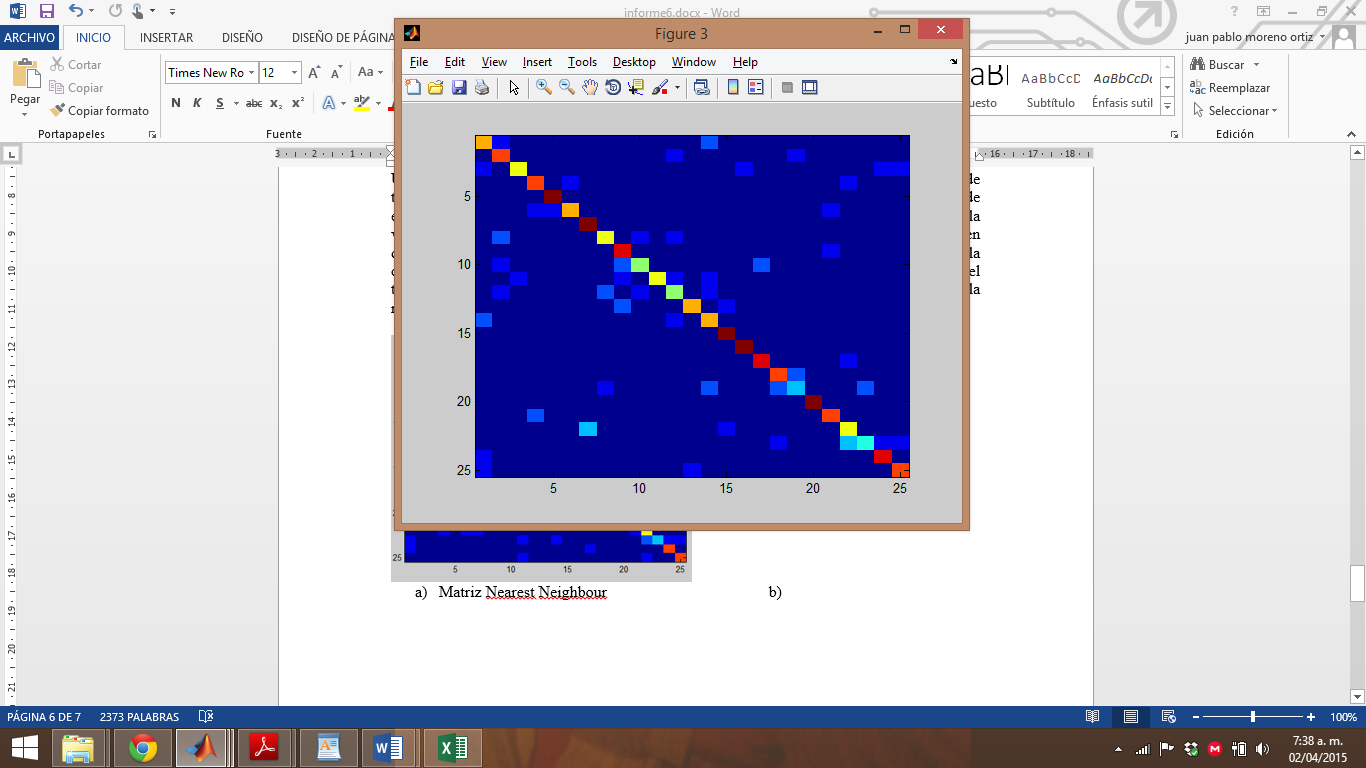
Dentro de los otros parámetros que se pueden configurar dentro de la función se definió el parámetro Method y el valor Classification, puesto que lo que se busca es clasificar de forma eficiente las imágenes incluidas en la base de datos de test.

Para ejecutar el modelo de clasificación entrenado que en este caso se encuentra en la variable B se utiliza la función .predict, sobre la variable que contiene los histogramas de textones, de las 250 imágenes de test, el resultado se utiliza para formar la matriz de confusión que permite evaluar el desempeño del clasificador.

Para la clasificación no fue necesario realizar ajustes sobre las imágenes de entrenamiento y de prueba, los ajustes fueron necesarios para crear el diccionario de textones, puesto que el utilizar las imágenes completas tomaba mucho tiempo de procesamiento, y en el momento de generar los mapas de textones e histogramas de textones, ya que se normalizó el brillo de las imágenes al dividir su contenido por 255, el programa también se asegura de que estas se encuentren en escala de grises aplicando un conversor de RGB a gris, esto porque las texturas no dependen en gran medida del color, sino de los bordes y las líneas dentro de las imágenes.

**RESULTADOS**

Una vez aplicada la configuración a los clasificadores se utilizaron los histogramas de textones de la base de datos de entrenamiento y de test, para verificar el funcionamiento de estos se utilizó la función confusionmat, esta función requiere de un vector que contenga la verdad terreno sobre las imágenes, es decir, el vector con las etiquetas reales de cada imagen dentro de las imágenes de test, por otra parte se requiere del vector resultado de la clasificación asignada para cada imagen por el clasificador, de esta forma se compara el trabajo del clasificador con lo que debería haber alcanzado. A continuación se muestra la matriz de confusión de los dos métodos.

1. Matriz Nearest Neighbour b) Matriz Random Forest

Ilustración 7. Matriz de confusión.

En las dos imágenes de matriz de ambos métodos se puede observar con tonos más oscuros los puntos en los que el clasificador fue mejor, en la matriz a se tienen dos categorías que fueron identificadas en 10 imágenes, mientras que la matriz b cuenta con 5 puntos donde esto ocurrió, para cuantificar exactamente el desempeño real de cada clasificador se calcula el promedio de la diagonal de la matriz y se multiplica por 100 para que sea un porcentaje. De esta forma se tiene que el clasificador de vecinos cercanos alcanza un desempeño del 64.8% mientras que el bosque de decisión un 74.4%, cabe resaltar que esto también implica mayores costos en el tiempo de procesamiento, por lo que para el clasificador de vecinos cercanos se requiere tan solo de 0,36s, mientras que en el caso del bosque de decisión con 200 árboles se toma 19.32s, esto para las 250 imágenes de la base de datos de test (para medir el tiempo de procesamiento se utilizan los comandos “tic” y “toc”), se puede concluir entonces que el bosque de decisión tiene mejores resultados de clasificación pero requiere de más recursos que el método de vecinos cercanos.

Evaluando las categorías y su clasificación con el método de vecinos cercanos se tiene mayor cantidad de clasificaciones erradas, esto se observa en la cantidad de puntos ubicados por fuera de la diagonal de la matriz, 68 puntos cumplen con esta condición, esto sin tener en cuenta el valor de cada uno, contra 47 puntos del bosque de decisión. Esto evidencia una mejor respuesta de clasificación del bosque de decisión. La categoría que presenta mayor error de clasificación en ambos métodos es la 23.

Con respecto a las limitaciones de los métodos se tiene que no existe para ninguno de los dos una configuración de los parámetros de entrada que pueda llegar a clasificar el 100% de las imágenes de test, esto porque solo se está utilizando una forma de representar las imágenes, por lo cual la información que se puede utilizar para la clasificación es limitada. Otro de los limitantes es la cantidad de recursos que se requiere para generar el diccionario de textones con las imágenes de entrenamiento completas, lo cual desmejora el espacio de representación que será utilizado. Se observó para ambos métodos de clasificación que la mejora en la respuesta para ciertas categorías al modificar los parámetros de configuración desmejoraba la respuesta de otras categorías, partiendo de esto se podría conseguir una mejor clasificación si se entrenara un clasificador para cada categoría, si bien esto aumentaría la cantidad de recursos necesarios para su funcionamiento podría llegar a clasificar mejor las imágenes de test.

Una de las principales fallas de las imágenes en la base de datos tanto de entrenamiento como de prueba es que todas se encuentran en escala de grises, esto elimina la posibilidad de poder utilizar el espacio de colores para clasificar los materiales, que sería de gran ayuda para diferenciar una imagen de agua, de una de madera o de cualquier otro material. La ausencia de información adicional sobre la forma en que se tomó la imagen (metadata) que si se tiene en los archivos de imágenes médicas que cumplen con el estándar DICOM, representa una desventaja con respecto al estándar de imágenes disponible actualmente, la información adicional facilitaría el proceso de clasificación de las imágenes.