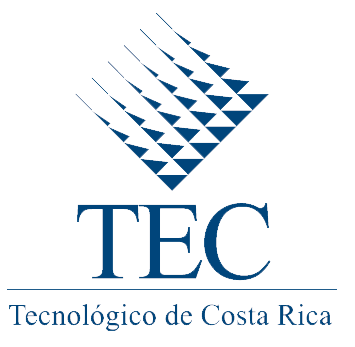
**Tecnológico de Costa Rica**

Sede San Carlos

****

**Escuela de Computación**

**Curso:** Lenguajes de Programación

**Profesor:**

Óscar Víquez Acuña

**Tarea Programada I:**

Procesador de textura de imágenes

**Estudiantes:**

Alonso Vega Brenes (201042592)

Kenneth Sancho Chacón (200941125)

Santa Clara, San Carlos

9 de Octubre, 2011

**Síntesis del problema**

Se debe realizar un programa escrito en el lenguaje de programación C que pueda tomar una imagen de un archivo de mapa de bits, y que calcule su textura a partir de los pixeles y sus valores.

El programa debe leer un archivo BMP, y almacenar en la memoria un valor en escala de gris para cada pixel. Una vez que se tiene en memoria, se calcula su textura utilizando los procesos de cálculo de coocurrencia y homogeneidad.

A partir de la matriz obtenida con los métodos anteriores, se almacenan los valores como pixeles en una nueva imagen en escala de grises. La imagen de salida se guarda en un archivo aparte de mapa de bits.

**Solución y estructuras**

Para modular la descripción de la solución y el uso de estructuras, se dividirá esta sección en dos temas. Uno dedicado al manejo de archivos de imágenes de mapa de bits, y el otro al cálculo y procesamiento de la matriz de valores y su subsecuente resultado en memoria.

**Manejo de archivos BMP**

Para la tarea programada, se han usado archivos de prueba con formato BMP, de 256 colores. La resolución en pixeles no está limitada, sin embargo se recomienda no usar imágenes muy grandes pues el posterior procesamiento es pesado y puede requerir de mucho tiempo. Para dar una referencia, una imagen de 45 x 40 pixeles, requiere 3 segundos para procesarse, mientras que a una de 128 x 128 pixeles toma 23 segundos.

Los archivos de mapas de bits poseen ciertas secciones definidas, entre las cuales están la cabecera de archivo y la cabecera de información. Ambas almacenan información relativa a propiedades de la imagen, tales como su ancho, alto, profundidad en pixeles (en el caso de este proyecto, solo se admiten imágenes con 8 bits de profundidad), posición en el archivo donde inicia el primer pixel de la imagen, etc. Esta información se utiliza para generar luego una matriz en memoria que tenga únicamente valores entre 0 y 255 que representan la escala de gris de cada pixel.

Luego de las cabeceras, se encuentra la paleta de colores, la cual es una tabla con 256 colores representados en estructuras de 4 bytes, uno para cada color base (rojo, verde y azul) y otro para el canal de opacidad (alfa), en este caso siempre al máximo.

Por lo general los pixeles de la imagen en sí, se encuentran justo después de la paleta. Para profundidades de 8 bits, lo que se almacena es la dirección del color (guardado en la paleta), así que cada pixel está almacenado en un solo byte. Sin embargo, a cada fila de pixeles se le agrega un offset de 0 a 3 bytes, el cual ajusta el total de bytes de la fila a un múltiplo de 4. Adicionalmente, se puede mencionar que las filas de pixeles se almacenan de abajo hacia arriba en estos archivos, por lo tanto, cuando se lee secuencialmente como archivo, la primera fila leída es la fila ubicada en el fondo de la imagen.

Finalmente, para realizar la escritura del archivo de salida, se utilizan las mismas cabeceras de información y de archivo que las que contiene el archivo original. La paleta se sustituye por una nueva con valores de escala de grises únicamente. Luego, se utiliza un sistema similar al de lectura, en el cual se escribe primero la fila inferior de la matriz. De nuevo, a cada fila se le agrega un offset para ajustar el total a un múltiplo de 4.

**Procesamiento de imagen**

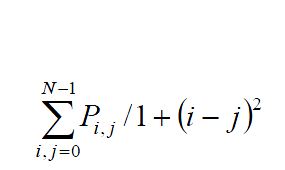
Para el procesamiento de la imagen se usaron algunas variables las cuales mantienen información necesaria para manipular la imagen, como: el tamaño, la matriz que mantiene exactamente las combinaciones que trae la imagen original, además de algunas otras que sirven para poder aplicar la homogeneidad, como la matriz de coocurrencia, además de la matriz ventana que donde se guarda una matriz de 5 x 5 de la matriz imagen u original.

El proceso sigue algunos pasos para poder manipular la imagen, primero se llena la matriz de 5 x 5 que nos ayudará a sacar las combinaciones o los pares ordenados que se estarán escribiendo en la matriz de coocurrencia que es de 256 x 256, el escribir hace referencia a sumar uno en la posición que indique el par ordenado.

Para sacar los pares se debe crear una función que obtiene la relación o par ordenado entre un punto y cada uno de sus vecinos, esto para cada uno de las coordenadas de la matriz de 5 x 5. Luego de que se evalúan todos los pares entonces nos corremos un pixel en la matriz de la imagen y obtenemos de nuevo la matriz de 5 x 5 para realizar otra vez el proceso y así, hasta que recorremos toda la matriz de la imagen de entrada.

Una vez que se tiene la matriz de coocurrencia llena lo que procede es normalizar la matriz esto se hace dividiendo cada punto o par ordenado de la matriz entre el total de combinaciones que se dieron, para una ventana de 5 x 5 siempre el total de las combinaciones es de 144.

La función homogeneidad se basa en una formula (1) la cual se aplica a cada uno de los puntos de la matriz normalizada esto nos va a producir un resultado el cual se tiene que multiplicar por 255 que es la cantidad de bits de la paleta lo que nos va a dar el aproximado de un color, luego se suma 0,5 y se redondea realizado un casting a entero, una vez aquí se envía a escribir el número que está en una variable en la matriz de salida y que además va a producir una nueva imagen con homogeneidad aplicada.

(1)

**Comentarios finales**

El procesamiento de imágenes puede convertirse en una tarea muy pesada para el procesador cuando no se utilizan algoritmos eficientes. En el caso del presente proyecto se pudo observar que imágenes grandes aumentaban el tiempo de ejecución exponencialmente.

El tema que se quería practicar para el proyecto, es el uso de lenguajes imperativos, así como la característica de C de permitir un gran acceso a la memoria mediante punteros. Para realizar los métodos necesarios, los punteros fueron de hecho necesarios, incluyendo su uso desde descriptores de archivos hasta para recibir la dirección de un nuevo arreglo cuando se le solicita espacio en memoria.

Otro aspecto interesante del proyecto, fue el uso de archivos de imágenes. Se necesitó algún tiempo de investigación sobre su uso para poder lograr un correcto funcionamiento. Su apertura, lectura y escritura fue básica y sin usar ninguna librería externa, sino que sólo se manejó como archivo binario secuencial.