Tarea 12

Axel Báez Ochoa

11 de noviembre de 2020

Se implemento el algoritmo de Jarvis para conjuntos conexos en imágenes pgm.

1. Introducción

Una componente conexa es un conjunto de píxeles, C, tal que para cualquier par de píxeles del conjunto, existe un camino digital que los une, donde este camino esta dentro de C. Sea S una imaginen digital, podemos verla en un mallado cuadrado o una matriz, donde cada entrada es un píxel de la imagen y el valor de esta representa el color del píxel. Ya mencionamos en el reporte anterior la relación entre píxeles y a partir de esto como identificamos conjuntos conexos para imágenes que solo tienen colores blanco y negro, lo que se busca ahora es encontrar un conjunto de píxeles que encierre todos estos conjuntos conexos encontrados, a esto lo denotamos como envolvente conexa. El proceso consiste en encontrar un píxel perteneciente a una de las figuras conexas, que se encuentre mas alejado que todos los demás hacia un extremo de la imagen, una vez encontrado calculamos el ángulo entre este píxel y todos los demás píxeles de los conjuntos conexos, aquel ángulo que sea mayor sera la dirección que tomaremos, esto se repite hasta que regresamos al primer píxel encontrado, de esta forma tenemos un conjunto de píxeles, llamémoslos vértices que al ser unidos por una recta en el orden en que se encontraron encerraran a toados nuestros conjuntos conexos.

Ahora este proceso se podría repetir pero en lugar de encerrar a todos los píxeles de los conjuntos conexos encontrados, hacerlo para cada conjunto conexo por separado, pero decidí implementar otra idea, para poder encerrar a cada uno por separado, si lo analizamos al encerrar cada conjunto por separado, podría suceder que dentro de esta envolvente, habría píxeles que no pertenecen al conjunto, es decir píxeles de color negro, esto no se puede evitar cuando se encierran todos los conjuntos pues estos están separados, pero si se hace para un solo conjunto, podríamos pensar en el perímetro del conjunto como la envolvente conexa, si recorremos todos los píxeles de cada conjunto uno a uno, nos fijamos en sus 8 vecinos, explicados en el reporte anterior, si uno de los vecinos es de color negro, entonces, este píxel pertenece al perímetro del conjunto conexo, al tener todos los píxeles que pertenecen al perímetro, estos pasan a ser la envolvente para dicho conjunto conexo.

2. Descripción

El programa **prueba2.cpp** es el que calcula la envolvente para cada conjunto, el programa **prueba3.cpp** se encarga de hacer la envolvente para todos los conjuntos, ambos se ejecutan de la misma forma, con dos parámetros de entrada, la direccion de la imagen original y la de la nueva imagen.

./ejecutable Figures/fig1.pgm Nuevas/Nfig1.pgm

donde **Figures** es la carpeta donde se encuentran todas las 5 imágenes de prueba y **envolvente** es la carpeta para las nuevas imágenes generadas.

Comenzamos leyendo el archivo de la imagen.pgm , creamos una matriz de datos que contenga el valor de cada píxel, y cuando el valor era mas cercano a 0 que a 255 lo guardábamos como 0 en caso contrario se guardaba como 255. Para llevar a cabo el proceso para crear la envolvente de todo el conjunto, creamos una matriz adicional con las mismas dimensiones de la primera, cuyas entradas son puros ceros, recorremos la matriz principal empezando en 0,0, hacia la derecha y luego hacia abajo y cuando un valor sea diferente de 0, significa que este píxel pertenece a uno de los conjuntos conexos, creamos un píxel anterior a la derecha de este que ocupamos para calcular el ángulo que se forma entre estos dos píxeles y todos los demás píxeles diferentes a 0.

Para calcular el ángulo dado tres píxeles, imaginemos que estos son puntos con coordenadas (x, y), así si tenemos los puntos ABC, para las rectas $\bar{AB}$ y $\bar{BC}$, el ángulo $\angle ABC$ es el que nos interesa,si a partir de estos tres puntos formamos dos vectores para calcular el ángulo de nuestro interés, podemos usar el producto cruz y producto punto definido para vectores, para saber que ángulo regresar.

Ya que calculamos el ángulo de forma adecuada nos interesa el ángulo mas grande, pues este asegura que el nuevo píxel encierra a los demás, este proceso se repite hasta regresar al primer píxel encontrado, donde en cada ocasión tomamos el nuevo píxel encontrado como píxel actual, el que era nuestro actual como píxel anterior y se vuelve a calcular el ángulo para todos los píxeles diferentes de 0. Ahora para poder mostrar la envolvente solo tenemos que unir con una recta los píxeles encontrados, esto se va haciendo conforme se van encontrando, es decir una vez que tenemos el nuevo ángulo máximo, marcamos todos los píxeles entre el actual y el píxel cuyo ángulo fue máximo, como píxeles que pertenecen a la envolvente, esto se logra ocupando la ecuación de la recta, evaluando y redondeando los resultados, puesto que nuestras coordenadas son en realidad, filas y columnas de la matriz. Aquí consideramos los casos en los que se esta en la misma fila o en la misma columna , ya que para estos no se ocupa la ecuación de la recta solo se hace un recorrido horizontal o vertical pintando todos los puntos intermedios entre nuestros dos píxeles, cuando si se ocupa la ecuación de la recta debemos considerar la diferencia entre que ejes coordenados es mayor para saber si evaluamos en x, o en y, para que a la hora de ir evaluando se evalúan mas coordenadas, se redondean y tenemos mas píxeles, ya que de otro caso se terminaría con un numero menor de píxeles entre los dos de interés, de forma que no se vería como una recta.

Para el algoritmo extra se ocupa la misma idea de la matriz extra que se ah ocupado en otras, tareas, hacemos el algoritmo para encontrar todos los conjuntos conexos, recorremos la matriz secundaria y cada que un píxel perteneciente a un conjunto conexo tenga un vecino que sea 0, este sera parte de la envolvente de un conjunto conexo.

Para ensanchar la linea de cualquier programa, se vuelve a ocupar la matriz secundaria y los 8 vecinos. Se recorre la matriz secundaria, cada que un elemento es diferente de 0 sus ocho vecinos se añadirán a la envolvente pero con otro índice para no visitar estos, mientras se visitan los píxeles de la envolvente, de esta forma no solo no visitamos los vecinos de estos vecinos originales sino que no visitaremos todos los píxeles originales de la envolvente pues al hacer el recorrido, uno si y uno no tendrán el índice diferente pues son parte de los 8 vecinos del píxel anterior perteneciente a la envolvente.

3. Resultados

Las imágenes para la envolvente para todos los conjuntos conexos se encuentran en la carpeta **envolvente**, los tiempos de ejecución son de aproximadamente 2 segundos para todas las imágenes a excepción de cuando se corre el algoritmo para la imagen numero tres, aquí tarda aproximadamente 12 segundos, en ejecutarse todo el algoritmo , para los conjuntos uno por uno se encuentran en la carpeta **perímetro**, para este los tiempos de ejecución fueron de menos de un segundo para todas las imágenes originales.

4. Discusión

Para todas las imágenes propuestas se logro encontrar la envolvente conexa que encierra a todos los conjuntos conexos de la imagen, el uso de cárdenas (X, Y) si bien parece natural a primera vista para manejar esto, debemos de recordar que para imágenes nuestros ejes no son los de un sistema normal, es decir que en una imagen se empieza desde la esquina superior derecha, la imagen se recorre con dos for, por lo que estás recorriendo por columnas y luego por filas y si tomas que para cada entrada de la matriz es un punto en el espacio, podrías tomar que para matriz[i][j], el $i = x$ y $j = y$, entonces cuando avanzas hacia la derecha estás avanzando en las columnas y cuando avanzas hacia abajo avanzas en las filas, por lo que esto al sistema coordenado que estamos acostumbrado es algo confuso, una vez superas esto, es decir entiendes como funciona el calculo y todo lo demas se vuelva bastante facil.

5. Conclusión

Se eligió usar desarrollar otro algo rimo cuando se desea va la envolvente para cada conjunto conexo por separado, considerando que si bien el algoritmo de la tarea calcularía una envolvente satisfactoria, esta nueva idea podría calcular una mejor, en el sentido de que esta delimitaría únicamente a los píxeles de cada conjunto conexo y no habría píxeles negros dentro de la envolvente. Esto arrojo los resultados esperados, pero al no tener un mayor conocimiento sobre envolventes conexas, desconozco si esto es mejor o peor que usar el algoritmo de Jarvis dado.