Geometría Computacional
Desarrollo Práctica Análisis numérico

Sebastián Arcila Valenzuela (*sarcilav@eafit.edu.co*),
Sergio Botero Uribe (*sbotero2@eafit.edu.co*),
Cristian Camilo Isaza Arias (*cisazaar@eafit.edu.co*),
Eliel David Lobo Vesga (*eloboves@eafit.edu.co*),
Hernán Darío Metaute Sarmiento (*hmetaute@eafit.edu.co*).

2 de octubre de 2009

Resumen

Desde hace ya algún tiempo, se ha notado que la geometría computacional cobra importancia en varias áreas, como la computación gráfica, la robótica y el diseño asistido por computador (CAD). Esto se debe a que la geometría en general, trabaja sobre una de las cualidades más importantes de la materia: su forma. Cuando queremos modelar objetos pertenecientes a la realidad, debemos dar cuenta de la forma que tienen y de ciertas propiedades que delimitan cómo esta forma interactua con el entorno. Desde aquí que sea esta materia la que nos permita trabajar propiedades tan importantes como el área en 2 dimensiones, el volumen en 3 y las relaciones de los objetos en el espacio (intersecciones, distancias, etc.). Sabido esto, es de extrañarse que en muy pocas universidades se imparta un curso que aborde los métodos computacionales que se pueden aplicar cuando se trata de resolver problemas relacionados con la geometría computacional. Es por esto que en el presente proyecto se hace un esfuerzo por acomodar de alguna manera los métodos numéricos computacionales al área de nuestro interés para así construir un conocimiento aplicado por medio de la evaluación de estos métodos, la aplicación al problema planteado y las comparaciones en rendimiento algorítmico de las soluciones propuestas al problema planteado.

1. Introducción

Los objetos geometricos, tales como los puntos, las lineas y los polígonos son la base de una gran cantidad de aplicaciones importantes y dan nacimiento a un interesante conjunto de problemas y algoritmos. El nombre geometría nos lleva a su primer uso: la medición de tierra y materiales. Hoy en dia, las computadoras se usan cada vez más para resolver problemas geométricos de mayor escala. En las últimas dos décadas se ha desarrollado un conjunto de herramientas y técnicas, sacando provecho de la estructura que provee la geometría. Esta disciplina se conoce como Geometría Computacional.

Shamos le dio nombre a la disciplina y la impulsó en gran manera alrededor del año 1975; su tesis de Ph. D. atrajo una considerable atención hacia el área. Después de una década de desarrollos, el campo se independizó en 1985, cuando se cumplieron los tres requisitos de cualquier disciplina saludable de: un libro de texto, una conferencia y una revista. El libro de Preparata y Shamos ·Çomputational Geometry: an introduction"que fue el primer libro de texto dedicado enteramente al tema se publicó casi al mismo tiempo que se llevó a cabo el primer Simposio de la ACM en Geometría Computacional; justo antes del comienzo de una revista de Springer-Verlag: "Discrete and Computational Geometry". El campo actualmente evoluciona; desde 1985 han aparecido varios textos, colecciones y monografías. El simposio anual atrajo de una manera continua 100 papers y 200 asistentes. Hay, por lo tanto, evidencia de que el campo de investigación se está expandiendo hasta tocar modelación geométrica y demostraciones de teoremas geométricos. Tal vez el elemento más importante es que los primeros estudiantes que obtuvieron sus Ph. D. en ciencias de la computación al presentar tesis en geometría computacional ya se graduaron, están bien posicionados y entrenando a la siguiente generación de investigadores.

La geometría computacional es importante puesto que el espacio euclideano de dos y tres dimensiones representan el campo en el que se organizan los objetos físicos reales. Un gran número de áreas de aplicación tales como reconocimiento de patrones, computación gráfica, procesamiento de imágenes, investigación de operaciones, estadística, CAD o diseño asistido por computadora, robótica entre otras han sido el nicho de incubación de la disciplina, puesto que aportan problemas inherentemente geométricos para los que hay que desarrollar algoritmos eficientes. Un gran número de problemas de manofactura, involucran actividades como el trazado de cableado, la ubicación de instalaciones, el corte de materiales entre otros problemas geométricos de optimización. Para poder resolver estos problemas de manera eficiente permitiendo que una computadora haga con rapidez los cálculos, se requiere el desarrollo de nuevas herramientas geométricas, del mismo modo que requiere el diseño de algoritmos veloces. Aquí se notará que no se trata de traducir sencillamente teoremas bien conocidos en algoritmos o programas de computadora. Desde el punto de vista teórico, la complejidad de los algoritmos que resuelven problemas geométricos es de gran importancia puesto que arroja nueva luz sobre las dificultades intrínsecas de la computación como tal.

2. Geometria Computacional

2.1. Descripcion General

La geometría computacional es una rama de la computación que es bastante amplia en si misma. Su objetivo principal es modelar los problemas geométricos mediante el uso de algoritmos. Esta representación algorítmica permite que los problemas se puedan trabajar mediante el uso de la computadora y en algunos casos los reduce a optimizaciones puramente matemáticas, como es el caso de interpolaciones y triangulaciones.

Con problemas geométricos aquí nos referimos a todos aquellas aplicaciones de la geometría en las cuales el número de calculos a realizar o la operaciones en si mismas son complejas, no solo por la cantidad sino por la complejidad que poseen, un claro ejemplo sería el calculo del volumen de un sólido irregular o sus similares.

Con la geometría computacional se pretende pues, dar solución a estos problemas, que en muchos casos son mas fáciles de resolver intuitivamente que hacer un algoritmo que calcule la solución. Aunque cuando el tamaño del problema aumenta, su complejidad tambien lo hace y la solución intuitiva se hace mas complicada, lo que hace mas versátil la solución algorítmica. Un ejemplo de ello es el cálculo del polígono convexo que envuelva una nube de puntos, conocido con el nombre de convex hull. Intuitivamente es sencillo de resolver pero realizar un algorimo para ello (aunque ya existe muchos) es más complejo. Esto se verá reflejado en el desarrollo de este proyecto.

El punto de partida y a su vez uno de los conceptos más utilizados en el área de la geometria computacional es el de nube de puntos. esto no es mas que un conjunto de puntos que pueden estar en el plano cartesiano (2D) o bien en tres dimensiones. Una nube de puntos es la forma mas sencilla de representar un objeto irregular en la computadora (un objeto regular como un cilindro o una esfera se puede representar con una ecuacion). Por ejemplo un contorno se puede mapear a una nube de puntos que siga aproximadamente sus limites. Añadiendo un tanto de complejidad y conceptos de algoritmia, mediante una nube de puntos y las relaciones entre los mismos en forma de grafo se puede crear lo que se conoce con el nombre de sitema de partículas. La relación de las particulas entre si puede ser un campo magnético que las mantenga estables, simples uniones fijas para formar una maya o las articulaciones de algun cuerpo. Así pues, una vez se tenga modelado un objeto real en esta forma se puede operar sobre el y hacer calculos y simulaciones sobre el comportamiento de la materia y sus propiedades, que es otro de los objetivos de esta área de la computación.

//nube de puntos y sistema de particulas (imagen)

Normálmente las nubes de puntos, por ser aproximaciones iniciales, son poco precisas con respecto al objetivo que se tiene para un problema geométrico

en particular. Por ejemplo, para calcular un área, una nube de puntos puede usarse para dar una buena aproximación inicial en términos del tiempo usado para dicho cálculo y la complejidad de las operaciones realizadas, pero, si lo que se requiere es precisión, el camino a seguir es usar esa nube de puntos y realizar operaciones más complejas sobre ella que eventualmente podrían llegar a trasformarla (entiendase por transformarla, modificaciones que sufre la nube de puntos o el ingreso de nuevos elementos a esta), de este modo se obtiene una solucion mucho mas excata del problema pero mas costosa y menos eficiente.

La búsqueda de buenas soluciones, eficientes, exactas, escalables entre otras, es uno de los motores que mueven la geometría computacional y que permite que día a día la investigación en esta área se mueva para dar nuevos resultados y mejores respuestas a las necesidades que se plantean. Estas necesidades son diversas dependiendo del campo del saber en el que se este inmerso, pero lo que es claro es que existe muchas aplicaciones de la geometria computacional. aqui se mostraran algunas.

2.2. Fundamentos

2.2.1. Grafos Geométricos en el Plano

Un grafo $G = (V, E)$, siendo V el conjunto de vértices o nodos y E, el conjunto de aristas, se denomina planar si se puede incrustar en el plano sin que se insersecten aristas.

Se le llama incrustar en el plano al proceso por el cual se mapea cada nodo en el conjunto V de un grafo $G = (V, E)$ a un punto en el plano, y cada arista en E a una curva simple entre las dos imágenes de los nodos extremos de la arista, de tal forma que no se intersecten dos imágenes de aristas más que en sus puntos terminales. Se conoce a la imagen del mapeo como un grafo geométrico en el plano. Si todas las aristas de un grafo geométrico G se representan como segmentos de línea recta en el plano, se dice que G es un grafo planar de línea (PSLG por sus siglas en inglés: Planar straight-line graph). Un PSLG G determina en general una subdivisión del plano. Cada región R de la subdivisión, junto con las aristas de G que están en la frontera de R forman un polígono en el plano.

Fórmula de Euler

Sean v, e y f el número de vértices, aristas y regiones (incluyendo la región que no está limitada por las aristas) respectivamente pertenecientes a un PSLG. La famosa fórmula de Euler relaciona estos parámetros de la siguiente forma:

$$v - e + f = 2$$

Si además podemos asegurar la propiedad de qe cada nodo tiene un grado del al menos 3, podemos entonces probar las siguientes relaciones:

- $v <= (2/3) * e$

- $e <= 3f - 6$
- $f <= (2/3) * e$
- $v <= 2f - 4$
- $e <= 3v - 6$
- $f <= 2v - 4$

Por lo tanto, decimos que para un grafo planar, el número de nodos, el número de aristas y el número de regiones están relacionadas de manera lineal.

2.3. Fundamentos Geométricos

De acuerdo con la naturaleza de los objetos geométricos involucrados, podemos identificar básicamente cinco categorías en las que se pueden clasificar de una manera conveniente la colección completa de problemas geométricos; a saber: Convexidad, proximidad, búsquedas geométricas, intersección y optimización.

2.3.1. Convex Hulls

Sea L un subconjunto propio del espacio euclidiano, L se denomina conjunto convexo si para cada par de puntos $p1, p2$ pertenecientes a L, el segmento de línea que une a $(p1, p2)$ está completamenet en L.

El convex hull $CH(L)$ es el conjunto convexo mínimo que contiene a L.

Dados n puntos en el plano, queremos encontrar su convex hull. Este problema es tan fundamental para la geometría computacional como el ordenamiento lo es para los algoritmos generales. También es un vehículo para la solución de muchos problemas que aparentementemente no están relacionados; todos estos pertenecientes al área de la geometría computacional. La construcción de el convex hull de un conjunto finito de puntos también ha encontrado aplicaciones en muchas áreas, tales como el reconocimiento de patrones, el procesamiento de imágenes, en Robótica y en el corte y ubicación de materiales.

Un polígono en el espacio euclideano es un conjunto finito de segmentos de línea que satisfacen las siguientes dos condiciones:

1. Cada punto extremo lo comparten exactamente 2 segmentos de línea y
2. No existe un subconjunto propio de estos segmentos de línea que cumpla la propiedad 1.

De estas dos definiciones encontramos que un convex hull del conjunto L perteneciente al espacio euclideano es un polígono que cumple a la vez la propiedad de ser convexo.

2.3.2. Problemas de Proximidad

Entre los problemas clasificados como problemas de proximidad encontramos: EL par más cercano, todos los vecinos más cercanos, el árbol euclideano de expansión mínima, triangulación y el máximo círculo vacío.

Los problemas de proximidad resultan en muchas aplicaciones donde se modelan objetos físicos o matemáticos como puntos en el espacio. Entre estos encontramos:

Agrupamiento: un cierto número de entidades se agrupan si están lo suficiente cerca entre ellas.

Clasificación: un patrón nuevo que se va a clasificar se le asigna a la clase de su vecino (clasificado) más cercano

Control de tráfico aéreo: los dos aviones que están más cerca son los que están en mayor peligro.

Hablaremos entonces que para cualquiera de estos problemas (en 2D) la entrada consistirá en un conjunto S de n puntos en el plano. La distancia entre puntos en S será la distancia euclideana entre los puntos.

Par más cercano: Encontrar un par de puntos en el conjunto S que esten separados por la menor distancia entre todos los pares de puntos.

Todos los vecinos más cercanos: Para cada punto en el conjunto S, encontrar un punto que sea el más cercano a él.

Árbol euclidiano de expansión mínima : Econtrar un árbol que interconecte todos los puntos con la distancia total mínima, cuyos nodos serán los puntos en el conjunto S.

Triangulación: Unir los puntos en el conjunto S con segmentos de línea recta que no se intersecten de tal forma que cada región interior al convex hull S sea un triángulo.

Máximo círculo vacío: Encontrar un círculo vacío tan grande como sea posible que no contenga ningún punto del conjunto S, aunque su centro sea el interior del convex hull de S.

Todos estos problemas se relacionan en el sentido en el que todos tienen que ver con las distancias respectivas entre puntos en el plano.

2.3.3. Intersecciones

Los problemas de intersección y sus variaciones se presentan en muchas disciplinas tales como diseño arquitectónico, computación gráfica, reconocimiento de patrones, etc. Un diseño arquitectónico no puede dejar que dos objetos impenetrables compartan una regiín. Cuando se muestran objetos en una pantalla de 2 dimensiones, las porciones que se ocultan (porciones que se intersectan)

deberían eliminarse para aumentar el realismo; este es un problema antiguo conocido como eliminación de superficies escondidas. En el diseño de circuitos electrónicos, dos componentes diferentes deben estar separados por una cierta distancia, y la detección de si esta separación se obedece o no, puede verse como una mezcla entre el problema de distancia y el de intersección. Debido a que esta tarea puede involucrar miles de objetos, se necesitan algoritmos veloces para reportar intersecciones o elementos traslapados. Otra motivación para estudiar la complejidad de los algoritmos de intersección es que se puede arrojar algo de luz sobre la complejida inherente de los problemas geométricos fundamentales. Por ejemplo, cuán difícil es decidir si un polígono dado que tiene n vértices es un polígono simple o cuánto tiempo se necesita para determinar si dos objetos de un conjunto de n elementos, sean polígonos, segmentos de línea, etc. se intersectan.

Intersección de segmentos: Dados n segmentos de línea recta en el plano, hallar todas las intersecciones

Intersección de semiplanos: Dados n semiplanos en el plano, computar su intersección común.

Intersección de poligonos: Dados dos polígonos P y Q con m y n vértices respectivamente, computar su intersección.

2.3.4. Busquedas Geométricas

Este problema geométrico tiene una buena motivación, que es el problema de la oficina postal, propuesto por Knuth. Dado un mapa fijo de n oficinas postales, para un punto de búsqueda arbitrario, ¿cuál será la oficina postal más cercana? La solución a este problema es sencilla: comparar la distancia entre el punto de búsqueda y todos los puntos que representan las oficinas postales para encontrar así la más cercana. La complejidad de este algoritmo es obviamente $O(n)$. También es fácil ver que para encontrar la oficina postal más cercana, se necesitan como mínimo n comparaciones, ya que si el algoritmo deja de computar la distancia entre el punto de búsqueda y alguna oficina postal, entonces siempre podremos construir una entrada para la cual el punto de búsqueda sea el más cercano a la oficina postal que no comparamos, lo que haría que el algoritmo diera una respuesta incorrecta. Por lo tanto, para un solo punto de bñusqueda, el algoritmo simple que ya se mencionó sencillamente no es óptimo.

Para este tipo de problemas son necesarias estructuras de datos más avanzadas, tales como los diagramas de Voronoi, que nos ayudarían a computar la oficina postal más cercana para un conjunto de puntos de entrada sin que sea necesario usar un tiempo $O(n^2)$ para solucionar este problema.

2.4. Usos

Como ya se dijo, existe una amplia gama de usos de la geometría computacional. Sin el ánimo de ser extensivo en cada uno de ellos y solo para ejemplificar

un poco el alcance de nuestra area de trabajo, acontinuacion se enuncian algunos de estos.

CAD (Computer Asisted Desing)

Los CAD o programas de diseño asistido por computadora, son como su nombre lo indica una herramienta usada por diseñadores, arquitectos y similares para hacer sus tareas diarias como lo son planos, modelos o prototipos de algun producto, edificación, maquina entre otras. los CAD tienen la ventaja sobre las tecnicas convencionales usadas en el diseño en que este permite hacer modelos virtuales de diferentes objetos o estructuras que de otra manera serian especificaciones en un dibujo muy detallado en un papel en 2D.

La interaccion con un CAD se da mediante una interfaz gáfica donde el usuario puede crear formas y solidos, e integrar varios de estos elementos en una sola pieza. además permite darle a cada elemento diferentes propiedades como color, textura e incluso material. Asi pues, se crean prototipos virtuales bastante cercanos a la realidad y partiendo de estos se pueden posteriormente generar las piezas en maquinas de corte numérico.

Videojuegos y Simulaciones

Un uso menos académico pero no menos exitoso de la geometría computacional ha sido su aplicacion a los videojuegos. esta va de la mano con la computación gráfica y su funcion principal aqui es la simulacion de propiedades fisicas de los objetos con el fin de que la experiencia en un videojuego sea lo mas realista posible.

De este modo se usa geometría computacional para calcular la deformación de los objetos ante un golpe, el movimiento de una persona al caminar, la destruccuion de objetos ante una explosion entre otras. en muchos casos estas simulaciones no son fisicamente confiables pero los resultados obtenidos a la vista son bastante realistas, esto en realidad no es de importancia porque dichos resultados no son utilizados para nada mas que algo grafico.

Una aplicacion bastante similar pero en donde si importa la exactitud de los resultados (fisicamente hablando) es en las simulaciones. Para estas se usan aplicaciones conocidas como motores graficos, en donde se pueden crear objetos y someterlos a fuerzas de diferentes tipos calculando así su deformación o sus puntos de quiebre (si se trata de una estructura por ejemplo), los motores fisicos tienen la caracteristica de que disminuyen de una manera dramática los costos en las simulaciones que llevan a acabo. Por ejemplo es muchísimo mas fácil y menos costoso simular el choque de un auto con integrantes en su interior y despues evaluar los daños, que hacerlo en la vida real. Incluso la medición de los resultados es instantanea por ser un modelo virtual.

Sistemas de posicionamiento global GPS

Este sistema, operado mediante el uso de satelites, requiere tambien de aplicaciones de geometría computacional, mas especificamente triangulaciones que

son bastante usadas en lo que a localizacion respecta. En cartografía antigua se usaba triangulación para determinar posiciones en mapas. en los sistemas modernos de GPS tambien se usa este concepto pero de una manera un tanto diferente. El sistema GPS se soporta sobre un conjunto de satelites ubicados alrededor de la tierra, un aparato receptor que quiere conocer su ubicación es localizado por al menos tres satelites que envian una señal, a partir de estas señales se hace una triangulacion conociendo la velocidad de propagacion de la seña y el tiempo de propagación de la mismas, usando la distancia de los satelites como una herramienta para este cálculo. De esta manera se obtienen una podición bastante excacta mediante el uso de tecnicas de geometria computacional.

3. Problema: Descripción y Justificación

Cuando se trabaja con representaciones de la realidad en el computador, a menudo se tiene que hacer una simplificación de esta. La modelación permite obtener información relevante del objeto que se estudia, dejando de lado las cualidades secundarias que no atañen a cierto problema que se desea atacar y obteniendo tanta información como sea posible de las características que se usan en el modelo como delimitantes.

En los diferentes campos de acción que tienen las ciencias relacionadas con (o que se ayudan de) la computación, a menudo se hace necesario representar objetos de la realidad para permitir que el computador agilice cálculos que, de otra forma, sería tedioso trabajar. Una primera aproximación a cualquier forma geométrica en 2D puede ser el contorno o silueta de la figura modelada. Desde esta silueta se puede comenzar un proceso que encaminado a hacer más compleja la figura para que dé cuenta de más propiedades, y que estas, a la vez, sean más fieles al comportamiento de las caracerísticas que tienen los objetos en el mundo real.

El problema viene entonces cuando se trata de pasar de la simplificación que se hace de una figura a la forma que originalmente tenía. En este traspaso se trabaja con unas unidades mínimas (puntos) que representan, en algunos casos, la medición de un aparato externo al computador, y que es necesario *traducir* de nuevo a una imagen única y completa para reconstruir el objeto (en este caso su contorno) y poder trabajar con él. En general, la primera técnica algorítmica que se aplica cuando se trabaja con nubes de puntos en 2D, se conoce como Convex Hull [1]. El convex hull es a la geometría computacional, lo que el ordenamiento es para muchos problemas algorítmicos; esto es, nos permite hacer un trabajo sobre una entrada de puntos sin ninguna estructura para poder hacer luego operaciones más interesantes sobre los datos.

El convex hull de un conjunto de puntos, se define como el polígono convexo (ángulos interiores menores de 180^o) mínimo que contiene todos los puntos de dicho conjunto. De aquí que, al tener esta primera aproximación se pueda trabajar con una delimitación del área de trabajo que tiene propiedades importantes para el cálculo de áreas, por ejemplo. Sin embargo, el convex hull termina siendo

sólo una primera aproximación a los datos, puesto que, por la definición misma del método, deja ciertos puntos sin incluir y nos da un *contorno brusco* de los puntos que se trabajan. Por otro lado, hay casi tantas técnicas para el convex hull como de algoritmos de ordenamiento; todos ellos basándose en algún criterio para clasificar los puntos. Lamentablemente, ninguno de ellos utiliza métodos numéricos. Para lograr superficies suaves y curvas más complejas, son necesarias técnicas más complejas que esta primera aproximación. Cuando necesitamos llegar a un contorno suave, estamos en el dominio de las curvas conocidas como splines, las que tienen la propiedad de poder representar contornos suaves con cierta facilidad para el programador, lo que hace que varios paquetes gráficos traigan ya implementadas una forma particular de estos splines, llamados Curvas de Bèzier.

3.1. Metodos Asociados

Como ya se planteó en la descripción del problema, los métodos que se van a utilizar para resolver el problema se toman considerando la complejidad y la precisión que se desea para la aproximación a la silueta. Esto, debido a que en los problemas de interpolación normalmente resultan funciones bastante complejas (polinomios de grado alto), las cuales dan un alto grado de precisión puesto que garantizan que la función hará pasar la gráfica por todos los puntos. En nuestro caso, para buscar una silueta. Sin embargo, estos polinomios resultan tan costosos de calcular que no se justifica su utilización. Algunos de los métodos que se pueden utilizar para abordar el problema son:

3.1.1. Polinomios de Lagrange [2]

los polinomios Lagrange definen una función $P(x)$ que aproximan una función $f(x)$ mediente un polinomio de grado menor al de $f(x)$, asi pues, sean

$$f(x_0) = y_0,\ f(x_1) = y_1$$

definimos las funciones

$$L_0(x) = \frac{x - x_1}{x_0 - x_1},\ L_1(x) = \frac{x - x_0}{x_0 - x_1}$$

y obtenemos nuestra función

$$P(x) = L_0(x)f(x_0) + L_1(x)f(x_1)$$

obsérvese que $L_0(x)$ es cero para todo valor diferente de x_0, de igual forma $L_1(x)$ es cero para todo valor diferente de x_1 de esta manera $P(x)$ evaluada en los puntos x_0 y x_1 será igual (o aproximada con error en algunos casos) a $f(x)$. Para este caso se obtiene una interpolación mediante una línea recta entre los dos puntos, lo que virtualmente es una función mucho menos compleja que el $f(x)$ original.

Para grupos de puntos mucho más grandes, existe otra forma de calcular la función $P(x)$ usando una sumatoria y una productoria, pero esta explicación está fuera del alcance de este documento.

Se tiene pues que para un grupo de $m + 1$ puntos , la ecuación obtenida sera como máximo de grado m, esto nos dice que mientras haya pocos puntos a interpolar, los polinomios de lagrange funcionarán (seran de utilidad dentro de nuestra investigación), pero si hay demasiados puntos, la función se vuelve dificil de calcular y computar, haciendo que la reducción de la complejidad de la que se habló en un principio no sea tan visible.

Ventajas:

- La gráfica del polinomio generado por Lagrange pasa por todos los puntos del arreglo.
- La función generada es suave en su recorrido por los puntos dados.
- La función construida tiene derivadas continuas de los ordenes necesitados (para conectar secciones diferentes).
- El polinomio de Lagrange se determina unívocamente para cada arreglo de puntos.

Desventajas:

- El grado del polinomio generado depende del número de puntos ingresados y, a medida que el conjunto de puntos aumenta, de igual forma lo hace el grado del polinomio; así también el número de cálculos necesarios.
- El cambio de un punto en el arreglo fuente causa que se recalculen todos los coeficientes en el polinomio de Lagrange
- Al agregar un punto, se cambia el grado del polinomio de Lagrange y se debe entonces recalcular todos los coeficientes del polinomio.
- Cuando se refinan los puntos iniciales indefinidamente (se agregan nuevos puntos al contorno) el grado del polinomio de Lagrange crece, de igual forma, indefinidamente.

Eficiencia de Polonomios de Lagrange

Como ya se mencionó, el método de Lagrange puede funcionar bien para valores n pequeños. Acontinuación se pretende demostrar esto mediante las ecuaciones que se usan para calcular $P(x)$ y que al igual que en la sección anterior no serán demostradas (Para una demostración detallada, véase[2]).

Partimos entonces de la forma de calcular la función $P(x)$ en lagrange:

$$P(x) = \sum_{k=0}^{n} f(x_k)\ L_{n,k}(x)$$

para cuyo cálculo computacional se requiere iterar desde $k = 0$ hasta n para hallar los valores de $f(x_k)$ y $L_{n,k}(x)$. Hasta este punto tenemos un algoritmo de orden n. continuando, para hallar la función $L_{n,k}(x)$ descrita arriba se lleva a cabo la siguiente productoria:

$$L_{n,k}(x) = \prod_{i=0\ i\neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

lo que nuevamente nos lleva a iterar desde $i = 0$ hasta n, dandonos una función de grado n dentro de cada iteración de la sumatoria anterior. De allí tenemos que por cada iteración n dentro de la sumatoria que describe $P(x)$ hay n iteraciones mas de $L_{n,k}(x)$. lo que nos deja un algoritmo de orden n^2.

como ya se sabe, en algunos casos las funciones exponenciales funcionan mejor que las lineales para valores pequeños (anteriores a la intersección de las dos funciones), lo que confirma el enunciado del que partimos y restringe la aplicación del método, o dado el caso, a hacer triangulaciones despues de aplicar un metodo de convex hull para reducir el numero de puntos a interpolar.

3.1.2. Interpolación con splines

Luego de haber visto que nuestro problema requiere de curvas suaves, y que las splines se han diseñado para resolver este problema, podemos tocar el problema de la interpolación con splines. En términos generales dado un arreglo de m coordenadas x, de la forma:

$$a = x_0 < x_1 < ... < x_{m-1} < x_m = b$$

teniendo también un arreglo de coordenadas y

$$y_0, y_1, \ldots, y_{m-1}, y_m.$$

Problema: Encontrar una función suave $\sigma(x)$ en el intervalo [a, b] cuyos valores coincidan con los valores de y en el arreglo anterior; es decir:

$$\sigma(x_i) = y_i, \quad i = 1, 2, 3, \ldots, m-1, m.$$

De esta forma, clasificamos las funciones para interpolar los puntos de acuerdo con los siguientes criterios:

- La función generada pasa por todos los puntos del arreglo de posiciones y que ya se mencionó.
- La función tiene derivadas continuas del grado necesario (para conectar segmentos del contorno).
- La función está determinada únicamente por el arreglo de puntos iniciales.
- Si se interpola con un polinomio, este debe ser de un grado tan bajo como lo sea posible para evitar al máximo errores propagados en las operaciones.

- La función deberá ser tan estable como sea posible a un cambio en uno de los puntos (recalcular el polinomio completo al cambiar uno de los puntos).
- La adición de puntos adicionales mejora la aproximación inicial(refinamiento de la matriz de puntos).

Resumen: Nuestro problema se sintetiza entonces en la reconstrucción de contornos a partir de una nube de puntos con base en métodos numéricos y splines que pasen por los puntos que representan dicho contorno. Se comienza con un contorno *brusco* obtenido a partir de un convex hull y se trabaja desde este para refinar el contorno de la figura.

4. Definición de la practica

La práctica es un proyecto de investicagion en el área de contornos de núbes de puntos y us diferentes usos y aplicaciones. Está estrechamente relacionada con ajuste de patrones y similares. El resultado del proyecto será tangible, se tratará de un proyecto de software libre, desarrollado inicialmente bajo la motivación de la materia de análisis numérico, que buscará abarcar inicialmente ciertos algoritmos particulares que involucren geometría computacional y métodos numéricos para ver luego hasta dónde se puede llevar esta fusión de técnicas. Así se desarrollarán en paralelo dos algoritmos para atacar este problema, a partir de la información de trabajos ya realizados en el área y en temas afines; se análizará su desempeño y las diferencias con otros algoritmos ideados para solucionar este tipo de problemas. Se presentarán informes de lo que se conoce como *perfil* de los algoritmos desarrollados; esto es, su desempeño bajo condiciones de prueba.

4.1. Alcance inicial del proyecto

Se tiene una nube de puntos sobre la que se va a trabajar, la idea es tomar dicha nube y mediante un convex hull hacer una aproximación inicial a su contorno, lo que tiene de innovador este proceso es que se busca "degenerar.^en convex, añadiendo o quitando puntos como sea necesario para hacer una aproximación más cercana al contorno. A partir de aqui se busca hallar el perímetro y el área de el espacio descrito por la nube de puntos. Esto es a grandes razgos lo que se busca hacer con la práctica.

Durante la creación de dichos algoritmos esperamos descubrir nuevas aplicaciones para estas técnicas y sobre todo para los resultados obtenidos de ellas.

4.2. Alcance del grupo 1

Este equipo partirá de la implementación del algoritmo Graham-Scan para hallar el convex hull, luego se hará el proceso anteriormente descrito para llegar a un contorno mucho mas aproximado y en este caso ya no convexo como esta dicho por definición para un convex hull. De allí se usarán los métodos numéricos de integración vistos en clase para determinar el área y el perímetro del contorno obtenido. se espera que esta solución sea computacionalmente rápida ya que esta basada en algoritmos que permiten rápidas paroximaciones al area de un poligono y se mescla con metodos numéricos de los cuales se conoce su eficiencia y criterios para una buena aplicación.

Objetivos Específicos

- Dada una nube de puntos hallar su convex hull mediante uno o más algoritmos
- Dado un convex hull, encontrar una forma de "degenerarlo"para hallar un polinogo (que puede ser no convexo) que se aproxime al contorno de una

figura

- Mediante triangulaciones o cualquier otra tecnica, refinar un convex hull obtenido de una nube de puntos.
- Usar el resultante del refinamiento del convex hull para hacer el calculo del área del polígono usando los métodos numéricos vistos en clase
- Hallar mediante la aproximacíon al contorno que se tiene, el perímetro de la figura dada.

4.3. Alcance del grupo 2

el grupo 2 se ocupará de asuntos algo diferentes pero no del todo distantes del alcance general. Se tomará un contorno, y de allí se procederá a suavizarlo, es importante tener en cuenta que el objetivo del grupo 1 no es obtener un contorno suave, sino, un contorno lo suficiente aproximado que permita hacer un calculo de área lo mas exacto y eficíente posible en terminos del numero de operaciones y la complejidad del algoritmo. asi pues al final del trabajo del grupo 2 se tendrá un contorno suavizado que corresponderá en mayor medida al contorno real dado por la nube de puntos.

Objetivos específicos

- Dada una nube de puntos hallar su contorno
- Dado el contorno de la nube de puntos, usar métodos que permitan suavizar dicho contorno
- Determinar la eficiencia de los metodos para hallar un contorno suave y hallar la mejor aproximación.
- Usar los resultados del trabajo del equipo 1 para suavizar el contorno dado

Es importante aclarar aquí que el trabajo del equipo 2 no parte estríctamente de los resultados del grupo 1 y por ello no hay problemas de tiempo ni coordinación, sino que se espera que la solucón obtenida sea genérica.

4.4. Organizacion del trabajo

como ya se mencionó, el trabajo esta dividido en 2 grupos que son:

Equipo 1: Eliel David Lobo, Hernan Dario Metaute, Sebastián Arcila (Líder).
Equipo 2: Cristian Camilo Isaza, Sergio Botero, Sebastián Arcila (Líder).
Asesor: Francisco Correa Zabala.

De esta forma esperamos abarcar lo suficiente como para llegar a una solución nueva para el problema plnateado y abordar el problema de manera que las dos soluciones encontradas se puedan integrar en un solo producto. la division

en equipo nos permite concentrar nuestra capacidades en diferentes microproblemas de una manera organizada y llegar a mejores soluciones de lo que seria trabajar todos en todo.

Por ser un proyecto de investigación, las entregas se hacen en capítulos que dan cuenta del proceso investigativo que se está llevando a cabo. Este primer capítulo es una introducción general a la geometría computacional y un acercamiento al trabajo que se va a ralizar. Los capítulos siguientes, pretenden explicar mas al detalle el trabajo de cada grupo desde dos áreas (que se traducen en nuevos capítulos para cada grupo). un área teórica donde se describa mas al detalle lo relacionado con los objetivos del grupo y lo que permitirá alcanzar la solución planteada. por otro lado se espera dar cuenta del proceso como tal y de los detalles de implementación que se den durante el proceso de creación de la solución.

al final habra conclusiones conjuntas de los dos equipos y se hará una relacíon de lo que se hizo para obtener conclusiones que permitan retroalimentar el trabajo realizado.

4.5. Entregables al final del proyecto

- Seudocódigo de los algoritmos.
- Implementaciones en C++ de los algoritmos.
- Paper planteado por nuestro asesor.

4.6. Extras planteados

Estos son los posibles aditamentos que pueden abordarse como una adición a la práctica. Se trata de trabajo posterior que no está contemplado en el alcance

- Un analizador gráfico en OpenGL que muestre como se comporta el algoritmo respecto a la nube de puntos a medida que itera.
- Una aproximación a la idea algorítmica para hallar el contorno en nubes de puntos en 3D. Desde aquí se planea plantear.
- El seudocódigo de esta aproximación a 3D
- Implementación en C++

Esperamos orientación de nuestro asesor.

Referencias

[1] Skiena, Steven S y Revilla, Miguel A. *Programming Challenges : The Programming Contest Training Manual* : Geometry, Computational Geometry (págs 291-337). 2003. New York : Springer 2003. 359p. TEXTS IN COMPUTER SCIENCE. ISBN 0387001638.

[2] Burden, Richard L y Faires, J. Douglas. *Análisis numérico* 7 ed. Mexico: Thomson Learning, 2002. 839p. ISBN 9706861343.

[3] Cormen, Thomas H. *Introduction to algorithms*. 2 ed. MIT Press y McGraw-Hill. ISBN 0-262-53196-8.

[4] M de Berg, M. van Kreveld, M. Overmars, y O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 2 ed, 2000.

[5] Shikin Eugene, Plis Alexander. . *Handbook on splines for the user*. Boca Ratón. CRC Press, 1995. ISBN 084939404X.

[6] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, 2 ed, 2000.

[7] Preparata, Franco y Shamos, Ian. *Computational Geometry: An introduction*. 1985, New York : Springer-Verlag. TEXTS AND MONOGRAPHS IN COMPUTER SCIENCE. ISBN 0-387-96131-3.

[8] Edelsbrunner, Herbert. *Algorithms in combinatorial Geometry*. 1987. Berlin: Springer-Verlag .MONOGRAPHS ON THEORETICAL COMPUTER SCIENCE. ISBN 3-540-19772.

[9] Melhorn, Kurt.*Data structures and algorithms 3: multi-dimensional searching and computational geometry*. 1984.

[10] De Berg Mark, Otfried Cheong, Van Krevld Marc, y Overmars Mark. *Computational Geometry* 3ra edición revisada. 2008 Springer-Verlag. ISBN 3-540-77973-6, 1st edition (1987): ISBN 3-540-61270-X

[11] Convex Hull Algorithms. Recurso en línea. `http://www.cse.unsw.edu.au/~lambert/java/3d/hull.html`. Consultado el 16 de agosto de 2009.

[12] 3D Convex Hull. Recurso en línea. `http://student.ulb.ac.be/~claugero/3dch/index.html`. Consultado el 18 de agosto de 2009.

[13] Selim G. Akl y Lyons Kelly A. *Parallel Computational Geometry*. 1993 Prentice-Hall. ISBN 0-13-652017-0.

[14] Computational Geometry Community Bibliography O'rourke. Recurso en línea `ftp://ftp.cs.usask.ca/pub/geometry`. Consultado el 25 de agosto de 2009.

[15] Computational geometry: theory and applications. Elsevrier. (Journal) Recurso en línea. http://www.elsevier.com/wps/find/journaldescription.cws_home/505629/description#description. Consultado el 25 de Agosto de 2009.

[16] Computational geometry and applications. World Scientific. (Journal) Recurso en línea. http://www.worldscinet.com/ijcga/. Consultado el 25 de agosto de 2009.