Algoritmos para tratamiento de mallas

La mayoría de los dispositivos de captura de información 3D que existen actualmente entregan la información en forma de nube de puntos (ref a Kyle, kinect y Sensores del chino). Es por ello que previo a la manipulación de la información tridimensional, es necesario procesar dicha nube de puntos para convertirla a formatos más manejables, como por ejemplo mallas triangulares.

Un típico procesamiento de malla, ya muy estudiado e implementado en bibliotecas como VcgLib [] o CGAL [] seria dada una nube de puntos de entrada, realizar un sub-muestreo y suavizado de la misma, calcular las normales en cada punto de la nube, y finalmente aplicar algoritmos de reconstrucción de malla.

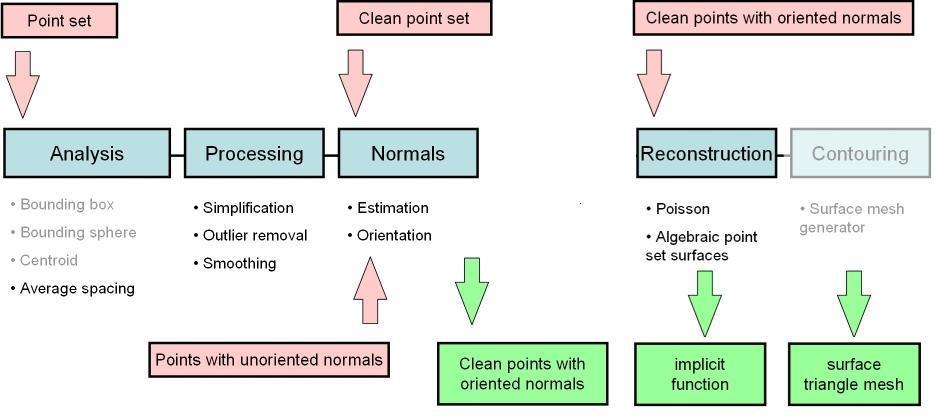


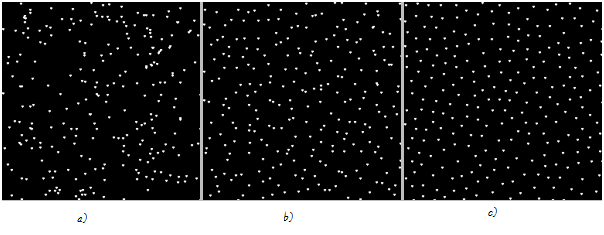
Figura: Típico flujo para el procesamiento de nubes de puntos (fuente: CGAL[])

Para la implementación de este modulo, se utilizaron algoritmos incluidos en VcgLib. Nos apoyamos fuertemente en MeshLab [] para visualizar y evaluar los resultados esperados. Particularmente se utilizaron los algoritmos de muestreo *Poisson-disk* para reducir y normalizar los puntos de la malla inicial, *Normal Extrapolation* para el cálculo de normales y reconstrucción de superficies de *Poisson* para la reconstrucción de la malla final.

* Muestreo Poisson-disk

El muestreo o sorteo de variables aleatorias es una técnica utilizada para una gran variedad de aplicaciones graficas, incluyendo dibujado (rendering), procesamiento de imágenes y de geometrías.

Particularmente, el muestreo Poisson-disk se utiliza para ubicación aleatoria de objetos en mundos artificiales, algoritmos de texturas procedurales y procesamiento de geometrías o mallas. Lo más interesante de esta técnica es que genera conjunto de puntos con buenas propiedades*: puntos suficientemente juntos, pero con la restricción de no estar más próximos unos de otros que una distancia mínima R predeterminada*.

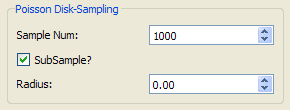


*Figura a) Posición x e y generadas aleatoriamente. Figura b) Imagen dividida en celdas. Puntos aleatorios generados en cada celda. Figura c) Muestreo Poisson-disk en 2 dimensiones.*

La idea básica de este algoritmo es generar puntos alrededor de los ya existentes en la muestra, y validar si pueden ser agregados al conjunto final en caso de no violar la regla de la mínima distancia a los vecinos. Se genera una grilla en 2 o 3 dimensiones dependiendo del escenario de aplicación, en la cual cada celda contendrá al final del proceso a lo sumo un punto. Una grilla adicional es utilizada para realizar búsquedas rápidas, y dos conjuntos de puntos son mantenidos durante el procesamiento para poder diferenciar los que han sido generados y los que aun necesitan procesamiento.

La implementación de VcgLib utilizada recibe 3 parámetros:

* La cantidad de puntos en la muestra. En este caso el radio o parámetro de cercanía es calculado en base a este parámetro.
* El radio, que es a su vez utilizado para calcular el tamaño de la muestra optimo en base a la malla inicial.
* Sub muestreo: indica si la muestra de Poisson es un subconjunto de la muestra inicial o si se deberán generar nuevos puntos aleatoriamente.

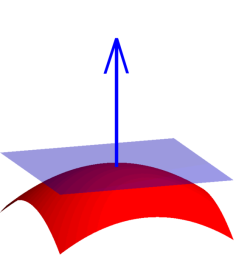


* Reconstrucción de normales

Este algoritmo computa las normales en cada elemento de un conjunto de puntos sin la necesidad de explorar la conectividad de los triángulos. Por ello es que es muy útil para objetos tridimensionales sin información de caras (*faces*).

Se detalla un pseudocódigo del método:

***Paso 1: Identificar los planos tangentes para aproximar localmente la superficie y estimar asi los vectores normales***

 Para cada vértice:

Calcular el centro geométrico (*centroide*) del plano tangente en el punto como el promedio de los K puntos más cercanos.

Calcular la normal asociada al centro geométrico. Se utiliza la matriz de covarianza en el punto contemplando los mismos K vecinos más cercanos de la muestra y los valores y vectores propios de la matriz de covarianza. Finalmente, ordenando los vectores propios, la estimación del vector perpendicular corresponde al vector propio de menor valor. Este método es conocido como PCA (*Principal Component Analisys*).

***Paso 2: construir grafo donde cada punto está conectado a los K vecinos más cercanos (grafo de Riemannian)***

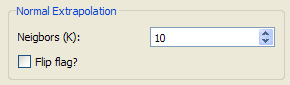
Se crea un grafo en cuyos nodos se guardan todas las aristas a los K vecinos más cercanos. Cada arista se pesa con el valor absoluto del producto escalar de la normal en el punto con la normal en cada uno de los K vecinos:

*fabs(nodoActual->normal . K\_Vecinos[n]->normal)*

***Paso 3: calcular el arbol de expansion minimo (MST) sobre el grafo de Riemannian y recorrerlo para orientar las normales***

Dado un grafo conexo, no dirigido, con sus aristas con un peso asignado, se llama árbol de expansion minimo al sub-grafo con forma de árbol que conecta todos los nodos con un peso total minimo. Contiene todos los nodos del grafo inicial. El grafo de entrada es el construido en el paso enterior y se utiliza el algoritmo de Kruskal, uno de los varios algoritmos glotones que resuelven el problema de encontrar un árbol de expansión minima de un grafo (referencias).

Una vez construido el MST, lo único que se hace es recorrer el árbol en orden y corregir el sentido de los vectores normales (multiplicando por *-1.0f*) en caso de ser necesario. La condición para efectuar dicha corrección se basa en el angulo del nodo siendo inspeccionado versus todas las direcciones de las normales de los vecinos conectados a dicho nodo.



* Reconstrucción de malla de Poisson

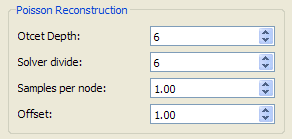
Finalmente, para reconstruir la malla a partir de la nube de puntos y sus normales se utiliza el algoritmo de Reconstrucción de Poisson.

Se computa una función indicador en 3 dimensiones definida de la siguiente forma:

Luego, se obtiene una reconstrucción de la superficie mediante la extracción de la *ISO-surface* (superficie de nivel en 3 dimensiones) al nivel apropiado.

La clave está en que hay una estrecha relación entre los puntos orientados (con sus normales) de la muestra y la función indicador de la muestra. Específicamente el gradiente de la función indicador es un espacio de vectores que valen cero casi en todo el espacio excepto en puntos cercanos a la superficie, donde es igual al vector normal a la superficie.

Es por eso que puntos orientados, pueden ser vistos como muestras del gradiente de la función indicador del modelo tridimensional en cuestión. Y es por este mismo motivo que el problema de reconstrucción de una malla puede ser visto como un problema de Poisson estándar: computar la función escalar F cuyo Laplaciano (o divergencia del gradiente) se iguala a la divergencia del espacio de vectores de las normales.

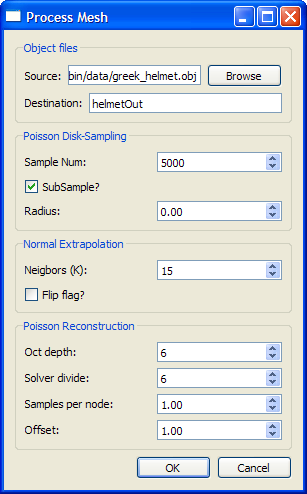


* Pruebas y resultados

Para validar el correcto funcionamiento de esta técnica utilizando los tres algoritmos descritos, utilizamos una malla inicial de 5021 vertices y 9608 caras triangulares. Cabe destacar que si bien se ha mencionado que la malla de entrada debe ser simplemente una nube de puntos, se pueden utilizar mallas con caras, solo que estas serán ignoradas e incluso removidas de la malla de salida del primer paso del procesamiento (Poisson-disk sampling).

Parametros iniciales

Luego de experimntar con varios juegos de datos iniciales durante varias ejecuciones del procesamiento, se fijaron de manera personalizada para la malla de entrada algunos parámetros clave. Dado que la muestra inicial tiene alrededor de cinco mil puntos, elegimos cinco mil como cantidad de muestras para el algorimso de Poisson-disk. Luego, para la extrapolación de normales se utilizaran K=15 vecinos para la toma de decisiones locales de aproximación.



Resultados

La aplicación de estos algoritmos resulto ser lo esperado, al menos en cuanto a la parte esctructural de cada malla que va siendo procesada en cada paso. No se llego a procesr mallas de ambientes tridimensionales escaneados para luego ser mapeados con la herramienta. **Creemos que esto debería ser validado en una etapa posterior de este proyecto ya que ese es el principal contenido del presente modulo.**

|  |  |  |
| --- | --- | --- |
| **Fase** | **Vértices** | **Caras** |
| Nube inicial | 5021 | 9608 |
| Luego Poisson-disk | 1776 | 0 |
| Luego Extrapolación Normales | 1776 | 0 |
| Luego de Poisson-recontruction | 1959 | 3910 |

Tabla: Comparacion de estructura de malla de entrada y salida de cada fase.

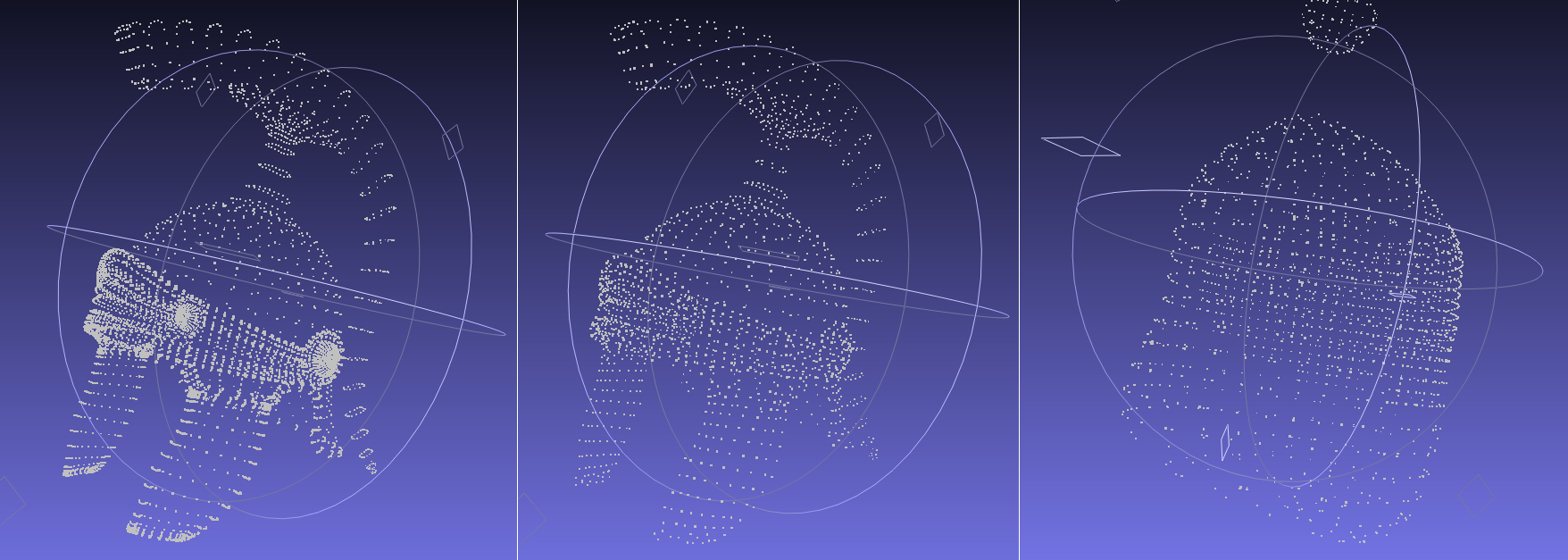


Figura: 1) Nube inicial con 5021 vertices. 2) Resultado de muestreo Poisson-disk con 1776 vertices. 3) Luego de extrapolar normales y reconstruir la malla con 1959 vertices y 3910 caras.

Se observaron buenos tiempos computacionales de respuesta. Si bien la malla utilizada no es de un tamaño considerable, estamos hablando de algoritmos de orden relativamente alto.

Referencias

VcgLib "*Visual computer graphics Library*”. <URL:http://vcg.sourceforge.net/tiki-index.php>. Biblioteca portable escrita en C++ para manipulación, procesamiento y despliegue con OpenGL de mallas triangulares. Liberada bajo licencia GPL por parte del “*Visual Computing Lab* “ (VCGLab) del "Institute of the Italian National Research Council " <URL:http://www.isti.cnr.it>.

MeshLab

Software de código abierto para manipulación de mallas tridimensionales en varios formatos.