+

1. Excelentísimos miembros del Tribunal, profesores de la Universidad de Navarra, damas y caballeros, voy a proceder a la lectura de la Tesis Doctoral titulada “Study of parallel techniques applied to surface reconstruction from unorganized and unoriented point clouds”.

2. Esta presentación estará organizada de la siguiente forma: [*] primero se introducirán algunos conceptos previos al problema de reconstrucción de superficies y se mencionarán algunos de los trabajos más relevantes en el área; [*] posteriormente se enumerarán los objetivos y limitaciones de esta tesis. [*] Luego se describirán las propuestas desarrolladas en este trabajo y al final de cada apartado se presentarán algunos de los resultados obtenidos. [*] A modo de resumen, se mostrará un vídeo de la batería de modelos reconstruida. [*] Por último se presentarán las principales conclusiones de esta tesis y se definirán algunas líneas de trabajo a seguir en el futuro.

3. [Introducción]

4. La reconstrucción de superficies puede definirse como el proceso que, partiendo de un conjunto dado de puntos que han sido muestreados desde una superficie cualquiera, construye otra superficie tal que los puntos de entrada pertenecen a esta nueva superficie, o bien ésta aproxima a la superficie inicial. Dependiendo de la naturaleza de la reconstrucción, podemos considerar la clasificación de las técnicas existentes entre interpolantes y aproximantes.

5. Los puntos pueden ser obtenidos mediante diversas técnicas, pero una de las más comunes es el uso de escáneres que digitalizan la superficie del objeto. En esta lámina se muestran dos escáneres usados en el TECNUN: uno manual y otro basado en imágenes estereoscópicas. El vídeo que se muestra es un ejemplo del proceso de digitalización.

6. Comparando las técnicas existentes según la clasificación anterior, tenemos que las interpolantes directamente reconstruyen la superficie usando los puntos de entrada como vértices del mallado, por lo que el nivel de detalle del mismo dependerá del muestreo del modelo. Por otra parte, las técnicas aproximantes generan una superficie que se ajusta a la escaneada, pudiendo variar su calidad, aunque en los casos de mayor resolución pueden requerir más memoria que las otras técnicas. Al buscar ajustarse

+

al conjunto inicial, las técnicas aproximantes son más tolerantes a los problemas de muestreo, como ruido y outliers, que las interpolantes.

7. Entre las técnicas interpolantes más conocidas encontramos la triangulación de Delaunay. Dado un conjunto de puntos P en R^n, la triangulación de Delaunay es tal que no existe ningún punto del conjunto que esté dentro de la circum-hiperesfera de ningún n-simplex de P. Un n-simplex es el análogo n-dimensional de un triángulo (en el caso 2D). Por otra parte, el diagrama de Voronoi es el dual de la triangulación de Delaunay, es decir, cualquiera de los dos puede calcularse partiendo del otro. En este diagrama, cada celda comprende la región del espacio que está más cerca de un punto que de cualquier otro. La triangulación de Delaunay es ampliamente utilizada porque posee características ideales para muchas aplicaciones, como la proporción de sus triángulos.

8. Entre los principales trabajos que hacen uso de triangulaciones de Delaunay están los mostrados en esta lista, donde uno de los más conocidos es el Power Crust.

9. Un enfoque interesante de reconstrucción interpolante es el uso de triangulaciones locales. Éstas buscan reconstruir la superficie un punto a la vez, generando los triángulos que están conectados al mismo. Los trabajos que se mostrarán son ejemplo de este tipo de técnicas: [*] Crossno y Angel van expandiendo un frente mediante la triangulación de los puntos de la frontera, [*] el grupo de Gopi usa triangulaciones de Delaunay en 2D, y [*] Linsen y Prautzsch asumen cierta uniformidad en los datos y simplemente unen los puntos de la vecindad.

10. Las técnicas aproximantes pueden a su vez dividirse en dos grupos muy generales. El primero está basado en la extracción de una isosuperficie de una función implícita (una función de distancia, por ejemplo). Uno de los trabajos más conocidos es el de Hoppe, a la vez que la reconstrucción de Poisson es referente en el área. Para poligonizar la isosuperficie se suelen utilizar algoritmos como el Marching Cubes.

11. El otro grupo está conformado por las técnicas dinámicas, donde un modelo deformable es ajustado a la superficie objetivo guiado por conjuntos de fuerzas internas (provenientes del modelo) y externas (definidas por los datos).

+

12. Por último, la programación en la GPU es una herramienta utilizada en los últimos años para acelerar la ejecución de los algoritmos existentes dado el alto nivel de paralelismo que poseen las tarjetas gráficas actuales. Así por ejemplo, encontramos implementaciones en la GPU del Marching Tetrahedra y, recientemente, de la reconstrucción de Poisson. [Agua]

13. [Objetivos] Una vez que se ha esbozado el estado del arte de la reconstrucción de superficies, se enumerarán los principales objetivos de esta tesis y se delimitarán las posibles propuestas.

14. Primero, dada la necesidad de procesar conjuntos de puntos cada vez más grandes, se estudiará la aplicación de técnicas paralelas en el diseño de algoritmos de reconstrucción de superficies a partir de nubes de puntos, que no poseen ningún tipo de información adicional, como sus normales. Segundo [*], se buscará la optimización de los métodos auxiliares que se utilicen en la reconstrucción. Y por último [*], y en la medida de lo posible, los algoritmos desarrollados serán implementados en la GPU.

15. Las limitaciones del estudio se centran en el conjunto de datos de entrada (puntos 3D sin organización ni información adicional) y en minimizar el número de parámetros requeridos. [*] Tampoco se espera que los métodos estudiados sean tolerantes a altos grados de problemas de muestreo. [*] Y finalmente, los métodos desarrollados deben ser escalables respecto al conjunto de datos.

16. [Propuestas] En los siguientes minutos se explicarán las propuestas de esta tesis y los aportes presentados en cada una de ellas.

17. Durante el desarrollo de este trabajo, se estudió el procesamiento de triangulaciones locales en paralelo. La independencia de los cálculos entre puntos las hacen particularmente favorables para una ejecución concurrente. Entre las técnicas antes citadas, se eligió la presentada por Gopi, por tener un mayor fundamento teórico y hacer uso de triangulaciones de Delaunay. Como se mencionó anteriormente, la triangulación de Delaunay en 3D está formada por tetrahedros, no por triángulos. [*] Para solventar este problema y generar un mallado de triángulos, el método de Gopi se basa en la reducción de la dimensión del problema, proyectando cada punto y su vecindad sobre su plano tangente, calculando

+

entonces una triangulación de Delaunay en 2D. La primera propuesta que se presentará lleva por nombre GPU Local Triangulation.

18. La reconstrucción local comprende el cálculo de las vecindades de cada punto, la estimación de las normales, la proyección sobre el plano tangente, el cálculo del ángulo de cada vecino, el ordenamiento radial de los vecinos, la creación del anillo de vecinos y la triangulación local. Al final del método, las normales son orientadas uniformemente.

19. El cálculo de las vecindades se realiza utilizando los k vecinos más cercanos a cada punto, dentro de un radio de búsqueda definido por la distancia media de muestreo del modelo.

20. En esta tesis se propone la paralelización de las fases locales a cada punto, a saber [*], las que se muestran en la transparencia.

21. [Estimación de la normal - T.A.]

22. Como se mencionó anteriormente, el método propuesto por Gopi se basa en la construcción de la triangulación local de Delaunay en 2D de un punto con los vecinos proyectados sobre su plano tangente. En esta tesis el plano, mejor dicho, su normal, es estimada utilizando una variante del método propuesto por Hoppe. El método original calcula el plano tangente utilizando análisis de componentes principales. En este trabajo [*] se ha sustituido por un análisis de componentes principales ponderado, que regula la influencia de los puntos más alejados. Estos puntos pueden perturbar la estimación en zonas próximas a irregularidades u hojas de la superficie no contiguas al punto.

23. [Proyección de la vecindad en el plano tangente - T.A.]

24. En este paso, cada vecino es proyectado sobre el plano tangente del punto utilizando la rotación mínima hasta éste. En la implementación del método se ha sustituido esta rotación por una proyección ortogonal del vecino sobre el plano tangente, y un escalado posterior.

25. [Cálculo del ángulo entre vecinos - T.A.]

+

26. Para realizar la triangulación, es necesario crear un anillo con los vecinos proyectados, ordenándolos circularmente alrededor del punto central. En esta fase, el ángulo entre cada vecino y el más cercano al punto es calculado. El vecino más cercano es, por definición, siempre un vecino de Delaunay, por lo que se usa como referencia para diversas operaciones. El ángulo sólo es necesario para el ordenamiento de los vecinos alrededor del punto, por lo que se ha eliminado el cálculo del arcocoseno; hemos denominado a este nuevo valor la posición angular del vecino.

27. [Ordenamiento y ensamblado del anillo - T.A.]

28. Como se mencionó anteriormente, los vecinos son ordenados alrededor del punto central usando la posición angular. En la implementación en GPU usando shaders se realizó una modificación al algoritmo Bitonic Merge Sort para extenderlo al ordenamiento de múltiples listas en simultáneo. Finalmente, estos vecinos ordenados forman un anillo alrededor del punto central, anillo que será usado en la siguiente fase para la triangulación local.

29. [Triangulación local de Delaunay en 2D - T.A.]

30. La triangulación local propuesta por Gopi consiste en un proceso recursivo de invalidación de puntos como vecinos de Voronoi respecto al punto central, el vecino anterior y el siguiente. Este proceso no determina la validez de un punto, sino simplemente descarta vecinos que no pertenecen a la triangulación; es por ello que debe ser aplicado de forma recursiva. [*] En esta tesis hemos demostrado que el proceso es correcto desde el punto de vista teórico, y a partir de ahí, que el orden de la validación no influye en el resultado final, por lo que es posible desarrollar un proceso de validación en paralelo e implementable en la GPU.

31. En esta lámina se muestra el algoritmo utilizado en la implementación del método con shaders. Esta versión es la que más aportes involucra, por lo que es la que se ha escogido para presentarse. [*] Basándose en una estructura de lista doblemente enlazada almacenada en una textura, cada vecino es marcado como inválido o no inválido. Esta estructura es creada almacenando el offset respecto a los vecinos no inválidos anterior y siguiente. [*] Este proceso se repite iterativamente hasta que ningún

+

punto es marcado como inválido. En ese momento, la triangulación local de Delaunay comprende, en orden, los vecinos no invalidados. En esta figura, los puntos en verde son vecinos no inválidos, mientras que los rojos corresponden a puntos ya invalidados.

32. [Orientación de las normales - T.A.]

33. La normal estimada durante la triangulación local no necesariamente tiene una orientación correcta. La orientación de las normales no influye en la reconstrucción, pero puede ser necesaria posteriormente dependiendo del uso que se le dé al mallado. El proceso de orientación de normales se realiza como última etapa de la reconstrucción: tomando una normal como correcta, su orientación se transmite a través del mallado usando su conectividad como grafo de propagación.

34. [Resultados] A continuación se mostrarán algunos resultados del método GLT. En esta primera imagen [*], se muestra el modelo del Caballo, [pausa y *] así como un detalle del mallado, donde se aprecia la calidad de los triángulos generados.

35. En esta lámina y la siguiente, se compara el método propuesto (a la izquierda) con la reconstrucción de Poisson (a la derecha). Ambos métodos generan un mallado de calidad semejante, aunque el GLT es capaz de recuperar algunos detalles que el Poisson no. Por otro lado, este último genera un mallado algo más suave. También puede observarse que el método propuesto es unas seis veces más rápido que la reconstrucción de Poisson. Como nota, en esta presentación todos los modelos han sido renderizados usando un sombreado flat, para poder ver mejor la calidad de las mallas de triángulos.

36. A la izquierda, el modelo del Happy Buddha reconstruido usando el GLT, y a la derecha usando Poisson. La proporción de los tiempos de reconstrucción es similar al modelo anterior, y las observaciones acerca de la calidad del mallado son las mismas: GLT es capaz de recobrar más detalles mientras que Poisson genera una superficie más suave.

37. Como muestra de los tiempos de ejecución de algunas de las fases, puede verse el incremento de velocidad de los métodos al utilizar la GPU, especialmente CUDA. En todas las gráficas, en el eje horizontal se muestra el tamaño de los modelos expresado en miles de puntos, mientras que el eje de

+

las ordenadas representa el tiempo de cálculo en segundos. Como caso particular, [*] notamos que la triangulación local del Happy Buddha tardó casi el doble usando CUDA que shaders, al contrario que en el resto de los modelos; suponemos que esto es debido a que la particular distribución de los puntos de entrada genera una peor coherencia de datos en la caché con las estructuras de CUDA.

38. Esta gráfica muestra los tiempos de las diversas etapas de la reconstrucción para un modelo grande, salvando la orientación de las normales, que fue realizada a parte por motivos de memoria (el modelo tiene 3,3 millones de puntos y unos 6,8 millones de triángulos). Las dos etapas en azul, las más grandes, son las etapas que se realizan únicamente en la CPU y corresponden a un 80% del tiempo total de reconstrucción. Las proporciones entre las diferentes etapas de la reconstrucción se mantienen más o menos constantes en el resto de los modelos.

39. Finalmente, se muestra la relación entre los tiempos de las tres implementaciones del método en CPU, en GPU usando shaders y en GPU usando CUDA, para una batería variada de modelos, que van desde los 20mil hasta los 880mil puntos. Modelos más grandes han sido reconstruidos aunque por razones de escala no se muestran en esta gráfica.

40. [PWLOP] El siguiente apartado estudia un operador de proyección de puntos que transforma un conjunto de puntos creando un nuevo conjunto más uniformemente distribuido y con menos ruido que aproxima a un conjunto objetivo dado. [*] En este ejemplo ilustrativo, los puntos negros representan al conjunto inicial, y los grises la proyección que, partiendo de un estado aleatorio, converge a una aproximación uniforme de los puntos iniciales.

41. El estado del arte menciona dos variantes de este operador de proyección, aunque, por motivos de tiempo, en la presentación se estudiará únicamente la segunda, la Weighted Locally Optimal Projection (o WLOP), para un mejor entendimiento de nuestra propuesta. De todas formas, el WLOP es una generalización del otro. El operador es iterativo y está formado por dos términos: [*] uno de atracción, que acerca los puntos al conjunto objetivo, ajustándose a los mismos, [*] y uno de repulsión, que evita que los puntos creen cúmulos y busca que se distribuyan uniformemente. Es importante destacar que en cada iteración [*], el operador sólo depende del estado anterior de los puntos, por lo que se puede intuir la posibilidad de paralelización. Aún así, el número de cálculos es abrumante y difícil de realizar

+

en el hardware gráfico. [*] La función theta es una función de peso similar a una gaussiana, que pondera la influencia de los puntos dependiendo de la distancia. [*] Por su parte, v sub j puede verse como una medida de atracción de los puntos del conjunto original que varía dependiendo de la densidad de los puntos más cercanos. [*] w sub i cumple una función similar, pero en el conjunto proyectado, ayudando a evitar la creación de zonas muy densas.

42. Analizando el operador de proyección, vemos que el término v sub j es propio del conjunto P y no cambia a lo largo de la ejecución del operador; es decir, ¡podría ser calculado únicamente en la primera iteración! Pero, por otra parte, w sub i varía en cada iteración. El radio de soporte de la función de peso no abarca todos los puntos del conjunto, por lo que un gran porcentaje de estos pesos es cero. Es posible entonces reducir el número de cálculos determinando los vecinos que caen dentro del soporte de la función para cada punto, por ejemplo, con un *k*-NN. Desafortunadamente, las implementaciones en GPU de este tipo de operaciones todavía no son tan eficientes como se esperaría.

43. En esta tesis hemos propuesto dos optimizaciones al operador de proyección: la primera consiste en precalcular v sub j al inicio del proceso de proyección. La segunda reduce el número de operaciones realizadas durante el cálculo de w sub i a la vez que se permite su implementación en la GPU. [*] Hemos introducido el concepto de vecindad de soporte extendida, la cual comprende no sólo los puntos dentro del radio de soporte, sino también aquéllos que pudieran caer dentro del mismo en alguna de las iteraciones. De esta forma, es posible calcular la vecindad de soporte de la función una única vez, en lugar de hacerlo en cada iteración.

44. Para que esta solución sea válida, es necesario asumir un conjunto inicial aproximado que garantice que los puntos no se mueven excesivamente. Éste puede ser calculado fácilmente mediante una voxelización de la nube de puntos inicial. Gracias a esta vecindad extendida, es posible precalcular todos los puntos del soporte de la función de peso. Este cambio permite, además de acelerar cada iteración, [*] implementar el operador en la GPU. Este nuevo operador se ha denominado Parallel WLOP.

45. [Resultados] En esta primera comparativa, se observa el comportamiento del operador original (en el centro) y el Parallel WLOP (a la derecha) ante un conjunto inicial aleatoriamente muestreado de la

+

nube de puntos. Es posible apreciar que, si bien logra una proyección aproximada, no se distribuye uniformemente, aunque el operador original tampoco lo logra en algunos casos.

46. En esta segunda comparativa, se parte de un conjunto inicial aproximado por voxelización (a la izquierda), garantizando ahora que el Parallel WLOP genere una distribución uniforme de puntos. Ambos métodos mejoran el tiempo de cómputo principalmente por requerir menos iteraciones, además de garantizar también que el WLOP distribuya los puntos más uniformemente.

47. En esta gráfica se muestra una comparativa de los tiempos del operador original, del nuevo operador implementado en la CPU, y de su versión acelerada por GPU ([*] la barra verde pequeña que casi no se ve ;)).

48. Como experimento adicional, se introdujo una pequeña modificación al operador propuesto para ver si era posible su uso con conjuntos iniciales aleatorios. En lugar de calcular la vecindad extendida una única vez al inicio, ésta se recalcula cada cierto número de iteraciones, para actualizar los puntos del soporte. En esta transparencia se observa la proyección del Happy Buddha a partir de una muestra aleatoria de puntos. De izquierda a derecha: el Parallel WLOP original, con el recálculo de las vecindades extendidas y el WLOP. Se puede ver cómo el nuevo enfoque genera resultados incluso mejores que el operador original, a expensa de un mayor tiempo de cómputo, pero que sigue siendo menor que el de éste.

49. [Reconstrucción híbrida] Como se explicó al inicio de esta presentación, los métodos interpolantes son susceptibles a problemas de muestreo como ruido y outliers. En esta tesis hemos utilizado el operador de proyección de puntos anteriormente descrito para mejorar la calidad de los datos de entrada. Esta combinación ha sido clasificada como un método híbrido dado que, si bien la reconstrucción se hace interpolando un conjunto de puntos, dichos puntos son una aproximación del conjunto inicial.

50. Se muestran ahora un par de imágenes descriptivas de los resultados del método híbrido. En esta primera se compara el modelo de la Mano; a la izquierda la reconstrucción original, con 327mil puntos, y a la derecha, una aproximación con 107mil muestras.

+

51. La segunda figura muestra la reducción de ruido tras aplicar el operador de proyección. La imagen de la izquierda es el modelo original del Noisy Foot, la imagen central muestra una reconstrucción tras proyectar sobre sí mismo el conjunto de puntos con una vecindad de 64 puntos. La última imagen es la reconstrucción tras la proyección con una vecindad de 128 puntos. Hemos observado que el tamaño de la vecindad ha de ser considerablemente más grande en los modelos ruidosos. [Agua]

52. [Multi-balones] El último apartado de la propuesta comprende las técnicas dinámicas de reconstrucción, más específicamente las técnicas basadas en balones. Un balón es, básicamente, un modelo deformable, usualmente una superficie de subdivisión, que evoluciona bajo la influencia de fuerzas externas (definidas por el conjunto de datos) y fuerzas internas (propias del modelo deformable), con el objetivo de minimizar una función de energía como la mostrada.

53. Dos de los principales trabajos usando balones son el Intelligent Balloon y el Competing Fronts, pero existen modelos donde dichas técnicas son "poco naturales", porque reconstruyen pequeños detalles antes de recuperar propiedades globales más importantes. [*] En esta tesis hemos propuesto un esquema de evolución en paralelo usando múltiples semillas, que permitiría extraer de una forma más natural la geometría y topología del modelo, partiendo de las propiedades más generales al detalle fino.

54. El flujo de trabajo del método propuesto es muy similar al de los balones clásicos: una etapa de inicialización, la resolución de los términos de atracción y tensión, detección de colisiones y control topológico. Los últimos cuatro pasos son efectuados de forma iterativa hasta completar la reconstrucción. La diferencia básica, además de algunos detalles en el proceso evolutivo, [*] es la presencia de varios balones evolucionando de forma concurrente.

55. [Inicialización - T.A.]

56. La etapa de inicialización comprende la creación de una función de distancia a los puntos de entrada, y de una función de satisfacción que indica al método el nivel de ajuste del modelo deformable. La función de satisfacción está almacenada en una estructura jerárquica cuya resolución se ajusta al nivel de detalle del modelo.

+

57. [Términos de atracción y tensión - T.A.]

58. Los trabajos previos utilizan el concepto de frente activo durante la minimización para reducir el tamaño del sistema. En este trabajo hemos usado un esquema global para mejorar la calidad del mallado final y reducir los problemas de seguimiento de los frentes. Para reducir el coste computacional de un problema tan grande, la minimización se ha dividido en dos etapas separadas: una de atracción y una de tensión, que pueden ser resueltas de formas más simples.

59. En el término de atracción, los vértices del modelo se mueven en la dirección de la normal hacia la superficie objetivo. Durante la evolución, pueden producirse situaciones problemáticas como túneles y agujeros, ya que son interpretadas de forma similar pero donde el modelo debe comportarse de una manera diferente. La detección de estas zonas ha sido realizada mediante una búsqueda local alrededor de cada vértice. La interpretación de cada caso recae en un término de modulación de gradiente, que frena el avance del vértice en las zonas de agujeros.

60. El término de tensión mantiene la calidad del mallado combinando un operador Laplaciano para el área de los triángulos y un operador de remallado adaptativo para la conectividad. Así mismo, un operador de refinado aumenta la resolución del mallado únicamente en las zonas de mayor detalle.

61. Los operadores de remallado tradicionales crean mallados excesivamente uniformes, y esto es un problema dado que incrementa innecesariamente la resolución del modelo. [*] En esta tesis hemos desarrollado un nuevo operador adaptativo basado en la ponderación de las longitudes de la vecindad de cada arista, permitiendo así un mejor ajuste a los cambios de resolución del modelo.

62. [Detección de colisiones y topología - T.A.]

63. Después de cada fase de la evolución, las regiones activas del modelo son comprobadas entre sí para detectar colisiones; esto permite incrementar el genus del modelo en las zonas en colisión. Por último, es posible recuperar los agujeros del modelo, usando la técnica de detección anteriormente descrita. [*] Se muestra ahora un ejemplo de la reconstrucción de un modelo sintético y el cambio topológico debido a la unión de frentes en colisión.

+

64. Este vídeo muestra el resultado de la reconstrucción del modelo de la Mano utilizando 23 semillas. Cada una de ellas evoluciona y se va ajustando a la superficie objetivo. Cuando dos balones colisionan, éstos son fusionados, aumentando el genus del modelo en algunos casos. [Esperar un poco a que esté casi reconstruido] En este ejemplo, sin embargo, el término de tensión frena la evolución en zonas finas [*] como la muñeca de la mano.

65. [Vídeo de resultados]

66. Por último, se presentará un vídeo en el que se muestran la mayoría de los modelos que conforman la batería de pruebas de esta tesis. Dichos modelos han sido reconstruidos usando el método GLT y el método híbrido. [T.A.]

67. Aunque se muestra la información de cada modelo, iré comentándola uno a uno para que puedan centrar su atención en las figuras si lo desean. [*] Stanford Bunny, 35mil puntos, 0,7 segundos; Armadillo, 172mil puntos, 3,6 segundos; Mano, 327mil puntos, 6,4 segundos; Happy Buddha, 543mil puntos, 11 segundos; Torre, 252mil puntos, 7 segundos; Caballo, 48mil puntos, 0,9 segundos; Blade, 882mil puntos, 18,4 segundos; Dragón del EuroGraphics 2007, 240mil puntos, 5 segundos; Asian Dragon, 3,3 millones de puntos, 75,9 segundos; Stanford Dragon, 437mil puntos, 8,9 segundos; Dragón del EuroGraphics 2011, 1,1 millones de puntos; 15,3 segundos; Ángel, 237mil puntos, 5,6 segundos; Noisy Foot, 20mil puntos, 1,52 segundos y el Running Shoe de 51mil puntos en 1,06 segundos.

68. [Conclusiones] A continuación se presentan las principales conclusiones de esta tesis y las posibles líneas futuras de investigación.

69. Se muestra que las triangulaciones locales son válidas en el diseño de reconstrucciones globales y particularmente útiles en el desarrollo de métodos paralelos. [*] Se presenta un algoritmo paralelo de reconstrucción de superficies basado en triangulaciones locales de Delaunay y su implementación en la GPU. [*] En esta tesis se demuestra que el método de verificación de vecindades de Voronoi es adecuado para procesamientos en paralelo.

+

70. Adicionalmente, se han realizado varias optimizaciones en las diversas etapas de la reconstrucción local. [*] Se presenta un operador paralelo de proyección de puntos y su implementación en la GPU que muestra un aumento de velocidad medio de entre 4 y 10 veces respecto al operador original. [*] Se ha usado este operador para mejorar la calidad de la nube de puntos a reconstruir con el GLT, creando así un método híbrido.

71. Esta tesis también estudia la aplicación de técnicas dinámicas en la reconstrucción de superficies, presentando un nuevo paradigma basado en el uso de múltiples semillas. [*] Se propone la separación de las fuerzas externas e internas para recuperar mejor las propiedades globales del modelo. [*] Y se ha desarrollado un operador de remallado adaptativo para superficies con múltiples resoluciones.

72. Como posibles líneas futuras de investigación, se propone estudiar técnicas que garanticen un mallado topológicamente correcto en el caso del método GLT, [*] estudiar el uso de varias distancias de muestreo e [*] investigar modelos paralelos de cálculo de las vecindades y propagación de las normales.

73. Respecto al método híbrido, sería interesante el estudio de éste en la generación de mallados multiresolución. [*] Así también, es posible profundizar todavía más en el estudio de las técnicas dinámicas, por ejemplo, pudiendo mejorarse la formulación del modelo del balón. [*] Por otra parte, el término de tensión puede generar sistemas muy grandes, donde el esquema de frentes puede aportar una útil ayuda. [*] Por último, la bibliotecas de mallados utilizada no es thread-safe, por lo que no se ha podido implementar una versión paralela del método de multibalones.

74. [Gracias] [Esperar a la mitad del GR] Con esto concluye mi presentación, y quedo a disposición de los miembros del Tribunal para que formulen las preguntas y dudas que consideren oportunas. [Agua]

75. [Preguntas]