REMUESTREO ESTRUCTURADO DE CONTORNOS DE HUECOS EN SUPERFICIES 3D DE OBJETOS DE FORMA LIBRE UTILIZANDO BRESENHAM

ESTRUCTURED RESAMPLING OF CONTOURS OF HOLES IN FREE FORM OBJECTS 3D SURFACES USING BRESENHAM

JOHN WILLIAN BRANCH

Ph.D., Universidad Nacional de Colombia – Sede Medellín, jwbranch@unal.edu.co

GERMÁN SÁNCHEZ

M.Sc., Universidad del Magdalena, gsanchez@unimagdalena.edu.co

PEDRO ATENCIO

Ing., Universidad Nacional de Colombia – Sede Medellín, psatencioo@unal.edu.co

Recibido para revisar Abril 19 de 2011, aceptado Diciembre 7 de 2011, versión final Enero 26 de 2012

RESUMEN: La etapa de integración dentro del proceso de reconstrucción tridimensional de objetos de forma libre, requiere de la descripción, análisis y corrección de huecos en superficies 3D. Ciertas evaluaciones cuantitativas en este tema implican contar con conjuntos de datos espaciados de forma regular o contenidos en estructuras que garanticen dicha propiedad, por ejemplo voxels, octrees o mallas estructuradas. Lograr lo anterior requiere un proceso de re-muestreo de los puntos que conforman el contorno del hueco en la superficie 3D. En este trabajo se describe un método para obtener conjuntos estructurados de puntos, a partir de los datos de contornos de huecos en objetos de forma libre. El método inicia con el ajuste de una curva NURBS al conjunto inicial de puntos con el fin de asegurar la suavidad del contorno, de lo cual se obtiene un conjunto de puntos ajustados. Finalmente se utiliza el algoritmo de discretización de Bresenham para obtener el conjunto de puntos estructurados. Los resultados obtenidos muestran que el método desarrollado asegura que el conjunto final de puntos estructurados preserven la forma del contorno original con altos niveles de detalle.

PALABRAS CLAVE: Visión por computador, computación gráfica, reconstrucción tridimensional, Llenado de huecos.

ABSTRACT: Integration stage in the process of three-dimensional reconstruction of free-form objects requires the description, analysis and correction of holes in 3D surfaces. Some quantitative assessments in this issue involve having data regularly spaced or contained in structures to ensure that property, for example, voxels, octrees or structured grids. Achieving this requires a re-sampling of points that make up the contour of the hole in 3D surface. This paper describes a method to obtain structured sets of points from contour data of holes in free-form objects. The method begins with fitting a NURBS curve to the initial set of points in order to ensure the smoothness of the contour, thus obtaining an adjusted set of points. Finally, it is used the Bresenham discretization algorithm to obtain the set of structured points. Results show that the developed method ensures that the final set of structured points preserve original contour shape with high levels of detail.

KEYWORDS: Computer vision, computer graphics, tridimensional reconstruction, hole filling.

1. INTRODUCCIÓN

el proceso de reconstrucción tridimensional mediante el cual se obtienen datos de forma, volumen y textura de objetos del mundo real para crear a partir de éstos, modelos digitales que puedan ser evaluados a través de métodos numéricos en un sistema de computación, actualmente es por sí mismo un problema abierto, esto debido en gran parte a la limitación actual de no contar con un sensor ideal que no altere las muestras obtenidas como menciona Sánchez en [1]. Por lo cual, obtener un modelo digital preciso implica tratar con múltiples problemas en cada una de las etapas del proceso de reconstrucción tridimensional.

Sumado a la limitación del sensor, características propias de los objetos de forma libre como superficies con características ópticas complejas, zonas de la

superficie con poca accesibilidad para el sensor, condiciones inadecuadas de iluminación, entre otras, generan anomalías en los modelos reconstruidos. Según Branch [2] estas anomalías se pueden ser clasificadas en *ruido, huecos* y *redundancia*. Generalmente estas anomalías son corregidas en una etapa llamada integración dentro del proceso de reconstrucción 3d, según explica Campbell en [3].

La corrección de huecos o *llenado de huecos* es un tema con amplia investigación en la comunidad científica, y prueba de ello es la gran cantidad de trabajos realizados al respecto desde mediados de los 90's hasta el presente. Sin embargo, la mayoría de trabajos se han enfocado en resolver el problema del llenado de huecos, i.e. Interpolar o aproximar los datos faltantes en la zona del hueco. Lo anterior no tiene en cuenta una limitante general del proceso de reconstrucción 3d, la cual radica en la necesidad de la intervención de un usuario para llevarla a cabo. Esta limitante repercute en la capacidad de automatizar dicho proceso. Específicamente en el proceso de llenado de huecos se requiere que un usuario seleccione los huecos que van a ser corregidos, debido a que se parte del hecho de que en modelos tridimensionales de objetos de forma libre, habrán huecos pertenecientes a la superficie real y por lo tanto no deben ser "llenados", como se puede apreciar en la Figura 1.

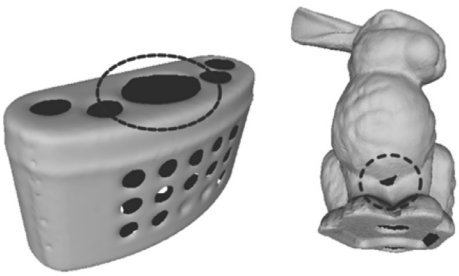

Figura 1. Hueco perteneciente al objeto, el cual no debe ser llenado (izquierda). Hueco que debe ser reparado (derecha).

El problema de determinar que huecos deben ser llenados o no en un modelo 3D, radica en diferenciar que huecos pertenecen o no a su superficie, esto sin duda, implica tener una previa caracterización o descripción de huecos producto de errores (*anomalías*) y de huecos reales. Ahora, caracterizar estas discontinuidades en la superficie, implica realizar mediciones sobre las mismas. En este marco de ideas, obtener este grupo de métricas en algunos casos implica contar con conjuntos de datos estructurados, como códigos de cadena y descriptores de Fourier, por mencionar algunos. Por conjunto de datos estructurados, entendamos datos que pueden ser accedidos de forma rectangular utilizando índices i-j-k [4] y para cada punto (*x, y, z*) el próximo punto estará en uno de los 26 puntos vecinos del espacio 3D (ver Figura 2). Cabe resaltar que contar con datos con estas propiedades, permite utilizar técnicas de cuantificación como cálculo de volúmenes, áreas y longitudes [4].

La organización de este documento es la siguiente: En la sección 2 se muestran algunos trabajos relacionados con la descripción de contornos en superficies 3D y métodos propuestos para la estructuración de los puntos. En la sección 3 se muestra el método propuesto para el remuestreo estructurado de contornos de huecos en superficies 3D. Posteriormente en la sección 4 se muestran los experimentos y resultados obtenidos para validar el funcionamiento del método. Por último, en la sección 5 se plantean las conclusiones y trabajos futuros.

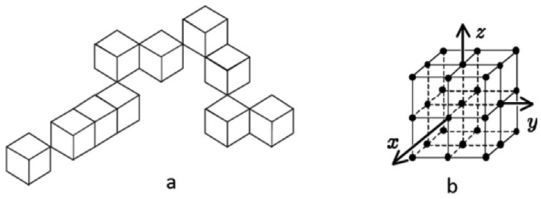

Figura 2. Conjunto de datos estructurados (a), respetando la continuidad de los 26 vecinos cercanos (b).

2. TRABAJOS RELACIONADOS

El proceso de tomar un conjunto de puntos en el espacio 3D, ya sean parte de una curva o de un volumen, y crear modelos estructurados con estos, es llamado en la literatura voxelización o conversión 3D-Scan. En este trabajo no se utiliza el término voxelización, ya que un *Voxel* es una estructura con volumen (*Volume Element*), y el resultado final de nuestro método son puntos, los cuales carecen de volumen desde su definición básica. Sin embargo, el trasfondo de determinar un conjunto de *Voxels* y un conjunto de

puntos estructurados sobre grillas cuadradas es parecido. La aproximación clásica para obtener modelos de *Voxels* es a través de subdivisión espacial [6]. Esta técnica se basa en descomponer el espacio que contiene el modelo 3D en subespacios rectangulares de forma iterativa, lo cual incurre en altos gastos computacionales para grillas de media y alta resolución. Algunos algoritmos para la voxelización de curvas 3D son propuestos por Danielsson [7] y Mokrzycki [8]. Sin embargo, estos se limitan a curvas definidas como la intersección de dos superficies implícitas. Kauffman en [9][10] [11] sienta las bases para las técnicas de discretización de líneas y volúmenes 3D o *Volume Graphics* como el mismo propone en [12], utilizando aproximaciones clásicas de algoritmos de discretización de líneas 2D como el DDA (*Digital Differential Analyzer*) y Bresenham. Por otra parte, propone la discretización de líneas paramétricas como curvas de Bezier Cúbicas. El rápido aumento de la capacidad de computo gráfico (*GPU*) es de interés para aplicaciones tales como la voxelización, debido al alto gasto computacional de la misma. Dong en [13] propone un algoritmo basado en la combinación de técnicas clásicas de voxelización corriendo sobre una tarjeta gráfica. En este trabajo se logran resultados de tiempo real para modelos de más de dos millones de triángulos. Novotny propone en [14] un método de voxelización de modelos sólidos manteniendo detalles pequeños a través de una combinación de técnicas de operadores booleanos o CGS. Por último, la aplicación específica de subdivisión espacial para la evaluación de códigos de cadena de curvas 3D es propuesta por Bribiesca en [15].

Según lo expuesto en este capítulo, la literatura encontrada al respecto es amplia y se encuentra en etapa de mejoramiento de algoritmos que satisfagan requerimientos tales como procesamiento en tiempo real, eficiencia y precisión en modelos masivos. Sin embargo la especificidad y novedad de la aplicación tratada en este trabajo, requiere del desarrollo de nuevos métodos basados en trabajos anteriores, que contemplen ciertas restricciones implícitas en los objetos de forma libre y el proceso de caracterización de formas de la visión por computador.

3. MÉTODO DE REMUESTREO ESTRUCTURADO

El método que se presentará a continuación trabaja con un vector ordenado *v* de puntos con coordenadas euclídeas *x, y, z,* pertenecientes al contorno del hueco en la superficie del objeto de forma libre. Claramente para llegar a este conjunto de datos es necesario identificar el hueco en la superficie 3d, seleccionar y almacenar los datos pertenecientes al contorno del mismo, lo cual no es objetivo de este trabajo. Por conjunto ordenado de puntos, entiéndase que estos deben conformar una secuencia del contorno del hueco, de modo tal que si se dibujan líneas entre los puntos V_i, V_{i+1} se obtiene el contorno cerrado (ver Figura 3).

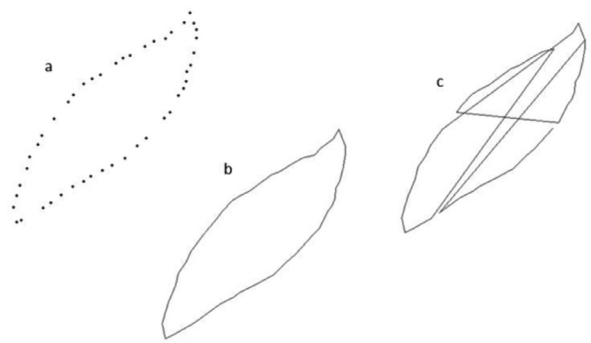

Figura 3. a) puntos del contorno del hueco. b) conjunto ordenado de puntos unidos a través de líneas rectas. c) conjunto no ordenado de puntos unidos a través de líneas rectas.

A continuación se describen cada una de las etapas del método propuesto.

3.1 Ajuste De Puntos Mediante Curva NURBS

Según la definición de objetos de forma libre propuesta por Besl en [16], las superficies de estos son suaves y diferenciables, de lo cual se podría inferir que los bordes de los huecos y su zona circundante deben ser suaves y diferenciables igualmente. Ahora, debido al hecho de que los puntos adquiridos mediante el proceso de escaneado representan muestras discretas del objeto real, la relación entre cada uno de estos es desconocida, e.g. Lineal, parabólica, etc. Lo anterior se resume en el problema de la parametrización de curvas el cual ha sido ampliamente estudiado por la comunidad científica de la computación gráfica. En este trabajo se hizo uso de las curvas NURBS para hallar dicha relación entre puntos, debido a que estas han sido ampliamente adoptadas para los sistemas de modelado computacional debido a su generalidad

según expresa Lengyel en [17]. Por otra parte según Piegl [18] las curvas NURBS se han convertido en el estándar industrial de facto y pueden ser utilizadas para el llenado de huecos en superficies 3D presentando como ventajas generalidad en la solución y control geométrico local de los datos, entre otras. Cabe aclarar que de acuerdo a la aplicación específica en la que se esté utilizando este trabajo, el ajuste de puntos podría requerir del uso de otros métodos como Splines o FBR (*funciones de base radial*), esto debido a que una selección indebida de dicho método de ajuste podría incurrir en una aproximación incorrecta respecto a la curva original. Por ejemplo, es bien sabido que las NURBS no presentan un buen resultado cuando se trata de aproximar curvas de figuras geométricas bien establecidas, i.e. Círculos, triángulos, cuadrados, etc. Para esto, las ventajas y limitaciones de las NURBS son expuestas en . Las NURBS (*Non Uniform Rational B-Splines*) son funciones polinomiales a trozo (*piecewise*) evaluadas vectorialmente según la siguiente expresión matemática [19]:

$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N}{\sum_{i=0}^{n} w_i N_{i,}} \qquad (1)$$

Donde C es el vector final de puntos de la curva, w_i son los pesos, P_i son los puntos de control, en este caso los puntos muestreados del contorno, y $N_{i,p}(u)$ son las funciones base de las B-Spline normalizadas de grado p, definidas recursivamente según:

$$N_{i,0}(u) = \begin{matrix} 1 \Leftrightarrow k_i \leq u \leq k_{i+1} \\ 0 \Leftrightarrow otherwise \end{matrix}$$

$$N_{i,p}(u) = \frac{u \quad k_i}{k_{i+p} \quad k_i} N_{i,p\ 1}(u) + \frac{k_{i+p+1} \quad u}{k_{i+p+1} \quad k_{i+1}} N_{i+1,p\ 1}(u) \qquad (2)$$

Donde $K = \{k_0, k_1, ..., k_m\}$ es el llamado vector de nudos (*knots vector*). Generalmente este vector toma la forma $K = \{\alpha, \alpha, ..., \alpha, k_{p+1}, ..., k_{m\ p\ 1}, ..., \beta, \beta, ..., \beta\}$, en la práctica, comúnmente $\alpha = 0$ y $\beta = 1$ [19].

Teniendo en cuenta lo anterior, los puntos del contorno del hueco son aproximados mediante una curva NURBS utilizando para ello el algoritmo CoxDeBoor [20][21]. En la Figura 4 se muestra el resultado de esta aproximación.

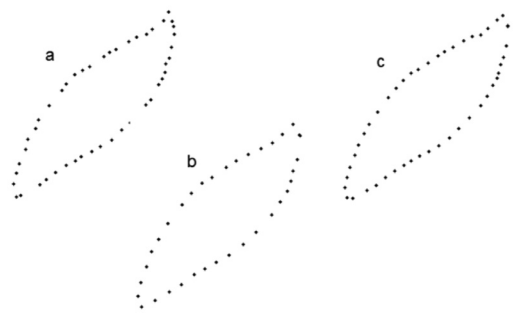

Figura 4. De izquierda a derecha a) Puntos Originales b) Puntos de la curva NURBS 75% c) Puntos de la curva NURBS 100%.

Se puede observar que la suavidad del contorno lograda mediante el ajuste de la curva NURBS es mayor que en los puntos originales. Este tipo de curvas paramétricas permite tanto reducir como aumentar el número de puntos final de la curva. El objetivo de este proceso es obtener un nuevo vector de puntos $V' = \{v'_1, v'_2, ..., v'_n\}$ del cual conocemos la relación entre cada secuencia v'_i, v'_{i+1} y se ajusta de forma más adecuada al comportamiento suave y uniforme de los objetos de forma libre. Determinar el número final de puntos de la curva NURBS dependerá de la aplicación que se esté trabajando.

3.2 Remuestreo Estructurado Mediante El Algoritmo Bresenham 3D

Si bien, una vez obtenemos los nuevos puntos ajustados mediante una curva NURBS, logramos una mayor suavidad y uniformidad en los mismos, estos no se encuentran espaciados de forma estructurada en el espacio 3D, i.e. no son accesibles a través de índices *i,j,k*. Para conseguir lo anterior es necesario un proceso de estructuración. El algoritmo de discretización de rectas propuesto por Jack Bresenham [22] (ver Figura 5), constituye aún hoy en día uno de los estándares de la computación gráfica, debido a que presenta como característica principal que sólo utiliza operaciones enteras, lo cual lo hace eficiente computacionalmente. Para este caso, se utilizó el algoritmo Bresenham 3D propuesto por Kaufman en [9].

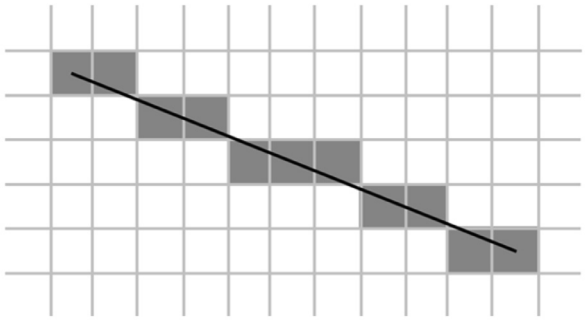

Figura 5. Resultado del algoritmo de Bresenham 2D. Los rectángulos de color gris representan la discretización de Bresenham de la línea de color negro.

Este algoritmo toma dos puntos en el espacio 3D y devuelve el conjunto de puntos que mejor aproxima una línea recta entre estos dos. Este conjunto tiene la propiedad de moverse a intervalos regulares de forma estructurada, de tal forma que para cada punto p_i del nuevo conjunto, p_{i-1} y p_{i+1} pertenecen a los 26 vecinos cercanos. Aunque este algoritmo fue propuesto para trabajar en el espacio de los enteros, debido a que las pantallas de los computadores trabajan de dicha forma, para este caso, el conjunto inicial de puntos del contorno puede moverse ya sea en el espacio de los enteros como de los flotantes, es necesario calcular el tamaño de la grilla que contiene los 26 vecinos cercanos. Un cálculo incorrecto del tamaño de la grilla de los 26 vecinos cercanos, incurriría en solapamiento de puntos del vector y por ende en pérdida de información, por ejemplo, si se tienen dos puntos $p_1 = (0.5,0.5,0.5)$ y $p_2 = (0.7,0.7,0.7)$ y la grilla es de tamaño 1, implica que los dos puntos están contenidos dentro de una misma sección de los 26 vecinos cercanos y por esto ambos se transforman en el mismo punto. Para solucionar este problema, es necesario realizar un cálculo de la resolución en que trabajará el algoritmo de Bresenham.

3.3 Cálculo de la Resolución del Vector V'

Si para cada punto del vector $V' \in R$ se necesita asegurar que el proceso de estructuración de los datos no genere solapamiento entre estos, es claro que se debe cumplir que $\forall (x_i, y_i, z_i) \in v'_i \in V', \min(\max(x_i \quad x_{i+1}, y_i \quad y_{i+1}, z_i \quad z_{i+1})) = r$, donde r es el tamaño de la grilla que utilizará el algoritmo de Bresenham, es decir, el valor mínimo que puede tomar la grilla para nuestro vector, sin que genere solapamiento de puntos, será igual al mínimo de los máximos valores de diferencias entre ejes de puntos consecutivos de nuestro vector. Sin embargo el algoritmo de Bresenham 3D no recibe algún parámetro que indique el tamaño de la grilla, sólo el número de decimales con que trabajara. Una solución sencilla a este problema radica en suponer que Bresenham trabajara sobre grillas de valor entero. En otras palabras la solución radica en hallar un $k \mid kV'$ que satisfaga la condición:

$$\forall (x_i, y_i, z_i) \in v'_i \in V', \min(\max(x_i \quad x_{i+1}, y_i \quad y_{i+1}, z_i \quad z_{i+1})) = 1$$

El siguiente algoritmo determina la constante de resolución:

```
para i = 1:x-1

    dx = abs(VP(i,1) - VP(i+1,1));

    dy = abs(VP(i,2) - VP(i+1,2));

    dz = abs(VP(i,3) - VP(i+1,3));

    VR(i)= max([dx dy dz]);

fpara

r = min(VR);

k = 1/r;
```

Cabe aclarar que la constante hallada mediante el algoritmo anterior asegura el mínimo espaciamiento entre puntos para que trabaje el algoritmo de Bresenham sin generar solapamiento de puntos, sin embargo, modificando el algoritmo 2 se pueden obtener resoluciones que brinden mayor detalle a los puntos finales. Una vez se obtiene el nuevo vector V'' escalado, producto de multiplicarlo por la constante determinada anteriormente, se procede a estructurarlo de forma regular mediante Bresenham 3D, el cual es calculado para cada secuencia de puntos $v''_i, v''_{i+1} \in V''$. El resultado de esta operación se puede observar en la Figura 6.

Figura 6. Nuevos puntos resultado de aplicar Bresenham 3D sobre los puntos del contorno.

Por último, debido a que el proceso anterior modifica el tamaño original de los datos, es necesario re-escalar el vector estructurado, mediante $\frac{V''}{k}$.

4. Experimentos y resultados

Para los experimentos se utilizaron conjuntos de puntos de contornos de huecos de objetos de forma libre procedente de diferentes fuentes, tanto de modelos libres asequibles mediante la web y otros escaneados específicamente para este trabajo. Con el fin de probar el funcionamiento del método propuesto, se presentan algunos ejemplos de puntos de contornos de huecos de objetos de forma libre y su respectiva estructuración. En las Figuras 7, 8, 9, se pueden observar el modelo 3D (a) en el cual se indica el hueco seleccionado, el conjunto inicial de puntos extraídos del contorno (b) y el conjunto de puntos estructurados mediante el método aquí propuesto (c).

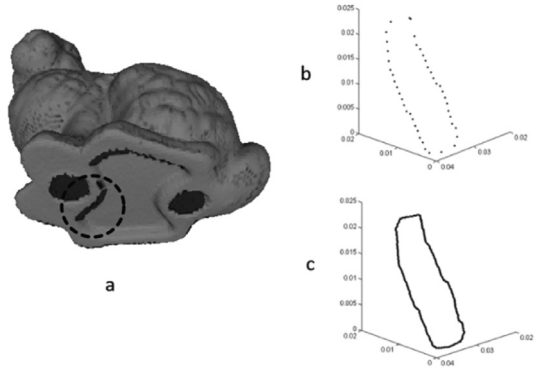

Figura 7. Standford Bunny.

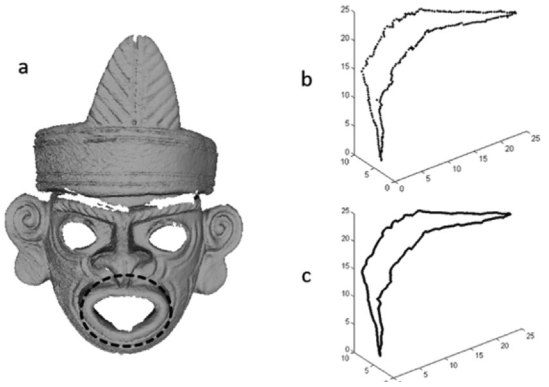

Figura 8. Máscara Indígena.

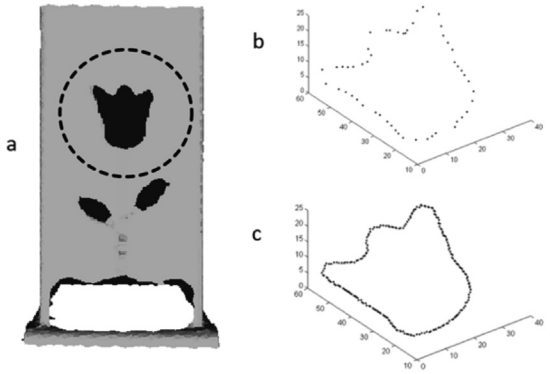

Figura 9. Objeto de adorno para el hogar.

Cabe aclarar que los objetos tratados para los experimentos tienen un procesamiento previo, específicamente, el proceso de registro de vistas parciales y en algunos casos, suavizado y reducción del número de puntos, procesos necesarios en cualquier proceso de reconstrucción 3D.

5. CONCLUSIONES

En este trabajo se ha presentado un método para estructurar puntos de contornos de huecos en objetos de forma libre. Debido a la naturaleza de las NURBS y de igual forma del algoritmo de Bresenham, el método propuesto es invariante ante *traslación* y *rotación*. Por otra parte, según lo observado en los experimentos, el método preserva la forma del contorno tanto en grandes como en pequeños detalles. El aumento en el número de puntos total del contorno es necesario con el fin de preservar de manera precisa la forma del contorno, sin embargo, el número de puntos obtenidos de la NURBS (*sección 3.1*) y el cálculo automático de la resolución

(*sección 3.3*), pueden ser modificados para interferir en el número de puntos del contorno estructurado. Lo anterior dependerá de la aplicación en la cual se esté utilizando el método, es decir, si se requiere precisión en la forma o simplificación de datos. Como trabajo futuro, se propone extender el funcionamiento del algoritmo para operar no sólo con contornos cerrados sino con curvas con más de un grado de libertad como la que se puede observar en la Figura 10. Por otra parte, experimentar el método propuesto con diferentes aproximaciones de ajuste de curvas sobre diversos modelos, podría brindar mayor claridad a la hora de elegir el tipo adecuado de ajuste de curvas de acuerdo a la aplicación que se vaya a trabajar.

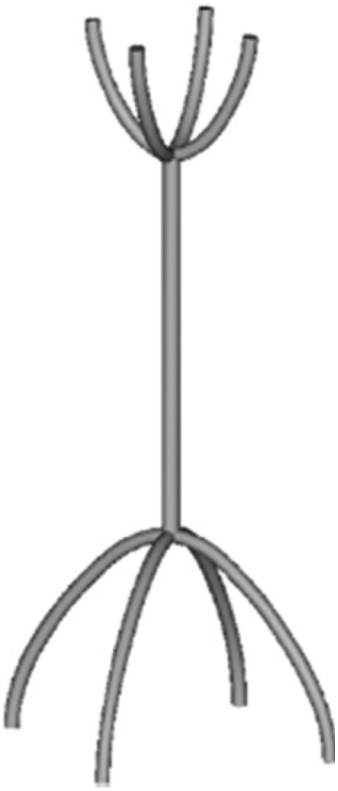

Figura 10. Curva 3D con 5 grados de libertad.

REFERENCIAS

[1] Sanchez, G., Branch, J., and Atencio, P., A Metric for Automatic Hole Characterization, in Proceedings, 19th International Meshing Roundtable, Springer-Verlag, pp.195-208, 2010.

[2] Branch J., Reconstrucción de Objetos de Forma Libre a Partir de Imágenes de Rango Empleando una Red NURBS [PhD Tesis]. Medellin, Universidad Nacional de Colombia. 2006.

[3] Campbell, R. J. and Patrick, F. J., A Survey of Free-Form Object Representation and Recognition Techniques, Computer Vision and Image Understanding, vol. 81, pp. 166-210, 2001.

[4] Dimitrov, L. I. and Sramek, M., Using 3D-Bresenham for resampling structured grids, 2nd International Symposium on 3DPVT, no. 3, pp. 926-930, 2004.

[5] Toriwaki, J. and Yoshida, H., Fundamentals of Three-dimensional Digital Image Processing. Springer, London, 2009.

[6] Lee, Y. T. and Requicha, A., Algorithms for computing the volume and other integral properties of solids. II. A family of algorithms based on representation conversion and cellular approximation, Communications of the ACM, vol. 25, no. 9, pp. 642-650, 1982.

[7] Danielson, P. E., Incremental Curve Generation, IEEE Transactions on Computers, vol. 19, (9), pp. 783 - 793, 1970.

[8] Mokrzycki, W., Algorithms of Discretization of Algebraic Spatial Curves on Homogeneous Cubical Grids, Computers & Graphics, vol. 12, (3), pp. 477-487, 1988.

[9] Kaufman, A. and Shimony, E., 3D Scan-Conversion Algorithms for Voxel-Based Graphics, in I3D '86 Proceedings of the 1986 workshop on Interactive 3D graphics, 1987.

[10] Kaufman, A. and Cohen, D., 3D Line Voxelization and Connectivity Control, IEEE Computer Graphics and Applications, vol. 17, (6), pp. 80-87, 1999.

[11] Kaufman, A., Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes, SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques, vol. 21, (4), pp. 171-179, 1987.

[12] Kaufman, A., Cohen, D., and Yagel, R., Volume Graphics, IEEE Computer Society Press, vol. 26, (7), pp. 51-64, 1993.

[13] Dong, Z., Chen, W., Bao, H., Zhang, H., and Peng, Q., Real-time Voxelization for Complex Polygonal Models, Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04), pp.43-54, 2004.

[14] Novotny, P., Dimitrov, L., and Sramek, M., Enhanced Voxelization and Representation of Objects with Sharp Details in Truncated Distance Fields, IEEE Transactions On Visualization And Computer Graphics, vol. 16, no. 3, pp. 484-498, 2010.

[15] Bribiesca, E., A method for representing 3D tree objects using chain coding, Journal of Visual Communication & Image Representation, vol. 19, pp. 184 - 198, 2008.

[16] Besl, P. J., The Free-Form Surface Matching Problem In Machine Vision Three Dimensional Scenes. Machine Vision for Three-Dimensional Scenes, Academic Press, pp.25-71, 1990.

[17] Lengyel, E., Mathematics for 3D game programming and computer graphics, 2nd ed. Charles River Media, Massachusetts, 2004.

[18] Piegl, L. and Tiller, W., Filling n-sided Regions with NURBS patches, The Visual Computer, vol. 15, (2), pp. 77-89, 1999.

[19] Piegl, L., On NURBS: A Survey, IEEE Computer Graphics and Applications, vol. 11, (1), pp. 55-71, 1991.

[20] De Boor, C., On Calculating With B-Splines," Jornal of Approximation Theory, vol. 6, (1), pp. 50-62, 1972.

[21] Cox, M. G., The Numerical Evaluation of B-Splines, Journal of the Institute of Mathematics and its Applications, vol. 10, (2), pp. 134-179, 1972.

[22] Bresenham, J., Algorithm for computer control of a digital plotter, IBM SYSTEMS JOURNAL, vol. 4, (1), pp. 25-30, 1965.