Seventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2009)
"Energy and Technology for the Americas: Education, Innovation, Technology and Practice"
June 2-5, 2009, San Cristóbal, Venezuela.

Recontrucción tridimensional de objetos a partir de una imagen digital bidimensional

Domingo Hernández
Universidad de Los Andes, Mérida, Venezuela, dhh@ula.ve

Mario A. Bislck
Universidad de Los Andes, Mérida, Venezuela, mario@ula.ve

Leira Chacón
Universidad de Los Andes, Mérida, Venezuela, leirac@ula.ve

RESUMEN

Este artículo describe el diseño, desarrollo e implantación de un sistema de software que permite realizar actividades básicas de reconocimiento y reconstrucción tridimensional de objetos a partir de una imagen digital. El sistema utiliza algoritmos de filtrado de imágenes digitales, tales como el operador Sobel y filtros gaussianos, junto con un algoritmo de segmentación de imágenes digitales basado en crecimiento de regiones, para detectar automáticamente el contorno de un objeto partiendo de un punto inicial de muestra. Además, utiliza los contornos activos para reconocer partes más específicas del objeto. Finalmente, genera una nube de puntos que son triangulizados, mediante un algoritmo de triangulación de Delaunay, para generar un estimado de la superficie tridimensional del objeto reconocido.

Palabras claves: Reconstrucción tridimensional, segmentación de imágenes, contornos activos, triangulación de Delaunay

ABSTRACT

This article describes the design, development and implementation of a software system that allows basic recognition and three-dimensional reconstruction of objects from a digital image. The system uses several filtering algorithms of digital images, such as Sobel operator and gaussian filter with an algorithm for segmentation of digital images based on growth areas, to automatically detect an object from an initial sample. In addition, it uses active contours to recognize specific parts of the object. Finally, it generates a cloud of points that are triangulizados through a Delaunay triangulation algorithm to generate an estimate of the visible surface of the object recognition.

Keywords: Three-dimensional reconstruction, image segmentation, active contour, Delaunay triangulation.

1. INTRODUCCIÓN

Este trabajo presenta un sistema programado que permite la reconstrucción tridimensional de la superficie visible de un objeto que a partir de una imagen digital bidimensional, realizando actividades de procesamiento y segmentación de imágenes digitales, así como de reconstrucción de escenas tridimensionales.

Actualmente, varios sistemas de software permiten reconstruir tridimensionalmente un objeto, utilizando un conjunto de imágenes digitales (Francois and Medioni, 1999), (Internet 1).

Algunos de estos sistemas logran generar un modelo tridimensional utilizando algoritmos de inteligencia artificial para inferir la profundidad de los puntos que conforman al objeto (Francois and Medioni, 1999). Otros se apoyan en información de alto nivel solicitada al usuario que pueda ser luego utilizada para obtener mejores resultados, ya

sea debido a una mejor precisión en el modelo generado, debido a un menor tiempo de cómputo, o ambos (Internet 1).

Además, en algunos casos también resulta posible el generar un modelo completo del objeto al procesar un conjunto de imágenes que representan al mismo objeto observado desde diversos ángulos (Saxena, et al, 2008). Sin embargo, el sistema expuesto en este trabajo se considera como una herramienta inicial que puede ser posteriormente mejorada, tanto en tiempo de cálculo como en precisión del modelo tridimensional generado, de manera tal que pueda luego derivar en sistemas más especializados. Por esta razón, este sistema permite generar sólo una estimación de la superficie de un objeto detectado en una imagen digital que recibe como entrada. Para esto se utiliza información de alto nivel pedida al usuario mediante una interfaz gráfica sencilla. La cantidad de esta información puede variar, en el caso mínimo el usuario sólo marca un punto indicando el lugar donde está el objeto y automáticamente el sistema estima su contorno mediante en un algoritmo de segmentación de imágenes basado en crecimiento de regiones.

En el resto de los casos, la información consiste en un polígono regular o irregular dibujado por el usuario cerca del contorno del objeto que le interesa. Este polígono inicial representa un contorno activo o *snake* (Kass, et al 1987), (Valverde, 2004), que puede ser deformado automáticamente para adaptarse a la forma del contorno. Una vez obtenido el contorno del objeto se procede a generar una nube de puntos internos al mismo. Estos puntos son luego triangulizados mediante un algoritmo de triangulación de Delaunay generando un mallado plano del objeto. Finalmente, para cada punto se procede a estimar su profundidad mediante un algoritmo que utiliza la intensidad del pixel correspondiente en la imagen y la asocia con una posición en el eje coordenado Z.

2. DESARROLLO DEL SISTEMA DE SOFTWARE

Para el desarrollo del sistema de software se utilizó el modelo de Espiral de Boehm, (Boehm, 1988). Este modelo divide el proceso de desarrollo de un sistema de software en 4 etapas: planificación, análisis de riesgo, ingeniería y evaluación. Además, al ser un modelo iterativo y evolutivo, permitió desarrollar el sistema de manera modular, es decir, un módulo a la vez. La principal ventaja de este modelo, es que permite evaluar la posibilidad de implantación de un algoritmo o un método antes de invertir tiempo en el diseño. Otra ventaja es que considera analizar los riesgos de utilizar un algoritmo, un lenguaje de programación, o un enfoque específico de desarrollo, permitiendo comparar las ventajas y desventajas de las distintas opciones para tomar una decisión antes de comenzar con el desarrollo del sistema. En total, se realizaron cinco iteraciones para obtener el sistema de software expuesto en este trabajo. En la primera iteración, durante la etapa de análisis de riesgo se decidió que el sistema podía ser desarrollado utilizando Qt (Internet 2), para la implantación de la interfaz gráfica y manipulación de imágenes digitales, y OpenGL (Internet 3), para la reconstrucción tridimensional de la superficie del objeto.

Ambos, Qt y OpenGL, al ser herramientas multiplataforma contribuyen con la portabilidad del sistema, permitiendo desarrollar a éste en cualquier sistema operativo, siempre y cuando no se utilicen instrucciones especificas de alguno de ellos, de manera tal que luego pueda ser ejecutado en cualquiera de ellos una vez el código haya sido compilado adecuadamente en el sistema operativo correspondiente. Esta etapa también permitió tomar la decisión de utilizar un paradigma de programación orientado por objetos, por lo que se consideró utilizar Técnicas de Desarrollo de Sistemas de Objetos (TDSO) (Besembel, 1998) junto con el Lenguaje Unificado de Modelado (UML por sus siglas en ingles) (Fowler 1999), (Internet 4), como herramientas para el diseño y documentación del sistema. Por su parte, durante la etapa de ingeniería se realizó el diseño y la implantación de: una interfaz de usuario básica con capacidad de mostrar imágenes digitales, una clase base polígono y una primera aproximación al algoritmo de triangulación de Delaunay.

Posteriormente, en la segunda iteración durante la etapa de ingeniería se realizó el diseño e implantación del algoritmo del operador Sobel, (Sobel, and Feldman, 1968) para detección de bordes, y un algoritmo de segmentación de imágenes utilizando contornos activos o *snakes* y sus distintas variantes propuestas en (Menet, et al, 1990), (McInerney, and Terzopoulos, 1995), (Zagorchev, et al, 2007). En esta etapa se desarrolló un algoritmo de detección de bordes mediante la comparación de los pixeles ubicados inmediatamente alrededor de un píxel central, formando lo que se conoce como una ventana cuadrada de 3 pixeles de longitud y ancho. Este

algoritmo requiere una menor cantidad de cálculo que el del operador Sobel y obtiene resultados parecidos sin necesidad de realizar convolución entre dos matrices. En la tercera iteración, además de la planificación de las actividades y del análisis de riesgo, en la etapa de ingeniería se diseñó e implantó un algoritmo de segmentación de imágenes digitales basado en crecimiento de regiones, así como varios filtros de imágenes digitales tales como el filtro Gaussiano, el filtro Laplaciano , y el filtro Laplaciano del Gaussiano (LoG por sus siglas en ingles). Además, en esta iteración también se modifico la interfaz del sistema para permitir que se pudieran abrir más de una imagen al mismo tiempo. También, debido a que los resultados obtenidos no fueron satisfactorios, se realizó un refinamiento de los algoritmos de segmentación diseñados en la iteración anterior.

Posteriormente, en la etapa de ingeniería de la 4ta iteración, se refinaron los algoritmos de deformación del *snake* y se diseñó e implantó un algoritmo basado en el *scanlLine* para generar un polígono que representa el contorno del objeto a reconstruir. Debido a que este algoritmo dependiendo del caso puede no proporcionar un contorno aproximado del objeto, se implementaron dos algoritmos que utilizan el mismo principio del *scanLine*, pero funcionan a nivel local y en un sentido específico, ya sea ascendente o descendente a partir de una posición inicial. En esta iteración también se diseñaron e implantaron los algoritmos y clases necesarias para la visualización tridimensional de la superficie del objeto reconstruida. Entre los algoritmos se encuentran los de desplazamiento y rotación de una cámara en un espacio tridimensional, así como los de dibujado de polígonos en el espacio. Por último, en la quinta Iteración se diseñó e implantó un algoritmo de estimación de profundidad basado en la intensidad del color del *píxel* de la imagen de entrada que está asociado a un punto de la superficie tridimensional generada. Además, se refinaron los algoritmos de reconstrucción tridimensional permitiendo asignar colores de la imagen de entrada a cada punto de la superficie generada, así como se optimizaron la mayoría de los códigos de los algoritmos mencionados anteriormente. Es importante mencionar que, para cada iteración, además de las actividades realizadas durante las etapas de ingeniería mencionadas anteriormente, también se realizaron actividades de planificación, de análisis de riesgo y de prueba en las etapas correspondientes. En ésta última, las pruebas realizadas abarcaron clases, algoritmos, funcionamiento de la interfaz, etc. Además, la interfaz del programa siempre se fue refinando según las necesidades y requerimientos que se contemplaban para cada iteración.

3. DESCRIPCIÓN DEL PROCESO DE OBTENCIÓN DEL MODELO TRIDIMENSIONAL

El proceso de generar el estimado de la superficie del objeto que se quiere reconstruir a partir de una imagen digital no es un proceso directo, por el contrario éste se divide en tres partes. La primera corresponde a la preparación de la imagen digital para que al aplicar los distintos algoritmos de segmentación se obtenga un mejor resultado. Esta primera parte es conocida como pre-procesamiento de una imagen digital.

La segunda parte corresponde a la segmentación de la imagen. En esta etapa se aplican distintos algoritmos para poder identificar la parte de la imagen que le interesa al usuario, la cual corresponde al objeto que se quiere reconocer. Finalmente, la tercera parte del proceso corresponde a la reconstrucción tridimensional del objeto. En esta etapa se genera el mallado que representa la superficie visible del objeto que aparece en la imagen y se dibuja en un espacio completamente tridimensional.

PRE-PROCESAMIENTO DE LA IMAGEN DIGITAL DE ENTRADA

Debido a que los algoritmos de detección de bordes son muy sensibles al ruido que pueda haber en una imagen digital, es necesario utilizar filtros que permitan disminuir la presencia de estos en la misma. Con esta finalidad, el sistema utiliza un filtro Gaussiano, el cual se basa en la distribución del mismo nombre (también conocida como distribución Normal) para generar una matriz de convolución que es luego aplicada a la imagen para eliminar el ruido que ésta posea. El sistema utiliza por defecto una desviación estándar relativamente alta de 4.0 y una matriz Gaussiana de tamaño 5x5. Estos valores pueden luego ser modificados por medio de la interfaz del sistema. Además de la reducción de ruido en la imagen, el sistema utiliza 3 filtros de detección de borde. Estos filtros pueden ser utilizados después de que se ha reducido el ruido en la imagen, e incluso el sistema permite que una vez aplicados los filtros de detección de bordes se le pueda aplicar un filtro Gaussiano a la imagen resultante. Cada uno de estos filtros genera imágenes que contienen únicamente pixeles de color blanco (máxima intensidad)

o negro (mínima intensidad). El primero de estos filtros corresponde a uno desarrollado durante la segunda iteración. Este filtro recorre todos los pixeles de la imagen, y por cada uno de revisa si para alguno de los pixeles inmediatamente alrededor de éste (incluyendo los ubicados en las diagonales) el valor en escala de gris del "píxel" actual es mayor o igual que el valor del píxel vecino más un rango. De esta manera, el filtro recorre la imagen con una ventana de dimensión 3x3 cuyo centro es el píxel actual. El valor del rango utilizando en el sistema es de 20.

El único caso especial que presenta este filtro corresponde a los pixeles ubicados en los cuatro bordes que presenta la imagen. Para este caso, se verifica la misma condición pero sin considerar las casillas de la ventana que caigan fuera de la imagen.

El segundo de los filtros utilizados por el sistema es el del operador Sobel. Este filtro utiliza dos matrices de convolución para calcular el gradiente de la imagen en cada uno de sus pixeles. Luego revisa si el valor del gradiente es mayor que un valor mínimo, en caso de que así sea marca ese píxel como negro; en caso contrario lo marca como blanco. El valor mínimo utilizado por el sistema es 40, mientras que las matrices son las mostradas en (1)

$$M_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} ; \; M_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (1)$$

Otro filtro de detección de bordes utilizado es el filtro Laplaciano, el cual acentúa las regiones de la imagen donde haya un cambio brusco de intensidad. Este filtro se implantó mediante convolución utilizando la matriz 3x3 mostrada en (2).

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (2)$$

Debido a que el filtro Laplaciano es muy sensible al ruido, el sistema utiliza el filtro conocido como el Laplaciano del Gaussiano. Este filtro aplica primero el filtro Gaussiano a la imagen de entrada y a la imagen resultante aplica luego el filtro Laplaciano, obteniendo un mejor resultado.

SEGMENTACIÓN DE LA IMAGEN DE ENTRADA

Para esta parte del proceso, el sistema utiliza la información obtenida después de haber aplicado alguno de los filtros de detección de bordes mencionados anteriormente.

Considerando el proceso de segmentación de dos fases propuesto por Allevato (Allevato, 2006), éste sistema permite segmentar la imagen de 2 formas: automática o manual. La primera utiliza un algoritmo de segmentación de imágenes digitales basado en crecimiento de regiones. Este algoritmo permite detectar la zona alrededor de un punto inicial que está conformada por pixeles cuyos colores se encuentran dentro de un rango, superior e inferior, con respecto al color que posee el punto inicial. Si todos los puntos dentro de ésta área son vistos como un grafo, donde los nodos de éste son los pixeles del área, y donde cada nodo asociado a un píxel está conectado con los nodos de los pixeles vecinos dentro del área, entonces se tendría un grafo conexo. La implantación de este algoritmo en el sistema arroja como resultados una máscara de la imagen que representa el área de ésta que fue segmentada, y una matriz bidimensional que indica el contorno del área segmentada, así como los puntos que pertenecen a la segmentación y los que no. La máscara es una imagen que lo único que tiene dibujado de color blanco es el área segmentada, mientras que el resto es de color negro. La información que provee la matriz es

utilizada por un algoritmo basado en el "*scanLine*" para obtener un polígono que representa una aproximación del contorno del área segmentada. Este algoritmo consiste en recorrer verticalmente la matriz anterior, desde la fila inferior hasta la superior; donde por cada fila el algoritmo se desplaza horizontalmente encontrando los pixeles ubicados más a la izquierda y más a la derecha de la misma. Esto lo repite hasta que alcance la última fila, o hasta que consiga una fila donde no haya puntos que pertenecen a la segmentación. Los puntos encontrados los agrega a una lista de puntos que representan al polígono, colocando los puntos más a la izquierda al inicio de la misma, mientras que los puntos más a la derecha los agrega al final de lista. Posteriormente, esta lista es simplificada mediante un algoritmo que, partiendo del primer punto en la lista, agrega a una lista auxiliar todos aquellos puntos que se estén a una distancia mínima de 10 unidades con respecto a cualquier punto que se encuentre en ella. De esta manera se simplifica la cantidad de puntos con las que tiene que trabajar el sistema, sin afectar de forma considerable el resultado obtenido. Finalmente, esta lista de puntos es convertida en un *snake*, el cual puede ser deformado y modificado mediante la interfaz del sistema.

Por otro lado, la forma de segmentación manual que permite el sistema consiste en generar un *snake* a partir de una serie de puntos ordenados que son introducidos por el usuario. El *snake* generado a partir de estos puntos puede, al igual que el caso anterior, ser deformado y modificado para obtener un contorno aproximado del objeto que se quiere reconstruir.

La energía del *snake* se calcula mediante la ecuación 3. Esta es utilizada para la deformación del *snake* y considera únicamente la energía interna y la energía de la imagen en cada punto del *snake*. En este sistema la energía externa no es considerada.

$$E_{snake}(v_i) = \propto * E_{interna}(v_i) + \gamma * E_{imagen}(v_i) \quad (3)$$

RECONSTRUCCIÓN DEL MODELO TRIDIMENSIONAL

En esta parte del proceso se utiliza la información generada en la etapa anterior. Una vez obtenido el polígono inicial se procede a generar una nube de puntos internos al polígono. Esta nube de puntos es generada mediante un algoritmo que recorre el área contenida por el polígono y va agregando puntos a una lista dejando una determinada separación entre ellos. La información del área contenida por el polígono se obtiene directamente de la máscara obtenida anteriormente. De este modo, dependiendo de cómo se obtuvo el polígono, se puede no considerar puntos internos a éste debido ya que éstos están asociados a un píxel de color negro en la máscara. Una vez obtenida la nube de puntos se procede a generar el mallado. Para esto se calculan dos triángulos iniciales que corresponden a la triangulación del rectángulo abarcador del polígono inicial. Posteriormente, se comienzan a agregar uno a uno los puntos del polígono inicial para luego continuar con los que pertenecen a la nube de puntos. De esta manera, cada vez que se agrega un nuevo punto se aplica un algoritmo de triangulación de Delaunay. Al finalizar se eliminan aquellos triángulos cuyas aristas no pertenecen al área segmentada y se obtiene el mallado definitivo. Con la malla calculada, se procede a estimar la profundidad de cada punto que la conforma. Para esto se utiliza un algoritmo que le asigna un valor en la coordenada z a cada punto dependiendo de la intensidad del píxel asociado a esa posición dentro de la imagen. Los puntos de intensidad mínima tienen un valor de cero en la coordenada Z y aquellos de intensidad máxima tienen un valor de 255, mientras que los que tienen un valor de intensidad intermedio tendrán una coordenada Z entre [0; 255] dependiendo del extremo de la escala que tengan más cercano. Por último, al mismo tiempo que a cada punto se le asigna un valor en la coordenada Z, también se les asigna el valor del color que está asociada a la posición (x, y) en la imagen. Esta información es utilizada cuando se dibuja la malla en el entorno tridimensional, simplificando la cantidad de información que el sistema necesita calcular en esa etapa.

4. SISTEMA PROGRAMADO

El sistema posee una interfaz de usuario multi-documento, permitiendo abrir más de una imagen al mismo tiempo. Como se pueden apreciar en las figuras 3 y 4, cada imagen es tratada de manera independiente de las demás, teniendo cada una su propia sub-ventana y sus propios *snakes*. De igual manera, cada imagen tiene una ventana donde se dibujan los modelos tridimensionales de la superficie reconstruida del objeto, pudiendo tener tantas de estas ventanas como imágenes se encuentren abiertas en el sistema. Además, el sistema permite realizar funciones básicas como crear un nuevo proyecto, abrir y guardar un proyecto existente, eliminar del proyecto algunas de las imágenes abiertas, así como dibujar, seleccionar, deformar y eliminar *snakes*, entre otras.

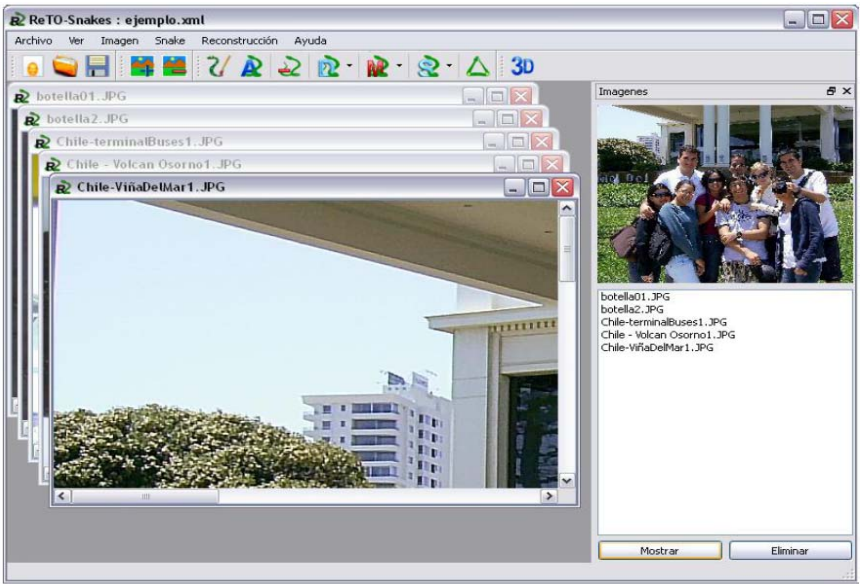

Figure 1: Interfaz principal del sistema

El sistema permite la generación automática de un *snake* o la generación de la malla de la superficie figura 2, de manera que se pueda iniciar el proceso de reconstrucción tridimensional del objeto cuya imagen se esta procesando figura 3. Por otro lado, el sistema permite modificar mediante una ventana de opciones los valores de los parámetros utilizados en varias operaciones, así como seleccionar los filtros que quiere usar. La mayoría de estas operaciones pueden ser accedidas mediante menús, barras de herramientas o diversas combinaciones de teclas del teclado.

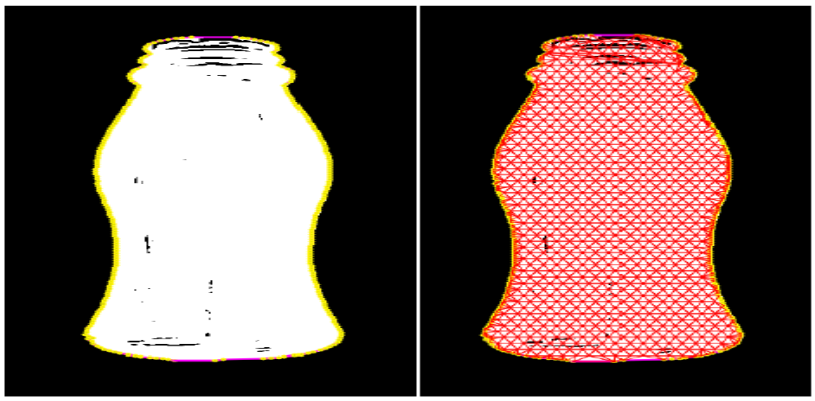

Figure 2: Generación del snake inicial y la malla de superficie

Figure 3: Reconstrucción tridimensional

5. CONCLUSIONES

El sistema propuesto en este trabajo representa la superficie de un objeto que aparece en una imagen digital bidimensional. Estas superficies generadas dependen en gran medida de la iluminación del objeto en la imagen, así como de la calidad de la misma. Los principales problemas que presentan los resultados obtenidos se deben a los productos generados al aplicar los algoritmos de segmentación automática y a los algoritmos de estimación de profundidad. El sistema genera reconstrucciones de superficies 3D con excelente precisión cuando el objeto posee una iluminación frontal y sus bordes están bien definidos, tal y como se pueden apreciar en las figuras Figura 5 y 6. Además, logra hacerlo en un tiempo relativamente corto, sin llegar a ser en tiempo real, a medida que las dimensiones de la imagen sean menores de los 1024 pixeles de ancho y de largo. Para imágenes más grandes el sistema puede requerir un tiempo de cálculo muy considerable.

REFERENCIAS

Allevato, Emiliano, **"**Segmentación de imágenes digitales 3D basado en regiones y contornos activos para la generación de mallas de superficie**."**Santa Fe, Argentina: Mecánica Computacional, 2006, Vol. XXV.

Besembel, Isabel, Técnica de Desarrollo de Sistemas de Objetos (TDSO). Mérida, Venezuela : Publicaciones de la Facultad de Ingeniería-ULA, 1998.

Boehm, Barry W**. "**A Spiral Model of Software Development and Enhancement", Los Alamitos, CA, USA : IEEE Computer Society Press, 1988, Vol. 21. 0018-9162.

Fowler, Martin UML gota a gota Addison Wesley México 1999.

Francois, A. & Medioni, G., "A human-assisted system to build 3-d models from a single image", Image and Vision Computing 1999.

Kass, M., Witkin, A. y Terzopoulos, D., Snakes: Active contour models, Proc. International Conference on Computer Vision, 1987.

Menet, S., Saint-Marc, P. & Medioni, G. (1990), 'B-snakes : Implementation and application to stereo', Image Understanding Workshop. pp. 720–726.

McInerney, T. & Terzopoulos, D. (1995), Topologically adaptable snakes, in 'Proceedings of the Fifth International Conference on Computer Vision (ICCV '95)', Cambridge, USA, pp. 840–845.

OpenGL® The Industry's Foundation for High Performance Graphics. [Internet 3] SGI. [Citado el: 8 de Agosto de 2008.] http://www.opengl.org/documentation/current_version/

Photo-To-3D.com. [Internet 1] [Citado el: 2 de 8 de 2008.] http://www.photo-to-3d.com/

Saxena, Ashutosh, Sun, Min y Ng, Andrew Y**. "**Make3D: Depth Perception from a Single Still Image" AAAI (Nectar track), 2008.

Sobel, I. y Feldman, G, **"***A 3x3 Isotropic Gradient Operator for Image Processing"*, Stanford Artificial Project, 1968.

Trolltech®, a Nokia company**.** Qt Cross-Platform Application Framework. [Internet 2] [Citado el: 8 de Agosto de 2008.] http://trolltech.com/products/qt/

UML® Resource Page. [Internet 4] Object Management Group, Inc. [Citado el: 08 de 08 de 2008.] http://www.uml.org/

Valverde, L. C. (2004), Modelos de contornos activos (snakes), Proyecto de Grado EISULA TA1632 V35, Universidad de Los Andes, Escuela de Insgeniería de Sistemas.

Zagorchev, L., Goshtasby, A. & Satter, M. (2007), 'R-snakes', Image and Vision Computing

25, 945–959.

Autorización y Renuncia

Los autores authorizan a LACCEI para publicar el escrito en los procedimientos de la conferencia. LACCEI o los editors no son responsables ni por el contenido ni por las implicaciones de lo que esta expresado en el escrito

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.