UNIVERSIDAD DE GUANAJUATO

DIVISIÓN DE CIENCIAS E INGENIERÍAS

CAMPUS LEON

NOMBRE DE LA TESIS:

"PROCESAMIENTO DE IMÁGENES DIGITALES: 2D A 3D APLICADO A TUMORES CANCERIGENOS"

TESIS PROFESIONAL

PARA OBTENER EL GRADO DE:

MAESTRIA EN FISICA

PRESENTADA POR:

JOSÉ ZACARIAS HUAMANI LUNA

ASESOR:

DR. TEODORO CORDOVA FRAGA

LEON-GUANAJUATO

2020

Al Área de Física Médica.

A mi asesor Teodoro Córdova Fraga

AGRADECIMIENTOS

Esta obra está dedicada al área de “Física Medica” a cargo de mi asesor el Dr. Teodoro Córdova Fraga por su paciencia y dedicación desinteresada.

Agradecer a mi adorada Sol por su compañía desinteresada y por ayudarme con la programación avanzada.

A Yuli por su compañía y distracción ofrecida en México.

A Tefy por ayudarnos en la redacción de este trabajo.

A Cesi por reunir todos los artículos que nos sirvieron en la realización de este trabajo.

Agradecer a mi gran amigo Meyker al cual le debo un agradecimiento especial por su amistad, apoyo y desinteresada colaboración en la realización del presente trabajo; ya que sin su aliento y abnegación no se habrían alcanzado los objetivos propuestos.
Es mi deseo sincero haber dejado a través de éste trabajo, un camino abierto para todos aquellos amantes de la investigación; recordando siempre que todo esfuerzo requiere de grandes sacrificios pero que a la larga los frutos obtenidos son gratificantes.

A todos ellos muchas gracias☺

DEJO ESTA OBRA A LA DIVISIÓN DE CIENCIAS E INGENIERIAS,CAMPUS LEÓN UNIVERSIDAD DE GUANAJUATO

"COMO EL INICIO DE MI MAS GRANDE LEGADO"

OSSU

ME DA MIEDO FRACASAR; ME DA MIEDO DAR UN PASO Y CAER; LE TENGO MIEDO AL DOLOR DEL FRACASO Y LA CAIDA

PERO MAS MIEDO ME DA VIVIR CON LA ANGUSTIA DE LO QUE PUDO Y NO FUE

EN EL SILENCIO ESCUCHO LA VOZ DE MI CORAZÓN

EL SABE

"ALE A BUDA"

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este trabajo primero recopilamos información de varios paper con el objetivo de proyectar imágenes en 3D a partir de 2D descubrimos una técnica llamada “Marching Cube” que utiliza nueva unidad de medida de intensidad llamada JET junto con un conjunto de técnicas de programación para lograr dicho objetico copiamos diferentes diagramas de flujos y los unimos en un solo programa, con distintas ecuaciones, utilizamos como herramienta principalmente en el programa MATLAB ya que todas las imágenes computarizadas están escritas a través de diferentes matrices llamados bit que expresados en distintos canales RBG (red, blue, green) representan el pixel que es la célula de la imagen que nos sirvió de mucho en la proyección de imágenes medicas en 3D a partir de 2D

Conseguimos leer los pixeles de color rojo; en las imágenes atravez de una camara esto con la finalidad de ver el aumento de temperatura en diferentes imágenes y la dilatación de la piel al sufrir aumento de temperatura

Como una gran victoria personal es que logramos proyectar un tumor cerebral en 3D a partir de una imagen en 2D que nos fue entregada por parte del IMSS T1 (Instituto Mexicano del Seguro Social) con un gran margen de error debido a que los tumores tienen formación muy antisimétricas pero logramos leer sus intensidades en toda la imagen .

Por ultimo descubrimos que necesitamos mas recurso computacional para poder procesar un conjunto cien billones de datos y poder unir toda esa información en la representación de una imagen en 3D, logramos observar que al ser proyectada en una unidad de JET estas generan que el conjunto de datos base se duplique produciendo saturación en el procesador computacional.

INTRODUCCIÓN

El objetivo de este trabajo es el conocimiento de las imágenes en 3D desde el nivel molecular y celular a partir de imágenes 2D mediante tecnicas de modelación propias de la física computacional con aplicación a la fisica medica[1].

La toma de imágenes por resonancia magnética (IRM) utiliza un poderoso campo magnético, ondas de radio y una computadora para producir imágenes detalladas del cerebro y otras estructuras craneales, que son más claras y detalladas que las que se obtienen por Rayos X. Esta técnica es de gran ayuda para el diagnóstico de los tumores cerebrales, de los trastornos y del oído interno entre otros. Pero esta técnica (IRM) inyecta al paciente gadolinio para obtener unas imágenes de los órganos y tejidos de gran detalle y calidad, siendo el gadolinio toxico para el cuerpo. Por otro lado, la radiación produce dos tipos de quemaduras, regenerativas y las no regenerativas; en las regenerativas el tejido humano puede regenerarse sin problemas en la segunda no es posible la recuperación. Con la finalidad de evitar someter a un paciente a estos peligros es que hemos elaborado este trabajo que obtiene imagen 3D de males del cuerpo [2],[3].

B. E. C. a. J. R. F. J. M. Butterworth, http://arxiv.org/abs/hep-ph/0201098, Phys. Rev. D 65, 096014, 2002 presenta un nuevo algoritmo, llamado “Marching Cubes”, que crea modelos triangulares de superficies de densidad constantes a partir de datos médicos 2D. Usándolo para dividir rodajas y conectarlas, crearon el caso que define triángulos topológicos en un algoritmo de procesos 3D y calculamos los vértices del triángulo con interpolación lineal [4].

En el trabajo de M. CACCIARI, FASTJET: A CODE FOR FAST kt CLUSTERING, AND MORE, Paris 6, France: LPTHE, Universit´e P. et M. Curie , 6 jul 2006. Las principales clases de algoritmos de agrupamiento de pixeles de baja resolución, como k_t, se discuten brevemente y se argumenta que el primero puede ser a menudo engorroso para definir la realización de la imagen en 3D, que es difícil de analizar en términos de su comportamiento con respecto a su base, y por otra parte, goza de una definición muy simple, y se puede demostrar fácilmente que es seguro. Su única desventaja potencial, es la cantidad de información, produciendo una gran escala de manipulación computacional complejo como el número de partículas o cubos (η), se supera mediante la introducción de un nuevo algoritmo geométrico que lo reduce [5].

A. G. J.-P. Thirion, The 3D Marching Lines Algorithm, Graphical Models and Image Processing, 58,pp. 503 – 509, 1996 ofrece la tecnica de “Marching Cubes” primero ofrecen acceso visual a los datos experimentales y teóricos. La aplicación de este método por lo general se basa en una pequeña tabla de búsqueda. Muchas mejoras y optimizaciones de “Marching Cubes” todavía lo utilizan, sin embargo, esta tabla de búsqueda puede dar lugar a deformaciones y topología inconsistente. Este artículo presenta una aplicación plena de la técnica para asegurar un resultado topológicamente correcto, es decir, una malla de colector, para los datos de entrada. Se completa el trabajo original para la resolución de la ambigüedad y de la viabilidad de la implementación. [6].

H. L. A. V. G. T. T. Lewiner, Efficient implementation of marching cubes’ cases with topological guarantees, ACM Press: Journal of Graphics Tools 8(2), pp. 1 – 15, publicado en el año 2003. Presenta un modelo de estructuras moleculares de la proteína 3D usando PDE (Ecuaciones Diferenciales Parciales) geométricas basado en el método de conjunto de nivel. El método de conjunto de nivel incorpora la forma del objeto molecular 3D dando como resultado la isosuperficies a nivel ajustado correspondiente a algún valor de un campo disperso como un discreto muestreado de rejilla rectilíneas, es decir, una rejilla volumétrica [7].

B. R. Schlei, “Hyper-Surface Extraction in Four Dimensions Theoretical Division Self Assessment, Special Feature, Los Alamos : a portion of LA-UR-04-2143-168” publicado el año 2004, realizó el estudio en donde visualizó un campo vectorial que es un tema importante en la visualización científica. Su objetivo es representar gráficamente los datos de campo en dos y tres dimensiones y en las superficies de una manera intuitiva comprensible. Aquí, se introduce un nuevo enfoque basado en la difusión anisotrópica no lineal. Permite una fácil percepción de los datos del vector de campo y sirve como un método de espacio escala apropiada para la visualización de patrón de flujo complicado [8].

Courtesy of G. T. Seidler, Physics Department, University of Washington, Seattle publicado en los Alamos en el año 2004. La obra trata de Espacio-Tiempo-Encerrando Extracción Volumen; Es un algoritmo es describe la hypersurperficie con valores que pueden estar contenidos implícitamente en (4D) los conjuntos de datos de cuatro dimensiones, como la secuencia de imágenes en tres dimensiones temporales de la tomografía computarizada variable en el tiempo. Cualquier hipersuperficie que sombrea el plano está garantizado para estar libre

de grietas accidentales, es decir, que siempre encierra completamente una región en el espacio 3D bajo consideración. Además, de información en el interior y exterior de las regiones encerradas se propaga a cada uno de los tetraedros, durante el uso de un mínimo uso de datos en la representación de los resultados finales, es más rápido que otras técnicas que generan múltiples dificultades [9].

B. Schlei, Volume-Enclosing Surface Extraction, GSI Helmholtz Centre for Heavy Ion Research GmbH, Planckstraße, 8 Jun 2012 "Modeling and computation of two phase geometric biomembranes using surface, Journal of Computational Physics", este trabajo consiste en múltiplos diagramas de flujo que pueden simplificar fenómenos de separación de imagenes que conducen a dominios coexistentes de diferentes composicione. En su trabajo propone una nueva herramienta computacional para calcular equilibrios basados en flujo de relación para un conjunto total de datos [10].

S. Fortune, in Proceedings of the second annual symposium on Computational geometry, p. 312, 1986. considera métodos de elementos finitos para aproximar la resolución de ecuaciones en derivadas parciales en las superficies. Realiza un estudio en superficie de elementos finitos triangulando, superficies implícitas usando el nivel de referencia establecido con el fin de formular los métodos que presentan la geometría necesaria y en el contexto de la evolución de las superficies [11].

A. R. M. Olshanskii, "A finite element method for surface PDEs: Matrix properties, Numer. Math. 114 491-520" publicado en el año 2010 trato un método de elementos finitos de superficie basada en un espacio estándar de elementos finitos de triangulación tetraédrica de un dominio externo que contiene una superficie 3D . Para el caso estacionario, se presentan los resultados de un método adaptativo basado en el indicador de error de tipo una superficie residual [12].

Crassin, F. Neyret, E. and Eisemann, "Beyond Triangles : Gigavoxels Effects In Video Games., SIGGRAPH Technical Talk" en 2009. publica que el fotorealismo siempre ha sido una Meta importante para gráficos por ordenador (CG). Hoy, uno de los principales problemas es la gestión de información. Representa grandes escenas de varias escalas y objetos muy detallados con precisión y es ahora uno de los problemas más difíciles en gráficos de computadora, no sólo para aplicaciones en tiempo real, sino también para películas destacadas. Métodos de

representación generalmente tienden a ser muy eficiente para escenas altamente complejas En tales escenas, muchos detalles geométricos tienen que ser contabilizados en cada píxel [15].

G. M. H. B. Nielson, “The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes., Proceedings of the 2nd Conference on Visualization” ’91,: IEEE Computer Society Press, pp. 83–91., publicado en 1991. estudia un método de elementos finitos para el cálculo de transporte y difusión de datos a lo largo de una superficie cambiante. El método no requiere una parametrización de una superficie o una extensión de una PDE de una superficie en un dominio externo. La superficie y su evolución pueden administrar de forma implícita, por ejemplo, como la solución de una ecuación de ajuste de nivel. Este enfoque permite, naturalmente, a una superficie sufrir cambios topológicos y la experiencia de las singularidades geométricas locales. La superficie y su evolución pueden administrar de forma implícita, por ejemplo, como la solución de una ecuación de ajuste de nivel [17].

A. a. W. J. Van Gelder, Topological Considerations in Isosurface Generation, ACM Transactions on Graphics: Volume 13, Issue 4, pp. 337–375, .publicado en 2004 propone una formulación de elementos finitos para aproximar el comportamiento de las membranas lipídicas. La membrana es discretizada por una malla de superficie compuesta de triángulos planos (velocidad de curvatura) se construye basado en la forma bilineal viscosa. Simulaciones numéricas muestran el comportamiento convergente del método propuesto en el límites de estabilidad; se obtienen numerosas pruebas de convergencia con mallas de alto rendimiento que forma el equilibrio de visión virtual también se reportan experimentos informáticos de la dependencia de la membrana deformada [18].

U. S. G. Survey, National Elevation Dataset. “Volume-Enclosing Surface Extraction, Helmholtz Centre for Heavy Ion Research” publicado el 8 Jun 2012 en su trabajo se presenta un nuevo método, que permite la construcción de isosuperficies triangulares de conjuntos de datos tridimensionales, como los datos 3D de imágenes y/o datos numéricos de simulación que se basan en paquetes que dan forma, de cubos. Esta novedosa técnica de volumen adjunta superficie de extracción, que ha sido nombrado VESTA, puede producir hasta seis diferentes espacios en 3D discretizados. Determinados con una técnica de construcción muy rápida y eficaz [19].

P. A. B. J. NING, An Evaluation of Implicit Surface Tilers, IEEE Computer Graphics and Applications, Volume 13, Number 6, "Characteristic cut finite element methods for convection-diffusion problems on time dependent surfaces" realizado en Uppsala University.: Tech. publicado en Abril del 2013 desarrolla un método de elementos finitos que se basa en un método de elementos de corte que es construido por incrustar la superficie en un fondo de cuadrícula y luego con la restricción a la superficie de un elemento, se define un espacio en la parte posterior a la base definida. La superficie se permite cortar a través de la red de fondo de una manera arbitraria [20].

Grande J, "Eulerian finite element methods for parabolic equations on moving surface, SIAM Journal on Scientific Computing" en el año 2014 desarrolla un método de elemento finitos para ecuaciones parciales planteadas sobre hypersuperficies en R^N, siendo N = 2; 3. El método utiliza las señales de las funciones de elemento oscuros en el plano y obtiene una superficie en un dominio volumétrico, según indicadores de error y valores estimados de las curvaturas de superficie. La estructura cartesiana de la malla mayor conduce a un proceso más fácil de adaptación mientras que el método trae elementos finitos para la instalación de la malla a la superficie [43].

M. C. a. G. P. Salam, http://arxiv.org/ "An eulerian space-time finite element method for diffusion problems on evolving surfaces" desarrolldao en 2010. Estudio un método numérico para la solución de ecuaciones deferenciales parciales sobre la evolución de las superficies. La hipersuperficie de evolución en R^N definiendo N como una dimensión de espacio-tiempo continuo. Que deriva y analiza una formulación variacional de una clase de problemas de difusión en el múltiple espacio-tiempo [45].

De la bibliografía revisada no se ha encontrado premisas que sugieran la utilización de imágenes 2D para generar imágenes 3D, por lo que el desarrollar una técnica que lo permita y lo vuelve innovador.

CAPÍTULO I

BASES TEÓRICAS DE CONSTRUCCIÓN DE IMÁGENES

1.1.- IMAGEN POR TOMOGRAFÍA COMPUTARIZADA (CT) DE RAYOS X

Hoy en día se denomina Imagen medica a la información virtual de enfermedades Esta información funcional permite diagnosticar algunas enfermedades y varias condiciones médicas mucho antes que otras modalidades de imagen médica, ya que se puede apreciar el trastorno (cáncer, tejido alterado, mal funcionamiento cerebral) antes de que haya dado lugar a alteraciones de la estructura (tumor, cicatrices), como muestra la Figura N°1.1.

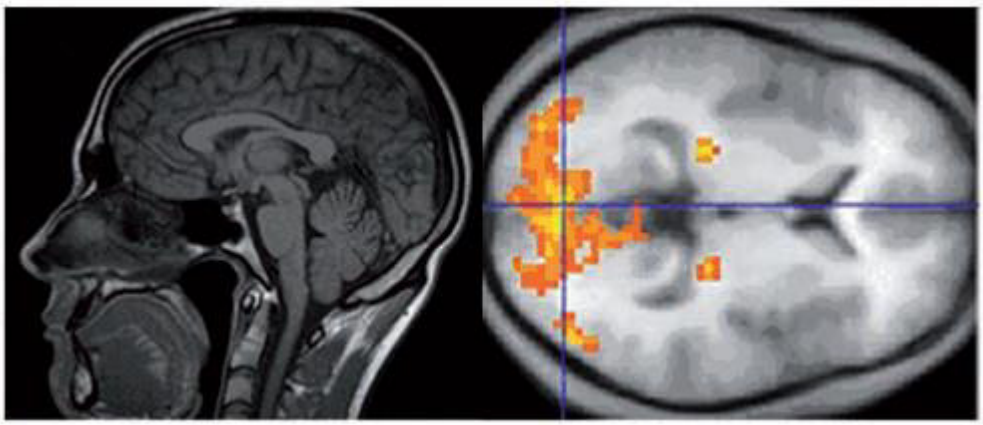


Figura Nº 1. 1*: Imagen CT de cráneo, corte sagital y axial. Se puede observar las tumoraciones carcinógenas [17].*

1.2.-MÉTODOS ESPECIFICOS PARA ELECTRONES EN LA OBTENCIÓN DE IMÁGENES MÉDICAS

1.2.1.-RANGO DE RECHAZO DEL ELECTRÓN

Una diferencia fundamental entre transporte de fotones y de electrones en código de simulación, es que los fotones viajan distancias relativamente largas antes de interactuar, mientras que el camino de los electrones es interrumpido no solo por los límites geográficos, sino también por los “pasos” de dispersión múltiple. El rango de rechazo de los electrones significa que los electrones con rango menor distancia a la frontera o región de interés es rechazada por la simulación computacional para ahorrar tiempo al momento de generar la placa radiográfica.

1.2.2.-REDUCCIÓN DE PARÁMETROS EN LOS PASOS DE LOS ELECTRONES

La simulación en la construcción de imágenes médicas permite el uso de pequeños pasos de los electrones en las proximidades de las interfaces y fronteras y pasos grandes en otros lugres [15]. Sus componentes son un algoritmo de correlación de longitud de paso que se basa en la teoría de dispersión múltiple [18], tengan en cuenta las diferencias entre la longitud del camino recto y la longitud total de camino curvo para cada paso de electrones, es algoritmo de correlación lateral que tiene en cuenta transportes laterales de frontera y un algoritmo de cruce fronterizo, que asegura que los electrones son transportados con precisión cercanía al interfaz de frontera [19].

Para terminar esta etapa de explicación concluiremos que todo lo relacionado a la construcción de imágenes médicas, hasta el comportamiento físico de los electrones en su totalidad está simulado y representado mediante algoritmos computacionales.

1.3-TÉCNICAS DE IMAGEN PARA LA OBTENCIÓN DE UN RANGO DE ESTUDIO DEL PIXEL

El tratamiento del pixel depende del estado actual de la enfermedad debido a su intensidad, la clasificación histológica y la identificación de los factores establecidos. El pixel es un predictor de respuesta al tratamiento, claramente establecido en el momento actual, este predice el riesgo de recurrencia de la enfermedad y la supervivencia total teniendo en cuenta factores tales como la edad, el estado actual de la enfermedad, el estado general del paciente, y la presencia o ausencia de la elevación de una enzima llamada lactato deshidrogenasa (enzima catalizadora que se utiliza para detectar lesiones del tejido) que es con la que nos ayudamos en la representación de pixeles para su posible representación y clasificación de intensidad del pixel que es fácil de medir computacionalmente hasta con muestro software Matlab .

1.4.-DETECTORES SENSIBLES

Los detectores sensibles, se utilizan para almacenar información sobre las interacciones de una partícula con la materia de bits (representación numérica del pixel, que para los médicos se representa como celula), usando información de los pasos a lo largo de la trayectoria de una partícula. Unos bits es la información producida por un fotón de una interacción física de una

trayectoria de una partícula en una región sensible del detector, dicho concepto es ilustrado en la Figura N° 1.2.

El ordenador, solo registra y almacena la información relacionada a los bits de los volúmenes que están conectados a un detector sensible. Toda la información sobre las interacciones que ocurren en los volúmenes no sensibles se pierde. Los detectores sensibles se le llaman GATE [25].

1.5.-DATOS DE SALIDA Y CONFIGURACIÓN DE RECONSTRUCCIÓN

El GATE, está siempre usado en la construcción de pixeles de imágenes médicas hay varios tipos de formato de salida que pueden ser leídos de manera predeterminada. Para todos los sistemas están disponibles los archivo tipo m,txt, py, y otros .El formato cmd los datos que son comprimidos automáticamente, por lo recomendado en caso de alta estadística. Los datos de salida cmd, es descrito durante el procesos de creación de imágenes por diferentes cámaras o aparatos de creación de imágenes y todo se almacenan en un archivo .m o de cualquier otra índole ya mencionada.

Comprender sus pruebas, qué son, cómo funciona y qué pueden mostrar o no, puede ayudarlo a sentirnos más cómodo y con un mayor control los programas que diseñamos.

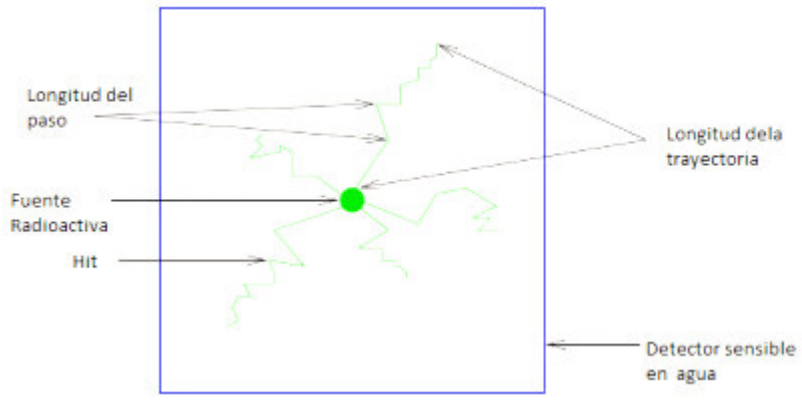


Figura Nº 1. 2: *Detector de movimiento de fotones [13].*

1.6.- ANÁLISIS MATEMÁTICO DE UNA CONSTRUCIÓN DE LA IMAGEN 3D

Siendo el dominio abierto en $\boldsymbol{R^3}$ y **Γ** una híper-superficie (imagen 3D representada por más de un millón de datos) compacta en una región en el plano $\boldsymbol{C^2}$ contenida **Ω** (es el conjunto de datos 2D compuesta por más de un millón datos). Para una función dada por $\boldsymbol{g : \Omega \rightarrow R^2}$ la derivada tangencial sobre **Γ** es definida por [5].

$$\nabla \Gamma g = \nabla g - (\nabla g . n() n) \qquad (1.1)$$

Ecuación que representara nuestro universo completo de imagen. Donde **n** el vector unitario normal a **Γ**. Denotando por $div_\Gamma = \nabla_\Gamma .$ es el operador divergencia de la superficie dada por $\Delta_\Gamma = \nabla_\Gamma . \nabla_\Gamma$ que opera a **Γ** teniendo la ecuación EDP (Ec. diferencial parcial) está dada por:

$$\nabla_{\Gamma\mu} = f \quad \text{en } \Gamma, \text{ con} \qquad \oint f ds = 0 \qquad (1.2)$$

Siendo: μ,en nuestro trabajo la representación del pixel que varía de acuerdo la calidad de color representada y $\boldsymbol{f}$ =una partición de la superficie en el plano de **Γ**

La ecuación (1.2) es una ecuación que tiene muchas aplicaciones en lo que llamamos superficies planas 2D, que surge del criterio de convergencia de la superficie en muchas aplicaciones, lo consideramos sobre **Γ**. Para poder conseguir su elevación consideramos $w: \Omega \rightarrow R^3$ donde $\boldsymbol{w}$ define la capacidad de elevación de la superficie **Ω**.

Para conseguir la elevación de **Γ** aplicaremos el vector $\boldsymbol{w.n}$, entonces la conservación de una cantidad escalar con una difusión en **Γ(t)** dejando en claro que no se debe de confundir $\boldsymbol{t}$ con una variable temporal sino con una cantidad de decimales del pixel que nos conduce a la siguiente EDP [6].

$$\dot{\mu} + (dir_\Gamma w)\mu - \xi \Delta_\Gamma \mu = 0$$
$$en\ \Gamma(t) \qquad (1.3)$$

Donde $\dot{\mu}$ representa la razón del cambio del pixel a través de la superficie $\xi > 0$ representa la razón de cambio a través de la imagen y la consideramos constante. Si asumimos $\boldsymbol{w.n=0;}$ no realiza ningún tipo de elevación de la superficie. Aunque la metodología del trabajo se aplica a las ecuaciones (1.3) vamos a presentar el método y el análisis del problema estacionario(entendiendo como estacionario una porción cerrada de superficie mínima) [7].

$$-\xi \Delta_{\Gamma\mu} + w . \nabla_{\Gamma\mu} + (c + dir_\Gamma w)\mu = f en\ \Gamma \qquad (1.4)$$

Para poder determinar el ciclo indefinido del análisis de imagen necesitamos reconocer toda la imagen mediante la ecuación:

$$\int_{\Gamma} q(dir_{\Gamma} f) + f.\nabla_{\Gamma} q ds = \int_{\Gamma} \kappa(f.n) q ds \quad (1.5)$$

Donde q es una función que suaviza la superficie de la imagen y f es un función de campo, donde $\kappa = div_{\Gamma} n$ significa la torcedura de la curvas en la superficie. Así introduciremos una función de vecindad para poder analizar el conjunto de pixeles dada por:

$$a(\mu, \nu) = \int [\xi \nabla_{\Gamma} \mu . \nabla_{\Gamma} \nu - (w.\nabla_{\Gamma} \nu)\mu + c\mu\nu].ds \quad (1.6)$$

Donde **ν** son los vecinos que están alrededor del μ que son los pixeles analizados.

La formulación (1.9) en la que se encuentra $\boldsymbol{\mu} \in \boldsymbol{\nu}$ al que se le tiene que integrar todo el conjunto de vecindades para poder obtener la imagen total.

$$\|\nu\|_{\Gamma}^2 = C_P \|\nabla_{\Gamma} \nu\|_{\Gamma}^2 \quad (1.7)$$

Suponiendo C_P , un factor de corrección debido a que las vecindades poseen error pero esto lo veremos más adelante aclararemos este factor y su funcionamiento .

1.7.-MÉTODO DE ELEMENTOS FINITOS (FEM)

Es un método que analiza los resultado del conjunto de datos de la imagen y da una explicación de proyección en toda la imagen .

1.7.1.-CÓDIGO DE ELEVACIÓN

Un jet(intensidad de pixel) es un principio de proyección de cubos en sucesión en la misma dirección de la intensidad ***w*** dada por la ecuación ***(1.3);*** se necesita aclarar por supuesto que no tiene solución teórica pero si numérica de acuerdo al conjunto de datos que el software puede leer con una definición de máximo cuidado para hacerlo una herramienta de análisis preciso en nuestro trabajo [8].

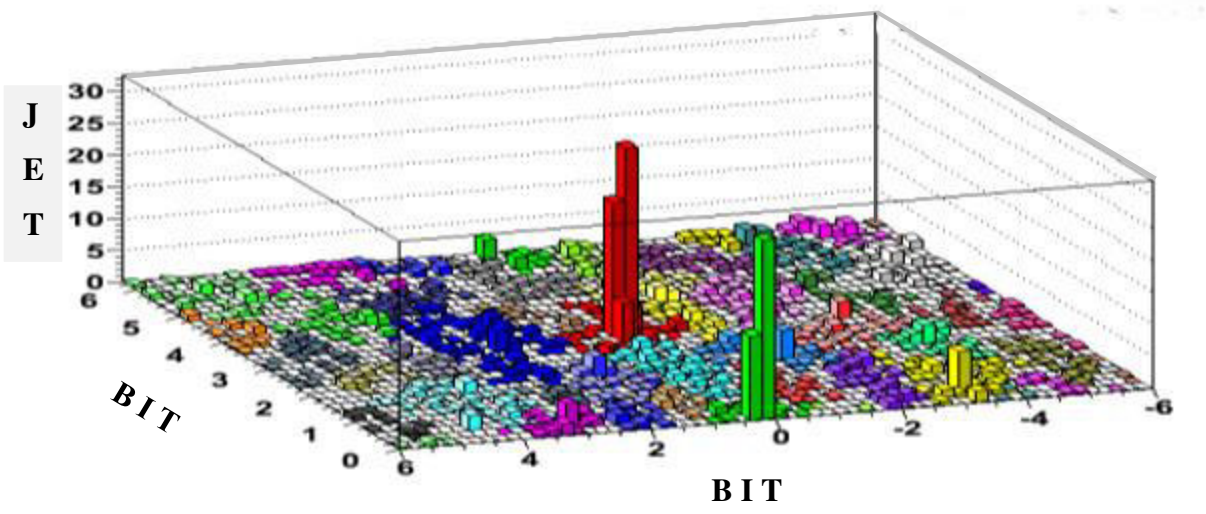
$$\Gamma = \prod_{0}^{\infty} w \tag{1.8}$$

El jet generalmente se basa en la recombinación de datos sucesivos, que tiene todo el pixel. Entre sus ventajas se encuentra (a) se limita deliberadamente a crecer hacia atrás o hacia adelante, a través de la secuencia de intensidad y ramificación, lo que significa que las proyecciones reconstruidas están a base de un patrón de base irradiado, (b) permite que cada pixel se descomponga en valores diferentes, lo cual es útil para identificar los productos de desintegración a través del espacio [42].

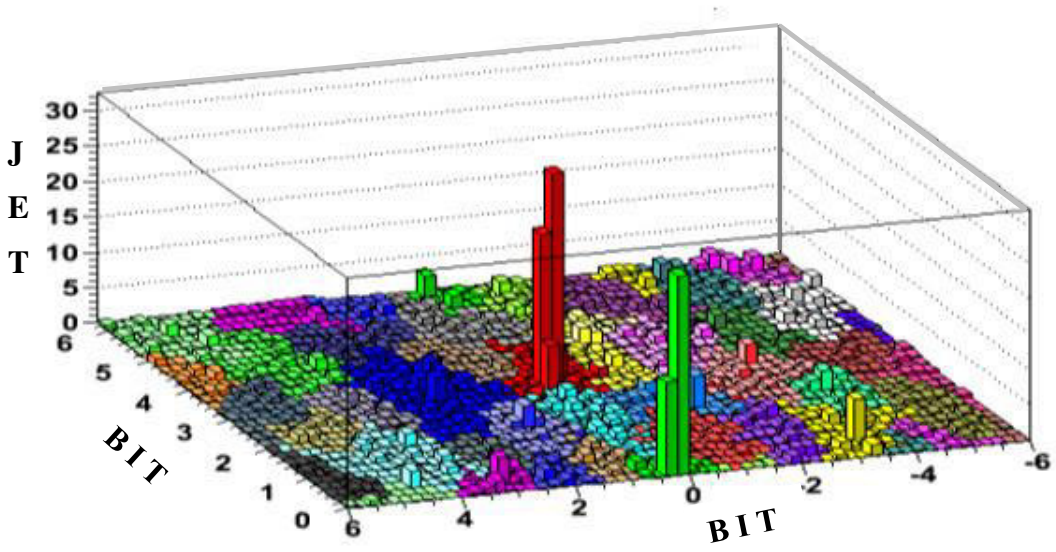
1.7.2.-EL ALGORITMO VECINDAD PROYECTADA

Para que cada partícula i establezca su vecino más cercano g_i y la construcción de los caminos de d_{ig_i} (datos de región) y d_{iB} (vecinos de la región). Se sigue los siguientes pasos:

- Encontrar el mínimo valor de d_{min}de los d_{ig_i}, d_{iB}.
- Fusionar o eliminar las partículas correspondientes a d_{min}según corresponda.
- Identificar que los vecinos más cercanos que han cambiado y actualizado las matrices de d_{ig_i} y d_{iB}.



(a)



(b)

Figura Nº 1. 3: *En esta figura mostramos. a) Las línea negras de diez puntos son representados por el vector* ***n*** *que es medidor de altura de* ***w*** *(b) Todos los valores* ***w*** *llegan a agruparse N partículas para diversos jet buscados [9].*

1.8.-EL MARCO DE EXTRACCIÓN DE ISOSUPERFICIE

Antes de explicar la extracción de superficie con VESTA (Extracción Superficial Volumétrica Encerrado por un Algoritmo), consideramos en primer lugar la DICOMEX (Extracción de Contorno en 2D con Degeneración de Intensidad Extraídos de la Base).

1.8.1.- EXTRACCIÓN DE CONTORNOS EN 2D CON DEGENERACIÓN DE INTENSIDAD EXTRAÍDOS EN LA BASE (DICOMEX).

La Figura N° 1.4 nos muestra ejemplo de imagen binaria con $6x6 = 36$pixeles (elementos discretos de imagen). Supongamos, que los pixeles grises han sido segmentados, es decir, que han sido seleccionados para un recinto. En lo que sigue, designaremos pixeles segmentados como “activo” y los otros pixeles “inactivos”. El objetivo ahora es encerrar los activos grises con pixeles con contornos mientras se hace uso del algoritmo DICOMEX. En primer lugar todos los ICV (Vectores de Contornos Iniciales) deben ser recopilados. Los ICV se orientan aquí como bordes de pixeles que separan de un pixel activo de un inactivo [10].

Por ejemplo para un solo pixel que no tiene ningún vecino activo en todos los contornos de la Figura N° 1.4. (b) disponemos de cuatro ICV individuales (en la figura que esta enumerada de 0 al 3). Debido a esta construcción, posible descartar la información del interior y/o exterior de una forma (es decir, una colección de pixeles segmentados) que necesita ser cerrado por un numero finito de contornos. En una segunda etapa, la ICV están conectados para encerrar todos los contornos de pixeles. Tenga en cuenta, que cada uno de ICV ambos comienzan en un punto y terminan en el mismo punto Figura N° 1.4. (c). Pero a veces es posible que un pixel activo libre está en contacto con otro pixel activo a través de un punto único Figura N° 1.4.(c) [11].

Los últimos resultados en una situación en la que tenemos dos caminos y dos ICV salientes para este punto de contacto en particular Figuras N° 1.4 (c, d). A continuación vamos a llamar a un POA (Puntos de Observaciones Ambiguas). Con el fin de evitar vacíos en el conjunto final de contornos, uno tiene que tratar estos POA de manera especial. En la Figura N° 1.4 (d) un diagrama de conectividad se representa. Si uno se conecta a un entrante ICV y un ICV saliente que pertenece al mismo pixel activo como para el ICV entrante, entonces uno realiza un giro a la izquierda, es decir, se sigue uno de los caminos de los vectores dobladas blancas. Por lo contrario, si uno conecta a ICV, entrante un ICV saliente que pertenece a otro pixel activo como para el ICV entrante, entonces uno realiza un giro a la derecha, es decir, uno sigue una de las trayectorias de los vectores doblados negros. Las vueltas particulares serán a conducir un modo de separación de forma al lado POA, respectivamente [12].

A continuación, mostramos la Figura Nº1.4 dejando en claro que los casos (e) y (f) son caso no obtenidos en este trabajo abriendo el camino para trabajos posteriores que sería un gran avance en este campo de la física informática. Por ahora expresaremos la ecuación con la que realizamos este tipo de corrección

$$\xi C_P^3 - |div_\Gamma w..x| \geq c_0 \qquad (1.9)$$

Donde C_P es un factor de corrección de este problema y c_0 es un contador del número de errores a través de la imagen.

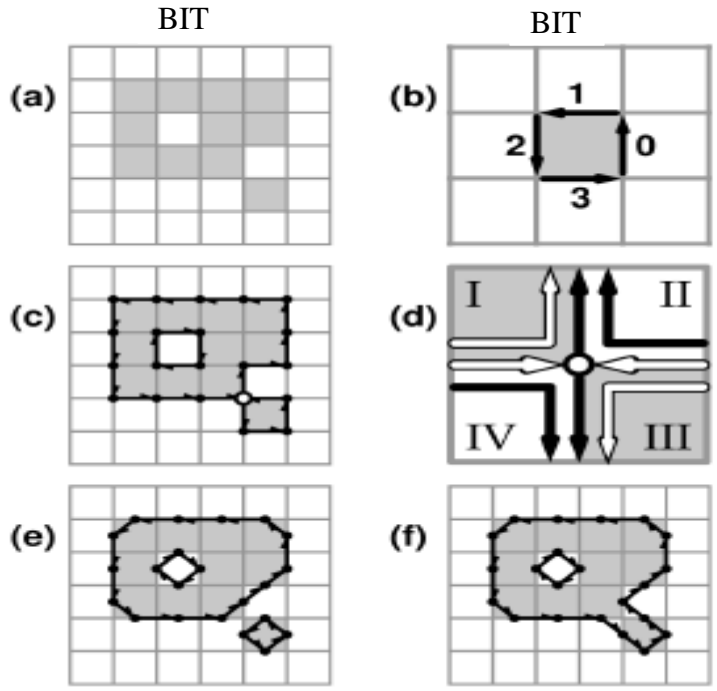


Figura Nº 1. 4:*(a) Es una imagen binaria; (b) vectores de contorno inicial para un solo pixel; (c) vectores de contorno inicial y coyunturas (puntos negros) para la imagen binaria ;(d) diagrama de conexión para un único punto de ambigüedad (punto blanco); (e) los vectores de contorno dilatados y puntos de apoyo (puntos negros) resultantes del modo "desconectado" (f) como en (e) pero como resultado de la modalidad de modo "conectado"[11].*

1.8.2.-CONSIDERACIONES INICIALES PARA EL VÓXEL

VESTA encerrara vóxeles que tienen un valor inherente de campo, por ejemplo, un tono de gris, para elevar un pixel a vóxel utilizando el valor ***w*** que representa la intensidad del pixel. Un vóxel es una proyección del pixel a 3D. Para ser más específicos, está representada por un cubo y, cuya altura esta calibrada por ***n.***

La Figura N° 1.5. (a), se muestra dos vóxeles vecinos, el vóxel activo, está marcado con una esfera en su centro que representa la determinante de la matriz del pixel. El segundo vóxel no tiene ninguna esfera en el centro, ya que se considera inactivo, es decir, este vóxel tiene un valor por debajo de la intensidad inicial. Entre un activo y un vóxel inactivo tendremos una contribución a la superficie deseada [13].

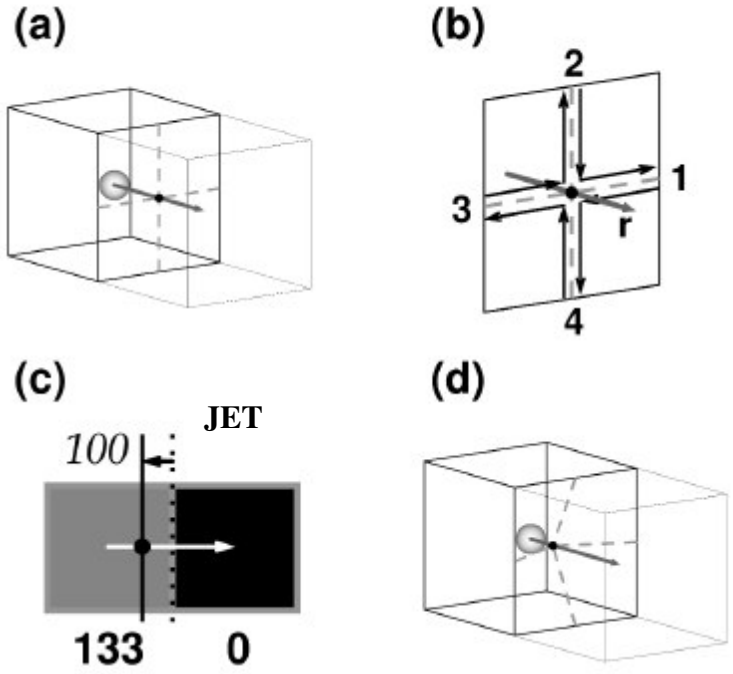


Figura Nº 1. 5: *Aquí representamos (a)dos vóxeles (uno es activo (esfera) y uno es inactivo) que están en contacto a través de cada vóxel, y un vector de rango correspondiente; (b) una cara vóxel con un punto negro en su centro, su gama de vectores* ***,r,*** *(gris) ,y cuatro puntos de borde (1,2,3,4) que cada uno se conectan con una línea discontinua en el vóxel del centro de la cara , y de la cara de vectores; (c) el desplazamiento de contorno para un par de pixeles (ver texto); (d) como en (a) pero un centro de la cara de vóxel desplazados[14].*

En la Figura N° 1.5. (b), mostramos una cara que está marcado por un punto negro en su centro. Tales centros en la cara del vóxel finalmente serán el apoyo de puntos de la superficie de VESTA. En el caso de que una isosuperficies debe ser construido, los centros de la cara del vóxel se puede mover dentro de los límites de su rango correspondiente del vector ***w*** que tiene su origen en el centro del vóxel activo y que tiene su origen el centro del vóxel inactivo vecino.

La Figura N° 1.5. (c), ayuda a ilustrar, la intensidad del pixel que puede variar y llega a transformarse en voxel teniendo en cuenta la analogía 2D del pixel y la perdida de intensidad a través de la elevación.[15].

La Figura N° 1.5. (d), ayuda a ilustrar, la intensidad del vóxel ya proyectado para poder transformar en isosuperficies teniendo como base la analogía 2D de la cara de un vóxel [20]

.

CAPITULO II

MARCHING CUBE

2.1.-ESPACIOS CON ALTA DEFINICIÓN

Se refiere especialmente a mostrar la figura al detalle en forma virtual proveniente de un mismo nivel bien definido, para mostrar su alto rendimiento.

2.2.- ESTRATEGIAS DE RELLENO PARA LAS DEFORMACIONES

En la Figura N° 2.1. (a) se muestra una abertura debido a la poca intensidad del pixel en la base de la imagen proyectada en cuestión, que se observa como una deformación a lo largo de las mallas en la superficie de frontera, esta posee un nivel diferente entre las otras mallas. En la Figura N° 2.1. (b) se usa la técnica de triangularización usando la existencia de los vértices, para nivelar al detalle la frontera pero esta técnica genera serias dificultades en la textura porque se podría falsear la verdadera formación 3D de la imagen. Este ajuste que es realizado por la triangularización al normalizar el vector de puntos que rodean la abertura en la frontera para completar el contraste. En la Figura N° 2.1. (c) existe una deformación de altura con respecto a la frontera de la textura debido a un alto nivel de intensidad del pixel en la región 2D para corregir esta deformación tenemos que cortar al nivel de las mallas. Este método también lleva consecuencias negativas cuando nos aproximamos a la verdadera frontera, si es que no sabemos la verdadera consecuencia del pico, ni la verdadera forma de este pico en la frontera. En la Figura N° 2.1. (d) la deformación esta intencionalmente llenada con un gran número de triángulos es casi igual que la Figura N° 2.1. (b). Esto hace posible la capacidad de construir la textura exterior [18].

Los métodos mostrados en las Figura N° 2.1. (e) y (f) se les conoce como “skirts” y “flanges” respectivamente. En ambos casos la falla es corregida por triángulos bajando o subiendo garantizando así que la textura sea la correcta en la imagen 3D. Por el borde de la falla los triángulos se extienden hacia abajo ocasionando así los mismos problemas de la Figura N° 2.1. (b). Estos problemas aparecen ya sea por un fallo experimental o por alguna deformación en la placa 2D [22].

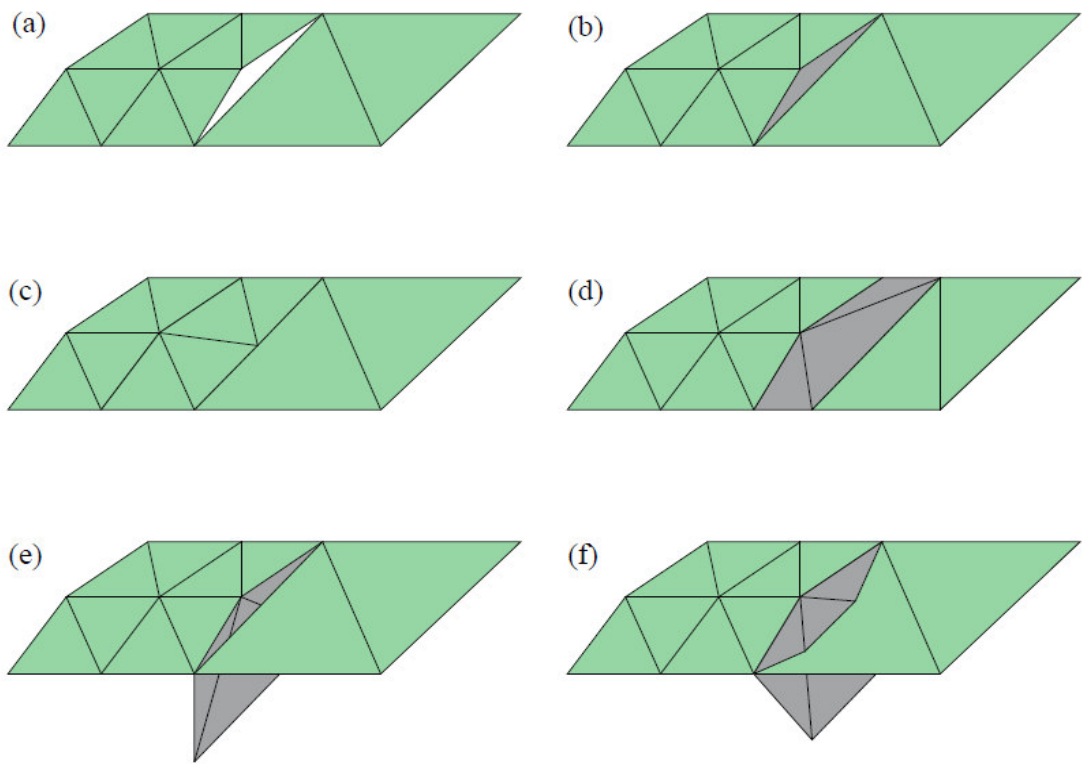


Figura N° 2. 1: *En la siguientes figuras están representadas (a) Es una deformación de hueco vacío triangular. (b) El triángulo es rellenado por mediante estrategias de triangularización consiguiendo así que la deformación este al nivel de la textura en la frontera. (c) El centro del vértice esta encima de la textura fronteriza. (d) Corte del pico y relleno del mismo por triangularización. (e) Borde adicional de triángulos de alta profundidad y corta base. (f) Ampliación de triángulos y anchura de base[24].*

2.3.- CONSTRUCCIÓN DEL VOXEL

2.3.1.-EXTRACCIÓN DE ISOSUPERFICIE

A partir de una base en el plano tenemos que ser capaces de construir computacionalmente un espacio en tres dimensiones, representado por vértices triangulares que pueden ser realizados por los graficadores computacionales. Las muestras computacionales pueden estar en el interior de superficie, en el exterior de la superficie y en la frontera de la superficie. El rango de distancias es limitado por la selección de precisión basada en el número de bits almacenando datos de cada muestra de valores y distancias fuera del rango que sujetan los mínimos y máximos valores. Ocho bits de almacenamiento por muestra de valores poseen suficiente precisión para realizar la construcción del volumen [25].

La construcción de un conjunto de vóxeles se da por ***n x m x l*** sometidos a una prueba regular de valores para obtener una imagen tridimensional, midiendo rejillas de ***n*** unidades en la dirección ***x***, ***m*** unidades en la dirección ***y***, así como ***l*** unidades en la dirección ***z***. Usando ocho bits por conjunto de vóxeles que son obviamente clasificados de un total de ***nml*** bits almacenados [27].

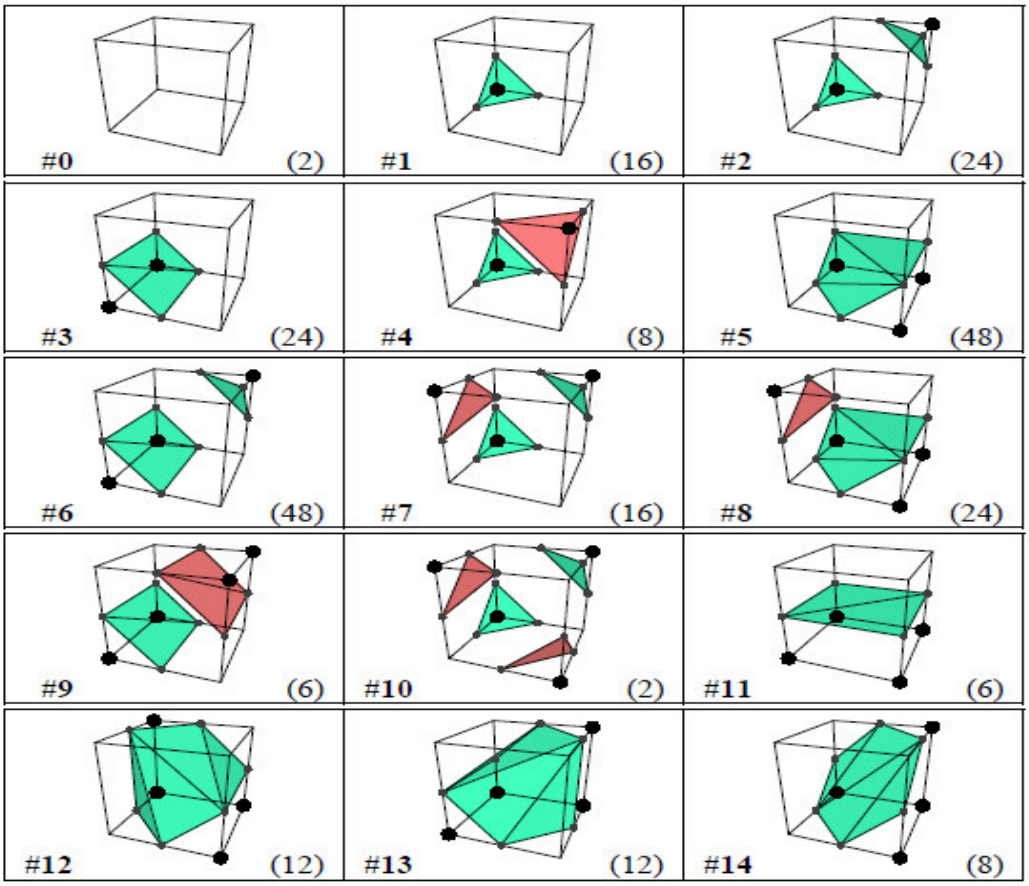
2.3.2.- MÉTODO DE MARCHING CUBE

Es un método muy conocido para la construcción de una malla de triangulo cuyo objetivo es nivelar la superficie para la obtención de la imagen [30]. El trabajo de este algoritmo consiste en extraer la isosuperficies de la imagen procesando cada cubo independiente de los otros en la superficie de vóxeles. En cada cubo determinamos qué parte de la imagen pasa por el cubo, si la superficie de la imagen pasa por dentro del cubo es necesario aplicar triangularización utilizando los ocho vértices del cubo. El resultado de todos los cubos ya operados por la triangularización se combina para la construcción de la imagen completa [31].

Cuando procesamos un solo cubo, de la sucesión de cubos primero clasificamos cada uno de los ocho vértices de las esquinas del cubo identificado como valores positivos si están dentro de la superficie de la imagen y como valores negativos si están fuera de la superficie de la imagen. Los vértices pertenecientes a la triangularización interna de un cubo cuando se encuentran en el borde o en un extremo se pueden clasificar computacionalmente como: "outside" fuera o espacio vacío, y por otro lado "inside"o dentro de la imagen[32].

Las clasificaciones varían entre **"inside/outside"** dentro y fuera sucesivamente de las ocho esquinas del cubo existen exactamente $2^8 = 256$ posibles distintas combinaciones diferentes para cada cubo. El algoritmo original de "Marching Cube" representa estos 256 casos en 15 casos equivalentes como se muestra en la Tabla N° 2.1. Dos casos pueden permanecer a una misma clase cuando hacemos girar un cubo alrededor de un eje que pasa por el centro para así pueda coincidir con otro caso exactamente. Otra manera de simplificar dos casos es cambiando lo exterior por lo interior de un cubo. En conclusión los casos que son reflejos de sí mismo no se consideran en el algoritmo de "Marching Cube".

Tabla N° 2. 1: *Estas son las 15 clases de equivalencia reconocidas por el algoritmo de "Marching Cube ". El número de la esquina inferior izquierda de cada entrada es el índice de clase y el número en la esquina inferior derecha es el número de casos que pertenece a la clase de equivalencia de los 256 casos totales. Un punto negro en la esquina indica que está dentro del volumen del sólido y la esquina sin un punto negro indica que esta fuera del volumen del sólido. Triángulos verdes indican que son el frente con respecto al punto de vista y triángulos rojos indican que son caras ocultas detrás del material del cubo[45].*



2.3.3.- CASOS AMBIGUOS

Este problema se observa cuando consideramos una sola cara para dos esquinas diagonales opuestas, estas pueden estar dentro de la imagen o fuera de la imagen como se muestra en la Figura N° 2.2**.** Un vértice se coloca en los cuatro bordes de la cara, desde la frontera del solido que genera un espacio vacío tiene lugar a lo largo del borde. Se dice caras ambiguas porque hay dos maneras de conectar estos cuatro vértices con los bordes para que las esquinas exteriores se encuentren en el exterior de la triangularización y las esquinas interiores se encuentran en el interior de la triangularización. En la Figura N° 2.2. (a) los triángulos del material de la imagen están separados por un espacio vacío que forma la cara del cubo. En la Figura N° 2.2. (b) los

triángulos que están ubicados en los bordes de la cara están separados por el material de la imagen del solido, estos triángulos son espacios vacíos [33].

Si optamos por hacer cumplir algún caso descrito en la Tabla N° 2.1 entonces se crea problemas en algunas situaciones con dos cubos que comparten la misma cara, esta se puede ignorar en algunas ocasiones, pero si no es así se tiene que invertir uno u otro cubo para la adecuada formación de la imagen utilizando la opción de **"inside/outside"** de acuerdo a la situación del cubo. Hay un resultado de falta de coincidencia entre dos cubos genera una especie de agujero rectángulo en la triangularización final de los dos cubos como se muestra en la Figura N° 2.3.

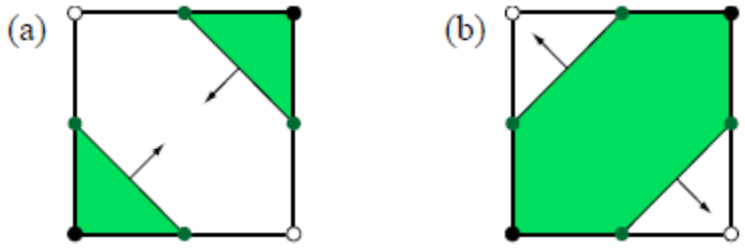


Figura N° 2. 2: *Hay dos maneras de conectar los vértices de una cara para la cual las esquinas diagonales opuestas tienen el mismo estado en el interior como en el exterior del solido de la imagen. Rincones con puntos negros se clasifican como interior y rincones con puntos abiertos se clasifican como fuera. La región verde representa la parte de la imagen del sólido y las flechas representan la dirección normal de la superficie hacia el exterior a lo largo de los bordes[37]*

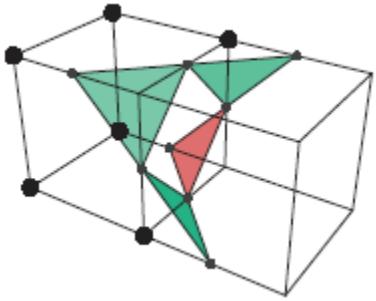


Figura N° 2. 3: *Dos cubos una cara ambigua y el cubo de la izquierda de invertirse para que coincida con la respectiva equivalencia del tejido de la imagen. Los bordes de las triangulaciones resultantes no se ajustan a lo largo de la cara compartida y se crea un agujero rectangular grande. Polígonos verdes están ubicados frente a la parte exterior del cubo y polígonos rojos están ubicados en la parte interior del cubo[38]*

Una solución notable al problema de cómo los bordes deben conectar a los vértices en una cara ambigua o compartida por dos cubos fue diestrita por Nielson y Hamann [47]. Su acción consiste en determinar de qué manera la interpolación bilineal (los valores de muestra) de la función *F(s, t)* se comporta en la cara. Representado en la Figura N° 2. 4 la función *F(s, t)* se comporta como [34]:

$$F(s,t) = (1-t)\,[(1-s)\,A + sB] + t[(1-s)\,C + sD[]] \quad (2.1)$$

Donde los valores de A, B, C y D son los valores de las cuatro esquinas de la cara, **s** y **t** varían linealmente desde cero como se muestra en la Figura N° 2. 4:. Nosotros ´podemos encontrar un punto $P_0(s_0, t_0)$ que es silla (no se sabe si es máximo o mínimo) al derivar $\frac{\partial F}{\partial s}$ y $\frac{\partial F}{\partial t}$ igualando a cero obtenemos [35]:

$$\begin{aligned} & s_o \\ & = \frac{A-C}{A-B-C+D} \ldots . asi \ldots \rightarrow 1 \\ & - s_o = \frac{D-B}{A-B-C+D} \\ & t_o \\ & = \frac{A-B}{A-B-C+D} \ldots . asi \ldots \rightarrow 1 \\ & - t_o = \frac{D-C}{A-B-C+D} \end{aligned} \quad (2.2)$$

Al reemplazar esto en la función *F(s, t)* obtenemos:

$$F(s_0, t_0) = \frac{AD - BC}{A-B-C+D} \quad (2.3)$$

Este valor $F(s_0, t_0)$ en el espacio decide si hemos de usar la triangularización de la Figura N° 2. 2:(a). o la Figura N° 2. 2:(b). Dado que A y D son ambos valores positivos así como B y C son ambos valores negativos el denominador de la ecuación (2.3) es siempre positivo. Por lo tanto el numerador se puede usar para clasificar si el punto está dentro o fuera del volumen de la imagen. Si AD < BC entonces el punto silla está dentro del volumen de la imagen entonces escogemos la triangularización como un solo componente.[39].

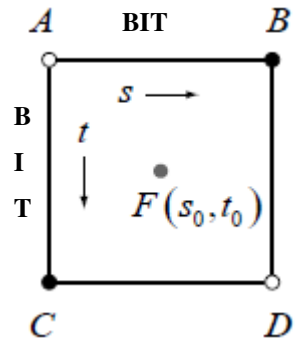


Figura N° 2. 4: *Una cara ambigua esta parametrizada para dos valores lineales de* ***s*** *y* ***t****. Teniendo* ***s*** *una variación desde cero a la unidad del lado en dirección horizontal y* ***t*** *una variación desde cero a la unidad del lado en dirección vertical[39] .*

2.4.-MODIFICACIÓN DE LA TECNICA COMPUTACIONAL DE "MARCHING CUBE"

Al generar la superficie de la imagen basada en vóxeles, nuestro interés más importante es que la superficie resultante sea preciso es decir que cada componente es conectado en forma continua y sin agujeros sin importar los valores de la superficie de la imagen. Asumiendo que este requisito se cumple nuestro objetivo principal entre las precauciones restantes es el máximo rendimiento usando el mínimo recurso computacional para obtener resultados en tiempos cortos.

Con estos objetivos de solidez y velocidad en mente, hemos desarrollado una nueva versión modificada del algoritmo de "Marching Cube" que elimina los casos ambiguos sin añadir un cálculo adicional por cubo. La clave sobre el algoritmo de "Marching Cube" que lleva nuestra versión modificada es que los casos ambiguos siempre surgen debido a la inclusión de los diversos casos que se muestra en la Tabla N°2.1. Vemos la necesidad de dividir algunos de estas clases en dos clases distintas [40].

Cuando consideramos los posibles estados computacionales **"inside/outside"** dentro y fuera sucesivamente de las cuatro esquinas de una cara de un cubo, en realidad solo hay cuatro casos no triviales después eliminamos los que son equivalentes a través de la rotación, y uno de

ellos es el caso ambiguo que se muestra en la Figura N° 2.2. Podemos satisfacer el requisito de congruencia de borde del volumen de la imagen cuando existe un caso ambiguo siempre conectamos vértices en los bordes adyacentes que comparten una esquina inferior (Esta regla se llama polaridad preferida [44]. Siguiendo esta regla, los cuatro casos mostrados en la Figura N° 2.5. son las únicas configuraciones de borde no triviales permitidos por nuestro algoritmo (y una quinta configuración es la cara trivial que no tiene borde por que las cuatro caras de las esquinas tienen el mismo estado en el interior o exterior)[41].

Nuestro algoritmo de "Marching Cube" modificado no permite la configuración de borde que se muestra en la Figura N° 2.2.(b), y por lo tanto la triangularización del cubo izquierda en la Figura N° 2.3 no se considera por ser elemento contaminante. Esta situación exige que se introduzca un nuevo conjunto de clases de equivalencia de triangularización que se exige configuración de bordes por caras como se muestra en la Figura N° 2.5

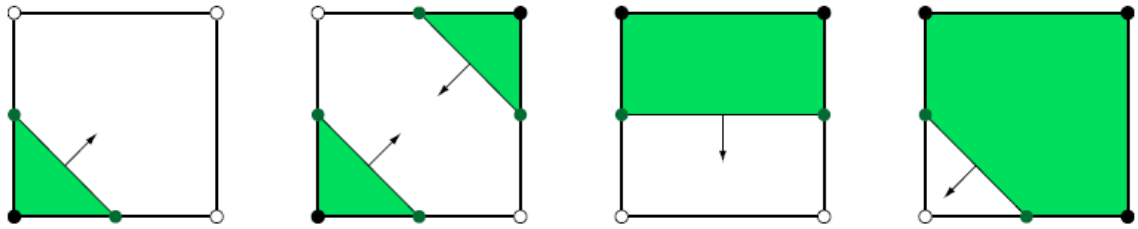


Figura N° 2. 5: *Estas son las únicas cuatro configuraciones de borde no triviales que pueden introducirse en la cara de un cubo por la modificación de "Marching Cube" con su debido a una rotación del cubo. Vértices con puntos solidos se clasifican en el interior y vértices con puntos se clasifican como fuera. La región verde representa el espacio sólido, y las flechas representan la dirección normal de superficie hacia el exterior a lo largo de los bordes[40].*

Remediamos esta situación mediante la eliminación de la inversión de la relación de equivalencia para las clases que tiene una cara ambigua, dejando solo la rotación. A continuación, establecemos tres nuevas clases de equivalencia para contener los casos que tengan más de cuatro esquinas interiores en el volumen de la imagen proveniente de la clase de equivalencia original #2, #6 y #7 de la Tabla N° 2.2. El resultado es el conjunto de las 18 clases de equivalencia mostrados en la Tabla N° 2.2, donde se muestran tres nuevas clases en la última fila. Las clases #15, #16 y #17 7 de la Tabla N°2.2.se componen de la representación inversa de los casos #2, #6 y #7, y sus respectivas rotaciones [42].

Tabla N° 2. 2: *Estos son las 18 clases de equivalencia identificados por nuestro algoritmo de "Marching Cube" modificado. El número en la esquina inferior izquierda de cada entrada es el índice de clase, y el número en la esquina inferior derecha es el número de casos que pertenece a la clase de equivalencia de los 256 casos totales. Un punto negro en una esquina indica que está dentro del volumen de la imagen, y la esquina sin punto indica que esta fuera. Triángulos verdes están al frente de la cara del cubo y triángulos rojos están detrás del material del cubo[45]*

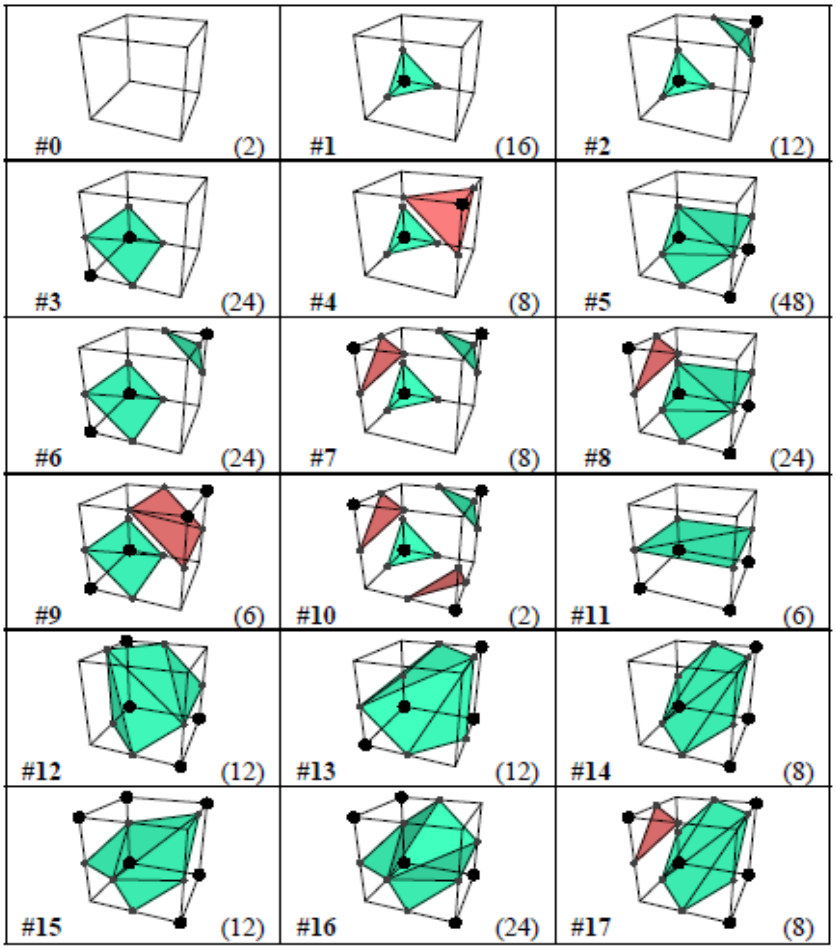


Figura N° 2. 6: *Dos cubos que comparten una cara ambigua se muestra por primera vez en la* ***figura (2.3).*** *Desde la inversión de los cubos que no es parte dela relación de equivalencia, la celda de izquierda es un miembro de uno de los nuevas clases de equivalencia.[41] .*

2.5.-IMPLEMENTACIÓN PARA UN ALTO RENDIMIENTO

Nuestra implementación del algoritmo de "Marching Cube" modificado fue diseñada con los siguientes objetivos en mente:

- Debe ser posible editar pequeñas regiones de superficie sin tener que reconstruir toda la superficie de la imagen.
- La presentación total de la superficie, debe de ser posible de mostrar los detalles más comunes de la imagen.
- La malla resultante debe estar perfectamente definida sin imperfecciones entre triángulos o regiones, donde diminutos triángulos se elevan de la superficie.
- Los vértices deben ser compartidos con tal frecuencia como sea posible entre cada cubo vecino, y aprovechar al máximo el hardware para transformar artefactos y minimizar los requisitos de almacenamiento de la superficie resultante.
- La cantidad de memoria requerida, mientras se ejecuta la generación de la superficie de la imagen debe mantener a un mínimo de recurso computacional [43].

Dividir la imagen total en vóxeles de igual tamaño relativamente pequeños ayuda a alcanzar estos objetivos. Para que el nivel de detalle de análisis del volumen total del solido sea alto se dividen bloques de 16x16x16 vóxeles para su respectivo análisis. Cada bloque posee un eje de coordenadas sea la división que sea y estos son **x, y** y **z** desde uno hasta cualquier conjunto de unión de bloques [44].

Cuando generan el acoplamiento triangular para un bloque de volumen, debemos tener acceso a un volumen de conjunto 17 x 17 x 17 vóxeles para poder ejecutar el algoritmo modificado de "Marching Cube" en 16 x 16 x 16 cubos. Algunos de los vóxeles en la esquina pertenecientes a las células en los límites del bloque positivos son propiedad de bloques vecinos. Además, con el fin de calcular un campo del vector normal, es necesario acceso a un vóxel adicional en el anterior, logrado por cada vóxel utilizado por el bloque hasta que se pueda calcular las diferencias centrales en las direcciones de los tres ejes coordenados. En total, generando la malla de triángulos para cada bloque requiere acceso a un volumen de 19 x 19 x 19 vóxeles, donde una capa de vóxeles precede los límites negativos del bloque y dos capas de vóxeles el límite positivo del bloque.

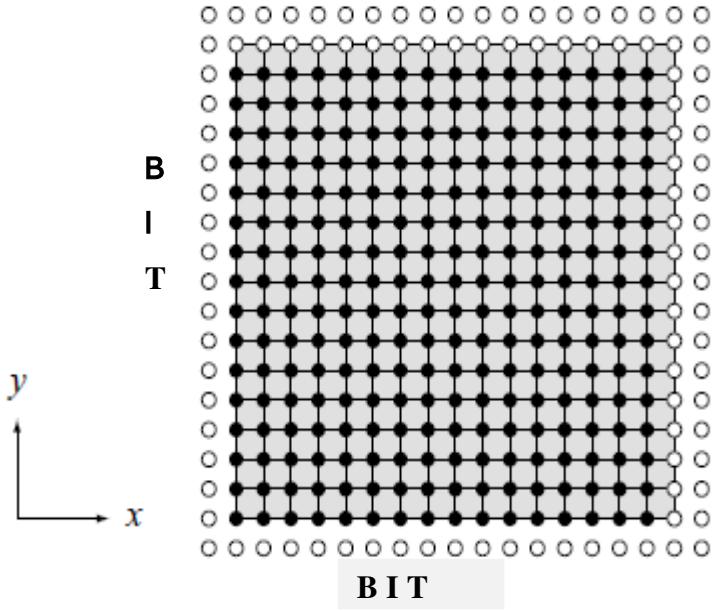


Figura N° 2. 7: *Esta es una porción horizontal del conjunto tridimensional de vóxeles que es necesario el acceso al generar la malla triangular para un solo bloque de cubos de 16x16x16. El área sombreada corresponde al volumen ocupado por el bloque, y los puntos solidos corresponden a los vóxeles que son propiedad del bloque. Puntos abiertos representan vóxeles de bloques vecinos[42].*

La Figura N° 2.7 es una muestra bidimensional del comportamiento de la separación de los volúmenes mencionados.

En primer lugar en la generación de mallas la triangularización para un bloque de volúmenes necesitamos descomprimir el conjunto de bloques para tener acceso a los datos de los vóxeles. En general tenemos que descomponer un acceso 27 bloques para obtener un pedazo de la superficie clara de la imagen, que contienen todos los datos requeridos para esto. Si la triangularización de datos se encuentra a lo largo del conjunto de datos, se descomprime menos bloques, y valores de vóxeles se duplican a lo largo de la frontera según sea necesario[45].

Una vez que los datos de los vóxeles están disponibles en la memoria analizamos cada una de los 4096 cubos en el bloque total y ejecutar el algoritmo de "Marching Cube" modificado. Comenzamos con el cubo en las coordenadas (0, 0, 0) y luego examinamos una a una la coordenada **x** a lo largo de la fila. Después de que cada cubo en **x,** ha sido examinado regresando a 0 la coordenada **x,** hacemos lo mismo con eje **y.** Después de que cada fila ha sido examinado, que incrementa la coordenada **y** al reiniciar la coordenada **x** desde 0. Esto continua hasta una

cubierta total de **16x16** se le coloca una coordenada **z** constante, y luego aumentamos la variable (0, 0, z) para conseguir la construcción de la imagen. Es importante que examinemos los cubos en este orden desde el vértice (0, 0,0) así saber que datos están disponibles para la construcción 3D [46].

Para poder enumerar y examinar la esquina de un cubo y posteriormente todos los cubos se coloca un eje de coordenadas como se muestra en la Figura N° 2.8. Las ocho esquinas pueden ser clasificadas como dentro o fuera del volumen del solido mediante un examen de bits. Si el bit de signo es cero, entonces el vóxel se encuentra fuera, y si los bits de signo en uno, el vóxel se encuentra dentro. (Por tanto, un valor de muestra de cero siempre es clasificado como fuera). Los bits de la señal se concentran en una sola cantidad de 8 bits para el cual el bits menos significativo en el vértice como 0, y el vértice más significativo es como 7. Esta cantidad proporciona el índice de casos del cubo, utilizado para obtener el índice de la clase de equivalencia de una entrada de 256 casos de tabla.

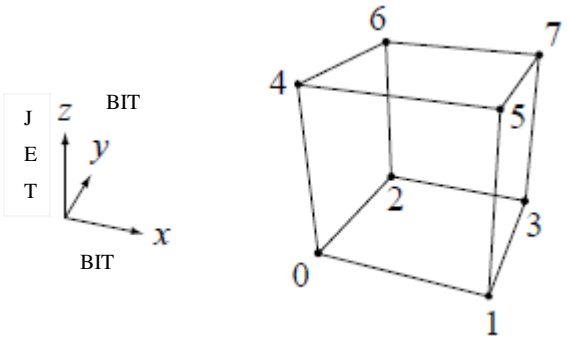


Figura N° 2. 8: *Cubo con sus respectivas coordenadas de análisis[47].*

Con el fin de maximizar el intercambio de vértice en el triángulo de malla generada por el bloque, tenemos que ser capaces de reutilizar vértices que antes eran fuera de la imagen. Esto se lograra mediante la limitación de los lugares en los que un nuevo vértice se puede crear para cualquier cubo y la reutilización de un vértice de un cubo previamente triangulado donde no se permite la creación de un nuevo vértice. Como se muestra en la Figura N° 2.9.(a), que solo permite nuevos vértices que se crean en la esquina máxima y bordes máximos de un conjunto de cubos, donde la esquina máxima es la que tiene los mayores coordenadas **x,y,z**, y una esquina máxima en un punto máximo como criterio de evaluación [47].

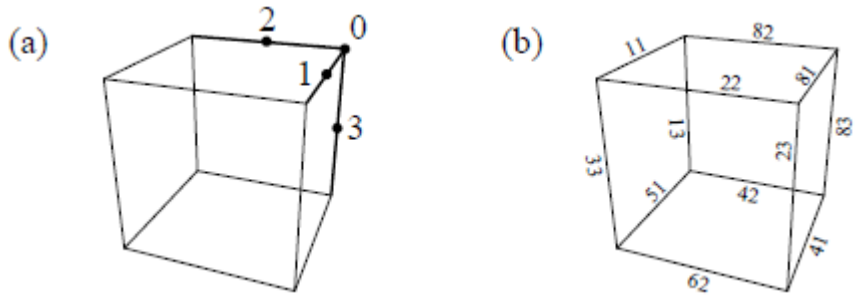


Figura N° 2. 9: *Para la mayoría de los cubos en un bloque, los nuevos vértices solo se pueden crear en la esquina máxima y tres bordes máximos (a) Los lugares en los que se permiten nuevos vértice están numerados de 0 a 3 como se muestra (b) Se le asigna un código hexadecimal de 8 bits a cada borde para proporcionar una asignación a un vértice reutilizable que pertenece a un cubo anterior[43]*

Cada borde de un cubo se le asigna un código de 8 bits, como se muestra en la Figura N° 2.9. (b), que proporciona un cubo anterior y el borde coincide en el cubo precedente para el que se le permitió nueva creación de vértice. Una parte de este código se indica en qué dirección ir para llegar al cubo anterior correcto para la construcción de la imagen. El bits de los valores 1,2 y 4 en este porción de imagen indica que hay que restar una de las coordenadas **x,y** o **z** respectivamente. Estos bits se pueden combinar para indicar que el cubo anterior es diagonalmente adyacente a través de un borde u otro lado de la esquina menor. El valor de 8 bit indica que un nuevo vértice se va a crear para el cubo tratado. El volumen menor de 8 bit es el código que da el índice del vértice en el cubo anterior que deben ser reutilizados o el índice del vértice en el cubo actual que deber ser creado [48].

En caso de que el valor del pixel de la muestra en una esquina seas exactamente cero, el vértice pegado en cualquier borde activo adyacente a la esquina se coloca precisamente en dicha ubicación. En una esquina en el que se permite un nuevo vértice se crea la esquina 7, por lo que los vértices de las otras esquinas deben ser reutilizados a partir de cubos anteriores. Un código de dirección de 3 bits que conduce un cubo apropiado se puede obtener fácilmente mediante la inversión del índice de esquina de 3 bits, donde el bit que son los valores de 1,2 y 4 indican que hay que restar una de las coordenadas **x,y** ó **z** respectivamente[49].

Para cada cubo, debemos tener espacio para almacenar cuatro índices de vértices correspondientes a las ubicaciones como se muestra en la Figura N° 2.9. (a) de modo que pueden ser reutilizados por el cubo triangularizado en un momento posterior. Nunca es el caso de que las cuatro ranuras de vértice se utilizan a la vez (porque un vértice dado en el interior de un borde no implica ningún vértice en la esquina), pero hay que ser capaz de identificar los vértices utilizando un fijo esquema de ubicación. Los vértices usados por un cubo siempre son propiedad del cubo, del cubo precedente, en la misma fila, uno de los dos cubos adyacentes en la fila precedente, o uno de los cuatro cubos en la cubierta anterior. Por tanto, podemos limitar los anteriores vértices que almacenamos al procesar un bloque de dos cubiertas que contienen **16x16** cubos y comparten entre ellos en dirección **z.**[45].

Para cada borde en el que tenemos que generar un nuevo vértice, calculamos un punto fijo de interpolación de parámetros **t** utilizando los valores de muestra d_0 y $d_1$en los extremos de borde:

$$t = \frac{d_1}{d_1 - d_0} \qquad (2.4)$$

Es extremadamente importante que este parámetro se calcula utilizando una ordenación coherente de los puntos finales, o de unión van a aparecer entre células adyacentes que pertenecen a diferentes conjunto de cubos. Siempre nos asociamos la d_0 distancia con el punto final con numeración más baja y la distancia d_1 con el punto más alto (ver Figura N° 2.8). Llamar a los puntos P_0 y P_1 posiciones de punto inicial y final, la Q posición de un nuevo vértice esta dado:

$$Q = tP_0 + (1 - t)\, P_1 \qquad (2.5)$$

Usamos una fracción de 8 bits, permitiendo que el nuevo vértice que se encuentra en uno de los 257 posibles posiciones a lo largo del borde cuando se incluyen los dos puntos finales. Si el vértice es coincidente con uno de los puntos finales, a continuación, los bits de fracción son todos ceros. Si se detecta esta condición, a continuación, separamos las ramas de ejecución que siguen porque el intercambio de vértices en el interior puede ser falso [30].

2.6.-CONSTRUCCION DE LA SUPERFICIE DE MALLA

Niveles más bajos de detalle son creados por muestreo de los datos volumétricos con una resolución media, la resolución de un cubo y así sucesivamente. Con el fin de evitar que la

cantidad de datos superficiales generados por cada conjunto de cubos debe disminuir exponencialmente, no triangulamos bloques de datos que tienen **8x8x8** cubos, **4x4x4** cubos, etc. En cambio, doblamos el tamaño físico del bloque cada vez que la resolución se reduce a la mitad para que los bloques siempre miden **16x16x16** cubos en la resolución de las que son triangulados[49].

2.6.1.-LAS TRIANGULACIONES ALTERNAS

Aunque no es particularmente importante para el más alto nivel de detalle, la ubicación especifica de los bordes internos puede hacer una diferencia significativa cuando la triangularización de los cubos de los niveles más bajos de detalle. Para la mayoría de las clases de equivalencia utilizados por nuestro algoritmo de "Marching Cubes" modificado, la triangulación buena que podría generarse cuando se presentan casos que pertenecen a esas clases de equivalencia. La triangularización para cada clases se compone de uno a cuatro componentes conectados que son topológicamente equivalente a polígonos convexos que tienen entre tres y siete vértices interiores [47].

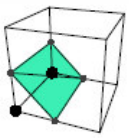
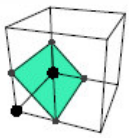
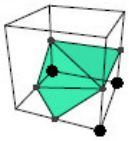
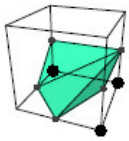
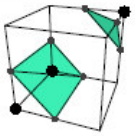
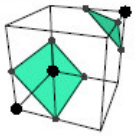
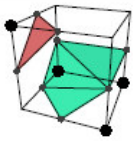
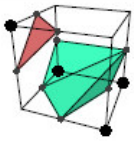
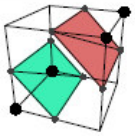
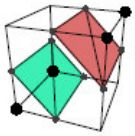
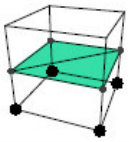
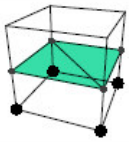
En el caso de que un componente de triangularización tiene cuatro vértices, es evidente que hay dos formas posibles de conectar vértices para formar una triangulación. Como el número de vértices aumenta, sin embargo, el número de posibles triangulaciones crece muy rápidamente. En general, el número de formas de triangulación de un polígono convexo que tiene **n** vértices está dada por el número de C_{n-2} donde:

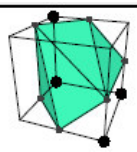
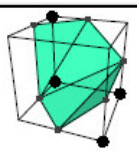
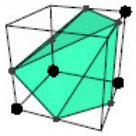
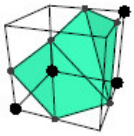
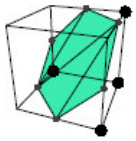
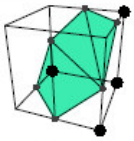
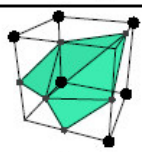
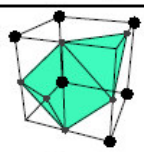
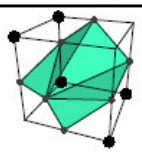
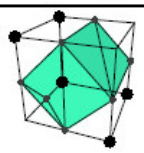
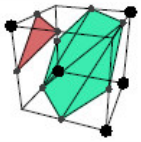
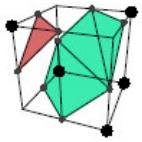
$$C_n = \frac{1}{n+1}\binom{2n}{n} \qquad (2.6)$$

Los valores de C_n para valores n=1,2,3,4,5 son 1,2,5,14 y 42, correspondientes al número posible de triangulaciones de polígonos que tiene 3,4,5,6 y 7 vértices [41].

Hay dos clases de equivalencia de los cuales los 5 son posibles triangulaciones, cinco clases para que 14 triangulaciones son posibles, y una clase para la cual 42 triangulaciones son posibles. Por componentes que tienen cinco o más vértices elegimos dos triangulaciones sean tan diferentes como sean posible en términos de sus direcciones principales de curvatura Tabla N° 2.3 enumera los posibles pares de triangulaciones que probamos por las 12 clases de equivalencia.

Tabla N° 2. 3: *Estas son, y las 12 clases de equivalencia para el que múltiples triangulaciones son posibles. Las dos triangulaciones que se muestran son los únicos que nos prueba para una mejor división a la alta resolución de datos de voxeles, son elegidos para ser tan diferentes como sea posibles en términos de sus direcciones principales de curvatura. La última columna muestra como son posibles muchas triangulaciones totales para cada clase de equivalencia[42].*

N°	ATRÄS	DELANTE	CLASE
#3			$C_2 = 2$
#5			$C_3 = 5$
#6			$C_2 = 2$
#8			$C_3 = 5$
#9			$C_2^2 = 4$
#11			$C_2 = 2$

N°	ATRÄS	DELANTE	CLASE
#12			$C_4 = 14$
#13			$C_4 = 14$
#14			$C_4 = 14$
#15			$C_4 = 14$
#16			$C_5 = 42$
#17			$C_4 = 14$

Dado que los niveles más bajos de detalle normalmente están procesados solo cuando la intensidad de pixel es baja, que la triangulación asignamos a cualquier cubo en particular solo hace una diferencia sutil que puede ser observado directamente en la malla de la superficie. Sin embargo, hay dos formas indirectas en las que la capacidad de elegir entre triangulaciones hace una diferencia importante. La primera forma consiste en aumentar el nivel de detalle a unos conjuntos de cubos en la superficie. Hay un "salto" perceptible que se produce en la malla cuando el nivel de detalle se cambia abruptamente, pero la magnitud visible de esta transición se reduce cuando la baja resolución de la triangulación es mejor división para la superficie de alta resolución. La segunda manera consiste en la aparición de sombreado. Un artificio muy

notable puede aparecer muy notable puede aparecer cuando se hace una mala elección de triangulación porque puede crear una concavidad que produce una sombra no deseada cuando se alinean de una manera desfavorable a la visión de la superficie. Elegir la mejor triangulación de la Tabla N° 2.3 ayuda a eliminar estos defectos mediante la creación de una malla de superficie de baja resolución más suave [47].

2.7.-TRANSICIÓN DE CUBOS

La presentación de la superficie de diferentes resoluciones una al lado de otro produce grietas a lo largo de los límites de malla, mostrado en la Figura N° 2.10. Estas grietas pueden ser bastante grandes, a pesar del hecho de que los vértices a lo largo de la frontera en bajo nivel de detalle de la malla siempre coinciden con los vértices de la malla de alto detalle. Las soluciones más fuertes a este problema se llaman técnicas de “stitching”, y trabajan rellenando las grietas con los triángulos adicionales que se unen a la perfección; los triángulos fronterizos de las mallas adyacentes. Las técnicas de costura “stitching” se implementan fácilmente en la superficie de la imagen basada en voxeles es mucho más complicado debido a la libertad topológica permitiendo los datos volumétricos subyacentes. Como ejemplo, un agujero gigante aparece en la Figura N° 2.10 donde una superficie de baja resolución es aproximadamente a la frontera plana de un bloque de alta resolución, y esto crea una topología que no puede existir en una malla de superficie basado en la altura[46].

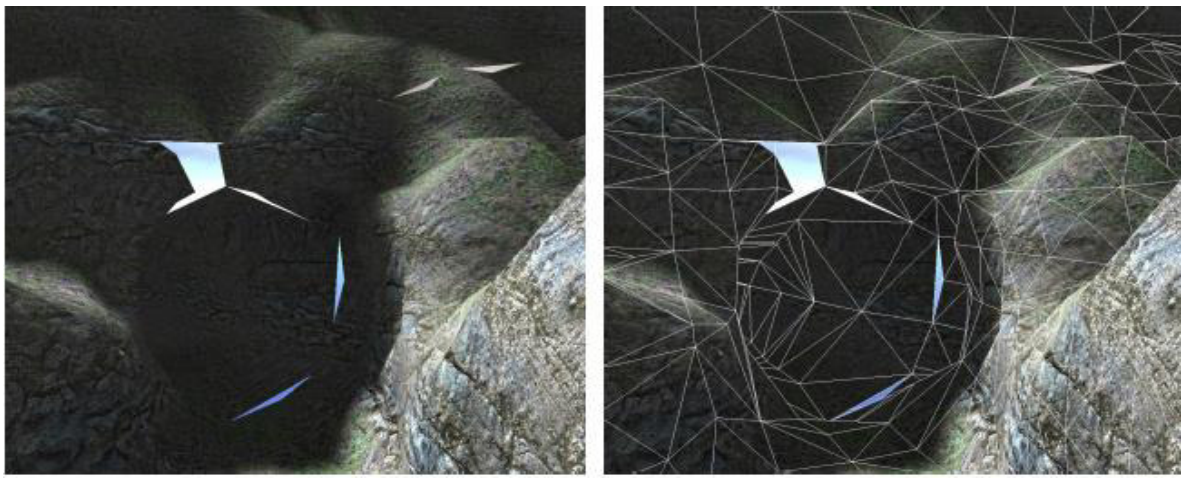


Figura N° 2. 10: *Bloques de superficie adyacentes cuyas mallas se representan en dos diferentes resoluciones con triangulación o sin ella. Las grietas son visibles a lo largo de la frontera entre los bloques permitiendo deformaciones tangentes a la superficie del conjunto de bloques[46]..*

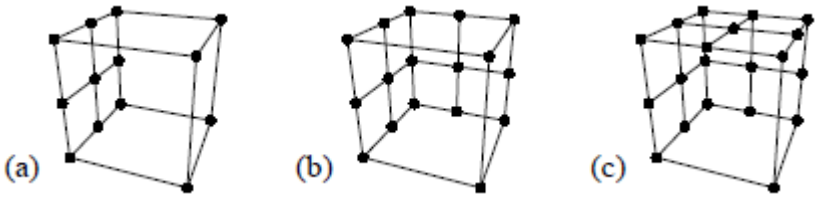


Figura N° 2. 11: *Los puntos que se muestran en los límites de estos cubos ilustran las posiciones de muestra para una cara de cubo en un bloque de media resolución cuando está bordeado por un bloque de resolución completa en (a) por un lado (b) dos lados y (c) tres lados[43].*

Consideremos dos bloques adyacentes de datos de vóxel de tal manera que un bloque se muestra a la mitad de la resolución del bloque. Llamamos a una cara dentro de un bloque de media resolución que se encuentra con la máxima resolución que se encuentra a lo de la frontera con la máxima resolución bloqueamos una cara de transición, y vemos que una triangulación compatible con el algoritmo de "Marching Cube" para cualquier cubo debe tener un total de 13 valores de la muestra pegado en la frontera. Como se muestra en la Figura N° 2.11. (a), nueve de estas muestras proceden de los datos de la resolución completa, y los cuatro restantes proceden de los datos de media resolución. La clasificación de cada uno de estos valores de la muestra como en el interior o el exterior del espacio solido produce $2^{13} = 8192$ posibles casos de triangulación. Esto es bastante grande en comparación a los 256 casos que se presentan en una implementación "Marching Cube" para una sola resolución, pero no hay razón para creer que un algoritmo análogo usando una tabla de consulta de 13 bits no produce un resultado satisfactorio y bueno[46]..

Si tenemos en cuenta, una sola cara del bloque de media resolución que esta bordeada por resolución completa de cubo en dos caras adyacentes, a continuación una triangulación de esa cara debe tener en cuenta los valores de muestra en la Figura N°2.11.(b), dando lugar a $2^{17} = 131072$ distintos casos. Por último, si una celda de media resolución es confinada con tres caras adyacentes por los cubos de resolución completa, una triangulación debe dar cuenta de los 20 valores de muestra dando lugar al conjunto de $2^{20} = 1048576$ distintos casos. Para implementar un algoritmo que conectara todos los casos de volumen de media triangulación, todos los casos que surjan dentro de los cubos de transición a lo largo de la frontera entre ellos, tendríamos que representar todos por $2^{13} + 2^{17} + 2^{20} = 1187840$ configuraciones[35].

Desde una perspectiva de la física informática determinamos el conjunto de clases de equivalencia topológica para los casi 1.2 millones de casos y luego generar una búsqueda de la tabla que contiene datos de triangulación para todos los casos de triangulación que es un trabajo monumental tedioso. Además, incluso si tal tabla de búsqueda podría construirse fácilmente, requería docenas de megabyte de espacio de almacenamiento. Este tamaño por sí mismo podría ser infinitamente grande en algunas plataformas, pero un asunto más importante es el elevado número de datos en la memoria que experimenta un algoritmo usando esta tabla de consulta sobre cualquier plataforma, afectará gravemente al rendimiento aquí estamos obligados a buscar alguna forma de simplificación del problema fundamental dado[49].

Idealmente, nos gustaría que el número de casos de triangulación distintas para una cara de transición en el mismo orden de magnitud como el número de casos que se presentan en los cubos convenientes sucesivos y que el número de casos posibles debería ser el mismo para toda la transición de cubos. Dos de estas propiedades pueden lograrse dividiendo cada celda de transición en dos cubos más pequeños de la manera ilustrada en la Figura N° 2.12. La celda izquierda se triangula utilizando valores de muestra solo de caras que bordean el bloque de resolución completa, y los valores de cuatro esquinas con la señal A,B,C y D en la figura se duplican en la cara opuesta del cubo. La transición de la resolución completa mitad de la resolución se lleva a cabo enteramente dentro de este cubo, para lo cual tenemos ahora unos valores muy manejables nueve muestras distintas a considerar. La celda derecha se triangularía de manera convencional usando los valores de los datos de resolución media [24].

Referimos el significado de transición para ser una de la forma mostrada en la Figura N° 2.12 que está influenciada por nueve vóxeles vacíos en una sola cara. En un cubo de transición, que llamamos la cara en la que nueve POA valores de las muestra que aparecen en la cara de resolución completa. La cara opuesta a la cara de resolución completa se denomina la cara de media resolución, y los valores de muestra en sus cuatro esquinas son idénticos a la de las esquinas correspondientes de la cara de resolución completa [44].

El ancho de un cubo de transición (es decir, la distancia entre la resolución completa y resolución media de caras) es una parámetro que se puede ajustar libremente en la superficie de la imagen.

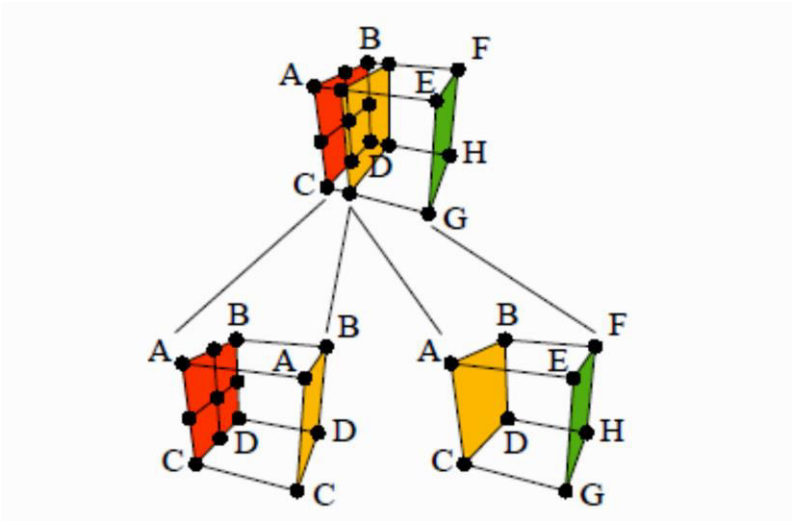


Figura N° 2. 12: *Un cubo de transición se divide en dos partes a lo largo de un plano paralelo a la cara de límite entre bloque de máxima resolución y resolución media. Nuestro nuevo algoritmo triangula la parte izquierda utilizando valores de muestra que solo aparecen en la cara de resolución completa con nueve POA. La parte derecha se triangula con el algoritmo de "Marching Cube" modificado convencional [40]..*

Los cubos de transición siempre tienen la misma configuración con respecto al número y ubicación de nuestros voxeles, y esto es cierto incluso cuando un bloque de media resolución esta bordeado por full resolución de bloques de dos o tres partes.

El problema de triangulación de cubos de transición se ha reducir ahora a aquel para cual hay 2^9=512 posibles casos de combinar. Una clase de equivalencia es la clase trivial que contiene los dos casos en los que el estado interior o exterior de los nueve valores de la muestra es la misma. Algunos casos no triviales se enumeran en la Tabla N° 2.4 a 2.5, en la que todos los casos que aparecen en una tabla en particular están relacionados de alguna manera. Para cada entrada de la tabla, el número en la esquina inferior izquierda es el índice de clase, y el número en la esquina inferior derecha es el número de casos que pertenecen a la clase de equivalencia. Un punto negro indica una esquina que se encuentra dentro del espacio sólido y esquinas sin un punto si están fuera. Triangulo verdes son de frente con respecto al punto de vista y triángulos rojos están de vuelta o atrás de las caras [41].

Tabla N° 2. 4: *Estas son las clases de equivalencia de 3 cubos de transición para que exactamente un valor de muestra está dentro del espacio sólido. Estas clases representan 18 los 512 casos distintos[45]*

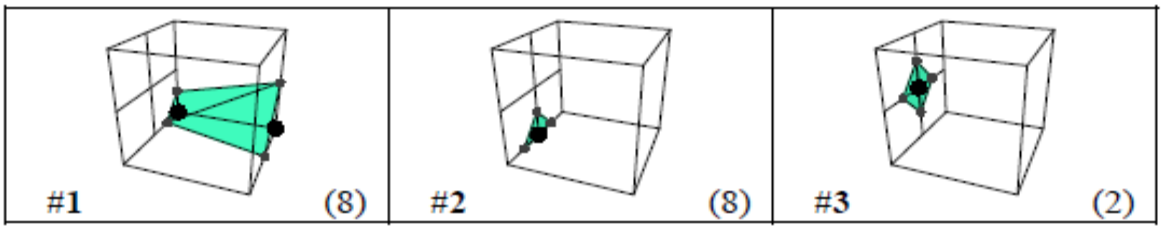
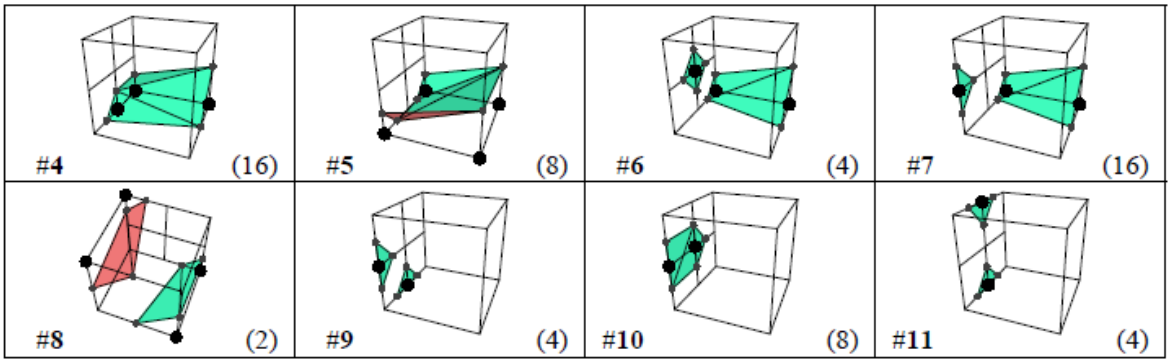


Tabla N° 2. 5: *Estás son las clases de equivalencias de 8 cubos de transición para que exactamente dos valores de muestra se encuentran dentro del espacio sólido. Estas clases representan 62 de los 512 casos distintos[45].*



2.8.-TRANSICIÓN DE CUBOS EN LA SUPERFICIE

Una gran superficie se compone de una jerarquía de medición de bloques 16x16x16 cubos, y estos bloques están dispuestos en un conjunto para que cada nivel corresponda a una resolución de mallas separadas. Cada bloque del más bajo nivel de detalle contiene exactamente ocho cubos de la siguiente resolución. Las posiciones de los vértices segundarios se pueden calcular mediante la transformación lineal de posiciones dentro del cubo de contorno de modo que los cubos de tamaño completo se escala a un tamaño más pequeño que permite que el espacio entre uno y tres cubos de transición, según sea necesario, dependiendo de la ubicación con respecto a los bordes y esquinas de todo un bloque. Esto se puede lograr para las compensaciones computacionales $(\Delta x, \Delta y, \Delta z)$ para las coordenadas (x, y, z) en cualquier cubo limite utilizando la formula [47].

$$\Delta x = \begin{cases} (1 - 2^{-k}x)w(k), & si \quad x < 2^k \\ 0 & si \quad 2^k \leq x \leq 2^k(s-1) \\ (s - 1 - 2^{-k}x)w(k), & si \quad x \leq 2^k(s-1) \end{cases} \tag{2.7}$$

Donde k es el índice de Nivel de Observación en Detalle (LOD), y s es el tamaño de un bloque en una dimensión cuando se mide en los cubos (por ejemplo un bloque 16x16x16 cubos de s=16). La función $w(k)$ específica el ancho de las transiciones de cubos de LOD que indica k y se define como $w(k) = 2^{k-2}$en nuestra aplicación. Para las variables de Δzy Δyse construyen con la ecuación (2.7).

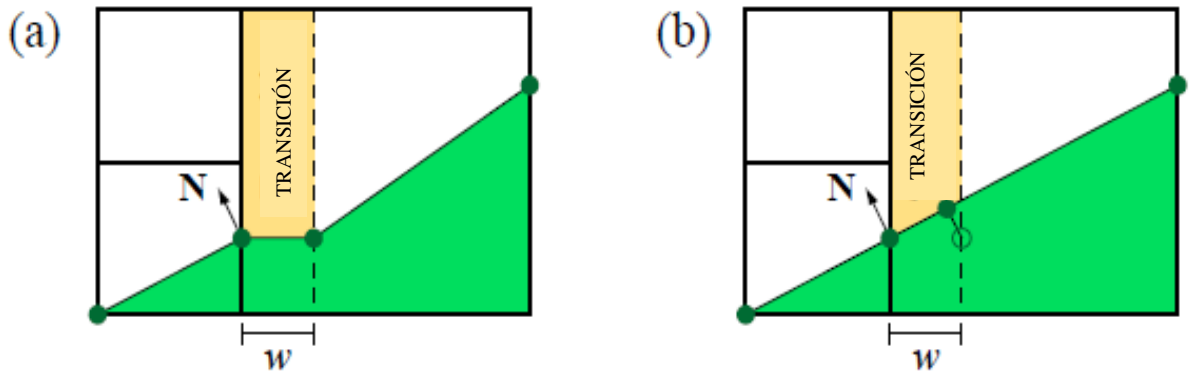
A lo largo de una cara del bloque en el que se está produciendo una transición entre niveles de detalle, aplicamos, la ecuación (2.7) para todos los vértices generados en la cara de media resolución de cada cubo de transición. Los vértices generados en las caras de máxima resolución de cubos de transiciones no se mueven. La adición de los desplazamientos directamente a las posiciones de los vértices originales tiene el efecto de crear un superficie aplanada de triángulos en la región de transición, como se muestra en la Figura N° 2.13 y se deforma la superficie en el cubo regular tal que concavidades no deseados se pueden crear. Estos problemas pueden se eliminados mediante la proyección del vector de desplazamiento para un vértice sobre el plano tangente que pasa a través de la posición original del vértice (x', y', z') usando la fórmula:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 1 - N_x^2 & -N_xN_y & -N_xN_z \\ -N_xN_y & 1 - N_y^2 & -N_yN_z \\ -N_xN_z & -N_yN_z & 1 - N_z^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \tag{2.8}$$

Donde **N** es el vértice de unidad de longitud normal. El efecto de la proyección dada por la ecuación (2.8) se muestra en la Figura N° 2.13. (b) [37].

El procedimiento de selección entre las posiciones de los vértices primarios y segundarios tiende a causar la posición de vértice principal para ser utilizado en las esquinas donde los bloques se encuentran y, más de un bloque se representa en el nivel inferior de detalle. La Figura N° 2.15 muestra ejemplos en el caso en el que cuatro bloques se encuentran y se prestan a diferentes niveles de detalle. En la Figura N° 2.14. (a) y (b) la posición de vértice principal

siempre se ha seleccionado en la ubicación de la esquina porque todos los bloques adyacentes



están siendo dispuestos en la resolución más alta. En la Figura N° 2.14. (c), se selecciona la posición de vértices secundaria ya que los tres bloques adyacentes se presentan con mayor resolución.

Figura N° 2. 13: *(a) la transformación lineal por la ecuación (2.7) se aplica a los vértices en la cara de baja resolución para hacer un espacio para un cubo de transición, pero esto tiene efectos no deseados la creación de una región plana y una concavidad. (b) Usando la ecuación (2.8) para proyectar la diferencia de los vértices sobre un plano tangente con respecto a la normal de vértice* ***N*** *elimina estos problemas[47].*

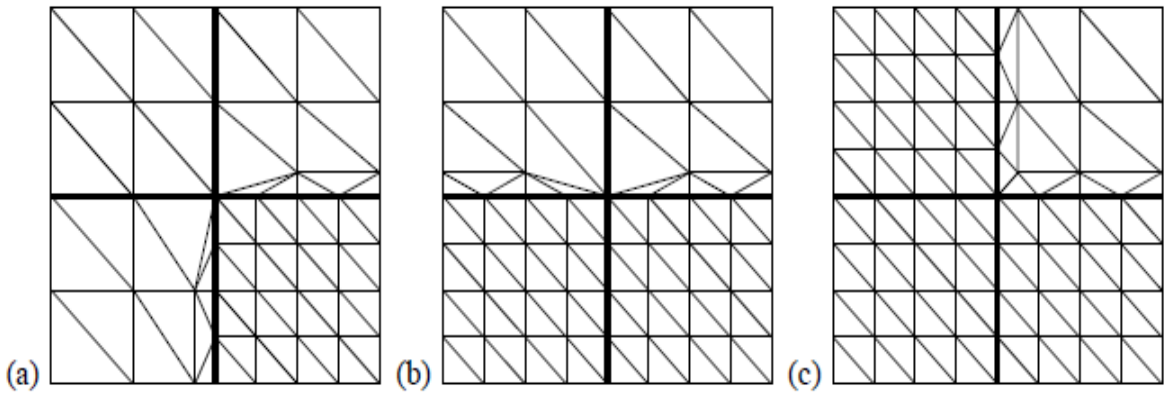


Figura N° 2. 14: *Estas son las tres transiciones posibles que puede tener lugar en la esquina donde cuatro bloques se encuentran y no están prestados en el mismo nivel de detalle. Todas las transiciones de cubos mostrados usan la equivalencia de clase #12 de la tabla 3.6. (a) Presentan tres bloques de baja resolución, y uno presenta de alta resolución. (b) Presentan dos bloques adyacentes de baja resolución y dos adyacentes de alta resolución proyectado. (c) Presentan un cuadrado de baja resolución y tres de alta resolución adyacente [43].*

2.9.- ORGANIZACIÓN DE TEJIDO

Por lo general, es necesario aplicar una variedad de diferentes intensidades a una malla amplia con el fin de lograr un resultado de deseado. Por ejemplo, pueden necesitar se prestados con diferentes partes de superficie de imagen la aparición de deformaciones y diferentes agentes externos de una imagen que aparecen y desaparecen a la perdida de intensidad de los pixeles. Adicionalmente a la capacidad de mezclar entre placas planas sobre la base de la proyección cubico en la dirección del vector normal, por lo general tienen la necesidad de mezclar entre dos o más placas de textura a lo largo de transición de un tipo de material a otro [42].

2.9.1.-TEXTURA DE PLANOS PARA DIFERENTES INTENSIDADES

Para hardware de gráficos que tienen capacidades de textura de matriz, que concluyen una rápida generación de imágenes como PlayStation que no tiene que recurrir a otro método de embalaje para múltiples planos de textura de un objeto. La textura tridimensional de planos no ofrece una solución viable, porque no hay manera de prevenir la filtración ente capas necesarias para un buen rendimiento. El único camino posible de acción en empalmar varias texturas en el plano en una más grande para construir la imagen[46]

En primer lugar, tenemos que ser capaces de extraer una muestra debidamente filtrada en la superficie de una sub-imagen ya que las texturas individuales a menudo se repiten a través de una malla de superficie. El hardware grafico típico solo puede realizar tales filtrados con envolturas adecuadas en las fronteras de todo el plano de textura, así que podemos confiar en el hardware para realizar el filtrado ajustado para cualquier sub-imagen. Estos nos dejan con las opciones siguientes:

- El filtrado de hardware se puede desactivar, y puede en lugar de realizar los cálculos de filtrado de manera explícita en la sombra de fragmento.
- Una porción de cada capa de textura puede ser repetido a lo largo de la frontera en los cuatro lados de cada sub-imagen envasado en mayor plano de textura.

La primera opción funciona correctamente, pero es lento, ya que refiere el cálculo manual de los parámetros de interpolación de la muestra y refiere ocho veces más capturas de textura como referiría normalmente para el filtrado trilineal. La segunda opción aprovecha al máximo las capacidades de filtrado de hardware, pero requiere más memoria para almacenar las

porciones repetidas de cada mapa de textura. Dado que el espacio de memoria es relativamente pequeño, elegimos la segunda solución para su mayor velocidad.

La Figura N° 2.15 muestra placas de textura de ejemplo que contiene nueve imágenes de **1024x1024** proyectados de una textura única de mapa de un arreglo de **3x3.** Alrededor de cada sub-imagen, una octava parte de la frontera la textura perteneciente a la sub-imagen se copia de los lados opuestos a producir una reproducción parcial [47].

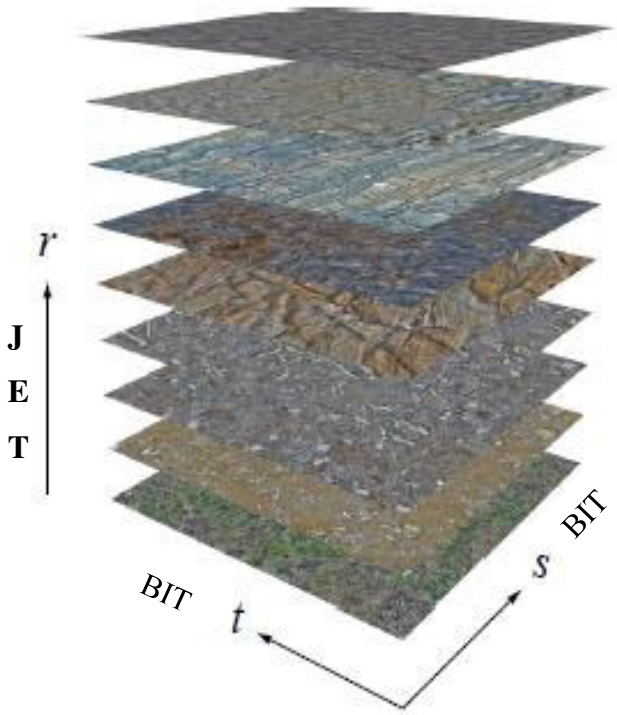


Figura N° 2. 15: *Hardware de gráficos modernos admite matrices de mallas de pixeles, que son placas de dos dimensiones de imágenes que se almacenan como una sola unidad de imagen direccionales, pero para el que cada imagen se filtra independiente a pérdidas de intensidad. La coordenada* ***r*** *se redondea al entero más cercano seleccionado en la textura de una imagen y los* ***s*** *y* ***t*** *coordenadas especificando la ubicación de muestra dentro de la imagen [47].*

CAPÍTULO III

ISOSUPERFICIES EXACTAS POR "MARCHING CUBE"

3.1.- ISOSUPERFICIES EXACTAS POR "MARCHING CUBE"

En este capítulo presentamos un nuevo método de parametrización de la imagen en las superficies de mallas triangulares del algoritmo de "Marching Cube" para nivelar a cero cierta triangulaciones de las imágenes que tendrían que aplicarse rodeando toda la imagen como si se tratara de un dominio esférico para cubrir toda la superficie de la imagen proyectada y nivelarla a cero.

En este capítulo estudiamos los exactos contornos exactos de un función definida a trozos en un campo escalar trilineal de cabeza. Mostramos como representan estos los contornos recortados exactamente como superficies de parches de cubos racionales triangulares. Como parte de esto, estudiamos una extensión del algoritmo de "Marching Cube" que da una aproximación triangular topológicamente exacta de los contornos para cualquier caso. Tratamos el algoritmo tanto teórico como practico como conjunto de datos.

La malla triangular resultante podría ser demasiado gruesa o demasiado fina. Si la malla es demasiado fina (es decir, el número de triángulos es demasiado alto), no existe problema en la superficie de la imagen pero esto no a ocurrido en nuestro trabajo y creo que nunca ocurre en una superficie proyectada [37]

El propósito de este capítulo es lidiar con ese problema propuesto, para obtener una mejor representación de contorno si la malla triangular de MC es demasiado gruesa. Este problema aparece:

- Con datos de volumen de baja resolución.
- Al explorar los detalles en los datos de volumen de alta resolución
- Con datos de volúmenes fractales

En la Figura N° 3.1.(a) muestra un ejemplo de un conjunto de datos de volumen donde MC da una malla triangular de esta clara Figura N° 3.1.(b) muestra un detalle de la parte interior de la superficie pero esta es demasiado gruesa.

Con el fin de mejorar la calidad de una superficie que muestra en la Figura N° 3.1. (b) las interpolaciones de orden más altas entre la red de puntos del campo escalar pueden ser aplicados. Este enfoque puede terminar en nuevas clases de las superficies que contienen auto interpolaciones y topologías complicadas Figura N° 3.1. (c) da una ilustración.

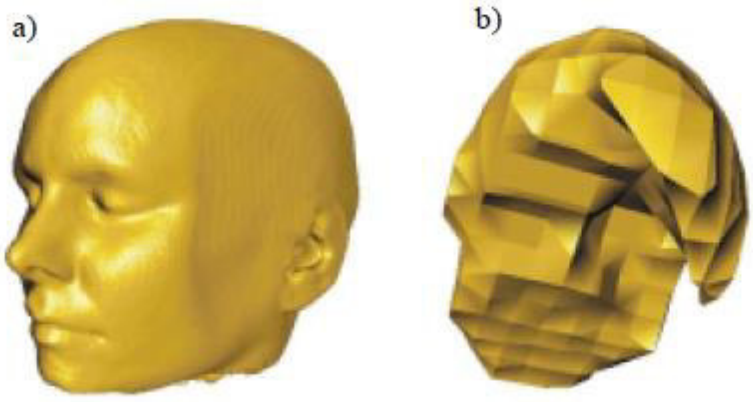


Figura N° 3. 1: (a) *Cabeza de CT formada consiste en 423963 triángulos un candidato al algoritmo de reducción de acoplamiento.* ***(b)*** *Detalles en el interior de la misma cabeza CT la malla triangular es demasiado gruesa[38].*

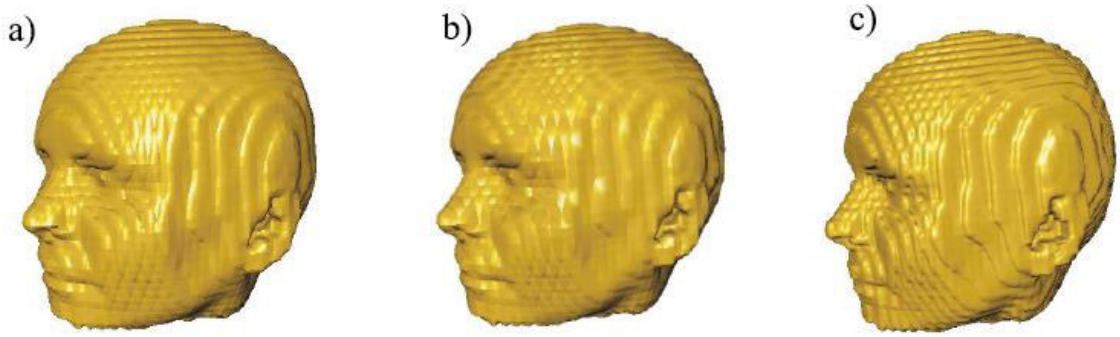


Figura N° 3. 2: *Ilustración de las ideas principales de este trabajo* ***(a)*** *Células y la aproximación triangular de un determinado contorno de dos pares interconectados usando MC* ***(b)*** *El contorno exacto representado por una serie de superficies recortadas de parches cúbicos triangulares* ***(c)*** *La modificación de G^1 a nivel mundial del contorno sin cambiar su topología [38].*

La superficie aproximada es G^0 continua a través de los límites de las celdas. También se utilizan parches cúbicos a trozos para refinar los resultados de un enfoque de "contorno y continuidad". En, el contorno se aproxima utilizando los parches con 4,5 o 6 curvas de contorno

se aproxima a los contornos de parches de superficies racionales cuadráticas triangulares en superficies

En este capítulo deducimos dos ideas. Primero nos encontramos con la representación exacta del contorno trilineal como una sola pieza de superficie paramétrica. Resulta que el contorno puede ser descrito por un número de superficies recortadas de porciones de cubos triangulares.[33].

3.2.-CONTORNO EXACTO DE UN CAMPO ESCALAR TRILINEAL

Un primer acercamiento es construir el contorno exacto de un campo escalar trilineal puede encontrarse en, donde es construido como una superficie de subdivisión. En esta sección nosotros introducimos una descripción paramétrica.

$$\begin{aligned} s(x,y,z) = {} & (1-x)(1-y)(1 \\ & -z)c_{000} \\ & +(1-x)(1-y)zc_{001} \\ & +(1-x)\,y(\,1-z)\,c_{010} + (1 \\ & -x)\,yzc_{011} \\ & +x(1-y)(1-z)c_{100} + x(1 \\ & -y)zc_{101} \\ & +xy(1-z)c_{110} + xyzc_{111} \end{aligned} \qquad (3.1)$$

Donde c_{ijk} (i, j, k ∈ {0, 1}) son los valores escalares en el vértices de un cubo. Un contorno está dado por un espacio ***r*** especificado; consta de todos los puntos $(x, y, z)^T$ transpuesta con:

$$s(x,y,z) = r \qquad (3.2)$$

Dando un punto al $a = (x_a, y_a, z_a)^T$. Queremos calcular la intersección de una línea en el contorno definida por las ecuaciones (3.1) y (3.2), porque tenemos que resolver una ecuación cubica [30].

Para el caso especial que la línea sea paralela a uno de los ejes de coordenadas, el problema se simplifica a la solución de una ecuación lineal: sea la intersección $p_x(a)$ de la línea con un punto $a + \lambda(1,0,0)^T$ el contorno definido por las ecuaciones (3.1) y (3.2). Además, dejó $p_y(a)$ ser la intersección de la línea $a + \lambda(0,1,0)^T$ con el contorno y dejar $p_z(a)$ ser el intersección de la línea $a + \lambda(0,0,1)^T$con el contorno. Entonces obtenemos de las ecuaciones (3.1) y (3.2):

$$p_x(a) = \left(\frac{r - c_{000}(1-y_a)(1-z_a) - c_{001}(1-y_a)z_a - c_{010}y_a(1-z_a) - c_{011}y_a z_a}{(c_{100}-c_{000})(1-y_a)(1-z_a) + (c_{101}-c_{001})(1-y_a)z_a + (c_{110}-c_{010})y_a(1-z_a) + (c_{111}-c_{011})y_a z_a}, y_a, z_a\right)^T$$

$$p_y(a) = \left(x_a, \frac{r - c_{000}(1-x_a)(1-z_a) - c_{100}(1-z_a)x_a - c_{001}z_a(1-x_a) - c_{101}x_a z_a}{(c_{010}-c_{000})(1-x_a)(1-z_a) + (c_{110}-c_{100})(1-z_a)x_a + (c_{011}-c_{001})z_a(1-x_a) + (c_{111}-c_{101})x_a z_a}, z_a\right)^T$$

$$p_z(a)$$

$$= \left(x_a, y_a \frac{r - c_{000}(1-x_a)(1-y_a) - c_{010}(1-x_a)y_a - c_{100}x_a(1-y_a) - c_{110}x_a y_a}{(c_{001}-c_{000})(1-x_a)(1-y_a) + (c_{011}-c_{010})(1-x_a)y_a + (c_{101}-c_{100})x_a(1-y_a) + (c_{111}-c\right.$$

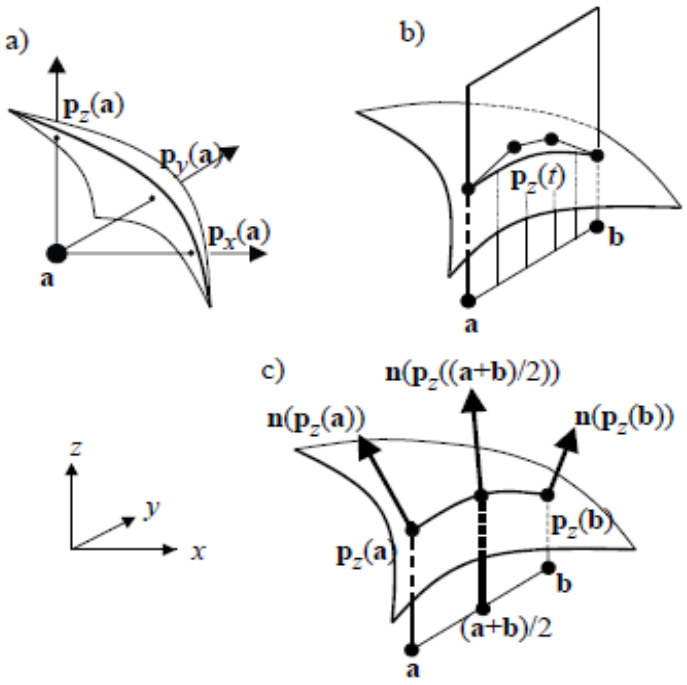


Figura N° 3. 3*: Las imagen nos muestra (a) proyecciones* ***$P_x(a)$, $P_y(a)$, $P_z(a)$*** *de un punto de un sobre la dirección del contorno en dirección x-,y- y z- (b) proyección* ***$P_z(t)$*** *de la línea segmento* ***(1-t)a+tb*** *sobre el contorno es una curva cúbica racional. (c) configuración para calcular los puntos de control de* ***$P_z(t)$*** *[30].*

Llamamos $p_x(a), p_y(a), p_z(a)$, que son las proyecciones **a** sobre el contorno en direcciones $x-, y-$ y $z-$. Esto significa que podemos construir tres puntos en el contorno para un punto dado en una forma sencilla. La Figura N° 3.3 da una ilustración.

Dado un punto ***a***, allí es solamente un contorno definido por la ecuación (3.1) a través de él. Podemos calcular su normal con el vector ***n(a)*** en un por:

$$n(a) \quad (3.3)$$
$$= (s_x(x_a, y_a, z_a), s_y(x_a, y_a, z_a), s_z(x_a, y_a, z_a))^T$$

Donde s_x, s_y, s_zson las derivadas parciales de s definido en la ecuación (3.1). Ahora construir curvas en el contorno mediante la proyección de segmentos de línea en la misma. En el segmento de línea $x(t) = (1-t)a + tb$. Entonces las curvas de:

$$p_x(t) = p_x(x(t)), p_y(t) \quad (3.4)$$
$$= p_y(x(t)), p_z(t)$$
$$= p_z(x(t))$$

Se obtienen mediante la proyección de cada punto de $x(t)$ en el contorno en dirección $x-, y-$ ó $z-$. La Figura N° 3.3 (b) da una ilustración para él $p_z(t)$.

Así con un proceso de algebra para demostrar que las curvas de $p_x(t), p_y(t), p_z(t)$ en el contorno definido por las ecuaciones (3.1) y (3.2) son cúbicas y racionales. Así la curva $p_z(t)$ puede expresarse como:

$$p_z(t) = \frac{\sum_{i=0}^{3} w_i b_i B_i^3(t)}{\sum_{i=0}^{3} w_i B_i^3(t)} \quad (3.5)$$

Donde $B_i^3(t)$ son los polinomios de [31] y:

$$w_o = z_{n(p_z(a))}, \quad w_1 = \frac{4}{3} z_{n(p_z(\frac{a+b}{2}))} - \frac{1}{3} z_{n(p_z(b))} \quad (3.6)$$
$$w_3 = z_{n(p_z(a))}, \quad w_2 = \frac{4}{3} z_{n(p_z(\frac{a+b}{2}))} - \frac{1}{3} z_{n(p_z(a))}$$
$$b_o = p_z(a) \quad b_3 = p_z(b)$$

Tenga en cuenta que en la ecuación (3.6) los valores $w_0,,,,,,,,w_3$ son definidos por las coordenadas $z-$ de ciertos vectores normales del contorno. Para ejemplo, $w_0$es definido por el componente $z-$del contorno normal en $p_z(a)$. La Figura N° 3.3 (c) da una ilustración de las componentes utilizadas en la ecuación (3.6). Las curvas de $p_x(t), p_y(t)$ puede ser calculado como racionales cúbicos de una manera similar.

Ahora extendemos el concepto de curvas de contorno a superficies paramétricas en el contorno definido por las ecuaciones (3.1) y (3.2). Da un triángulo.

$$x(\mu, \nu, \omega) = \mu a + \nu b + \omega c \qquad (3.7)$$

Las coordenadas del baricentro son los vértices a, b, c, planteamos $\mu + \nu + \omega = 1$. Entonces $p_x(x(\mu, \nu, \omega)), p_y(x(\mu, \nu, \omega)), p_z(x(\mu, \nu, \omega))$ son las proyecciones de **x** sobre el contorno definido por las ecuaciones (3.1) y (3.2). La Figura N° 3.4 (a) da una ilustración de $p_z(x)$.

Las superficies de $p_x(x(\mu, \nu, \omega)), p_y(x(\mu, \nu, \omega)), p_z(x(\mu, \nu, \omega))$ son cúbicas de racionales. Por ejemplo, puede ser $p_z(x(\mu, \nu, \omega))$ descrito como un triángulo racional de Bézier.

$$p_z(x(\mu, \nu, \omega)) = \frac{\sum_{i+j+k=3} \omega_{ijk} b_{ijk} B_{ijk}^3(\mu, \nu, \omega)}{\sum_{i+j+k=3} \omega_{ijk} B_{ijk}^3(\mu, \nu, \omega)} \qquad (3.8)$$

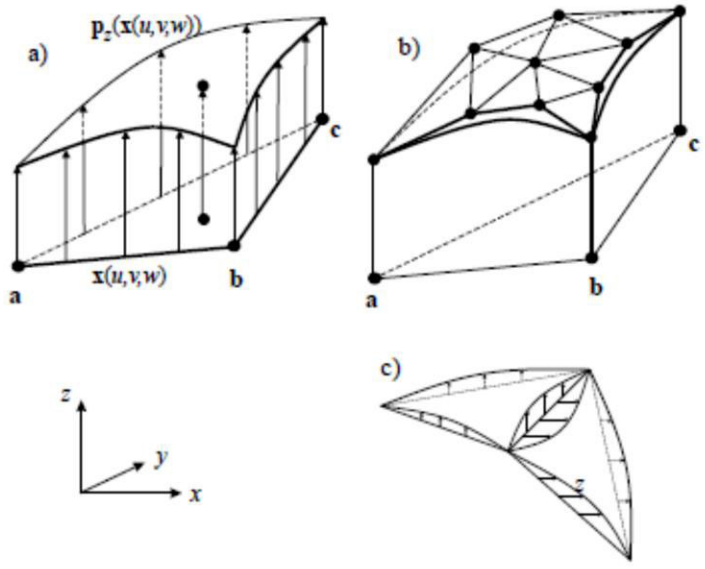


Figura N° 3. 4: *En la ilustración (a) p_z (x (u, v, w)) se obtiene mediante la proyección de cada punto de x (u, v, w) = ua+ w b +v c en dirección z- en el contorno definido por las ecuaciones (3.1) y (3.2). (b) p_z (x (u, v, w)) es un superficie cúbica racional. (c) dos triángulos adyacentes de MC proyectado en diferentes direcciones: los paquetes de contornos resultantes tienen diferentes gráficas [31].*

Donde los puntos de Bézier y sus valores en el límite de las curvas pueden ser calculados en la ecuación (3.6), y representados en otro fractal como:

$$\omega_{111} = \frac{\omega_{201} + \omega_{102} + \omega_{021} + \omega_{012} + \omega_{120} + \omega_{210}}{4} + \frac{\omega_{300} + \omega_{030} + \omega_{003}}{6}$$

$$x_{b_{111}} = \left(\frac{\omega_{012} + \omega_{021}}{4\omega_{111}} - \frac{\omega_{030} + \omega_{003}}{12\omega_{111}}\right) x_a + \left(\frac{\omega_{102} + \omega_{201}}{4\omega_{111}} - \frac{\omega_{300} + \omega_{003}}{12\omega_{111}}\right) x_b + \left(\frac{\omega_{120} + \omega_{210}}{4\omega_{111}} - \frac{\omega_{300} + \omega_{030}}{12\omega_{111}}\right) x_c$$

$$y_{b_{111}} = \left(\frac{\omega_{012} + \omega_{021}}{4\omega_{111}} - \frac{\omega_{030} + \omega_{003}}{12\omega_{111}}\right) y_a + \left(\frac{\omega_{102} + \omega_{201}}{4\omega_{111}} - \frac{\omega_{300} + \omega_{003}}{12\omega_{111}}\right) y_b + \left(\frac{\omega_{120} + \omega_{210}}{4\omega_{111}} - \frac{\omega_{300} + \omega_{030}}{12\omega_{111}}\right) y_c$$

$$z_{b_{111}} = \frac{\omega_{210} z_{b210} + \omega_{201} z_{b210} + \omega_{120} z_{b120} + \omega_{021} z_{b021} + \omega_{012} z_{b012} + \omega_{102} z_{b102}}{4\omega_{111}} - \frac{\omega_{300} z_{p_z(a)} + \omega_{030} z_{p_z(b)} + \omega_{003} z_{p_z(c)}}{12\omega_{111}}$$

La Figura N° 3.4 da una ilustración. Las superficies de $p_x(x)\,, p_y(x)$ puede ser descrito de una manera similar.

La idea básica de que representa un contorno definido por las ecuaciones (3.1) y (3.2) es aplicar el algoritmo de MC y proyecto de cada uno de los triángulos resultantes en el contorno. Desafortunadamente, esto puede producir deficiencias de triángulos adyacentes si su proyección las direcciones son diferentes. La Figura N° 3.4 muestra un ejemplo. Para superar este problema, usamos las superficies cortadas de la triangular cúbicas de racionales en vez de las partes triangulares [32].

Es el triángulo ***(a,b,c)*** que se obtiene de algoritmo de MC. Estas mediciones (a, b, c)representan el contorno realizando:

1. Determinar las proyecciones en las direcciones, $q_{ab}, q_{bc}, q_{ca} \in \{x, y, z\}$en las curvas de límite.
2. Determinar la proyección en dirección q_{abc} del triángulo entero

3. Proyectamos los límites del triángulo (a, b, c)en las indicaciones definidas en el paso 1 alrededor del contorno. Nosotros obtenemos las curvas en el contorno. Las curvas x_{ab}, x_{bc}, x_{ca} son la curvas de límite de la revisión final sobre el triángulo (a, b, c).

$$x_{ab}(t) = p_{q_{ab}}((1 - t)\,a + tb)$$
$$x_{bc}(t) = p_{q_{bc}}((1 - t)\,b + tc)$$
$$x_{ca}(t) = p_{q_{ca}}((1 - t)\,c + ta)$$

4. Proyectamos x_{ab}, x_{bc}, x_{ca}en la dirección q_{abc} en el plano definido por a, b, c. Obtenemos las curvas de y_{ab}, y_{bc}, y_{ca}en el plano (a, b, c)que son la curvas de límite del dominio de la versión final.
5. Calculamos la superficie recortada de la revisión proyectada $p_{q_{abc}}(\mu\, a + \nu\, b + \omega\, c)$con $\mu + \nu + \omega = 1$. El dominio de la superficie recortada está dada por el límite de curvas y_{ab}, y_{bc}, y_{ca} [33].

3.3.-TOPOLOGÍA EXACTA DE MARCHING CUBE (MC)

Es el propósito de esta sección para encontrar un conjunto de puntos interiores que son suficientes para obtener una triangulación exacta topológicamente para cada caso. Nosotros nombramos los vértices del polígono $(v_1, \ldots\ldots, v_6)$. La triangulación de este polígono, son los bordes (v_2, v_6)y (v_3, v_5), que no deben utilizarse porque el contorno no tiene aristas entre estos vértices en la cara superior de la celda[36].

Consideramos que todos los puntos del contorno que tener una dirección normal, ya sea en ***(x,y,z).*** Dado un contorno definido por las ecuaciones (3.1) y (3.2), hay por lo más dos puntos x_0, x_1 en el contorno con un normal en dirección ***x.***

Tales vectores estarán ubicados en las siguientes ecuaciones de contorno usadas para el cálculo computacional en la superficie dada.

$$x_m = \frac{(c_{111}-c_{011})(r-c_{000}) - (c_{110}-c_{010}) \times (r-c_{001}) + (c_{100}-c_{000})(r-c_{011}) - (c_{101}-c_{001})(r-c_{010})}{2((c_{111}-c_{011})(c_{100}-c_{000}) - (c_{110}-c_{010})(c_{101}-c_{001}))}$$

$$y_m = \frac{(c_{111}-c_{101})(r-c_{000}) - (c_{110}-c_{100}) \times (r-c_{001}) + (c_{010}-c_{000})(r-c_{101}) - (c_{011}-c_{001})(r-c_{100})}{2((c_{111}-c_{101})(c_{010}-c_{000}) - (c_{110}-c_{100})(c_{011}-c_{001}))}$$

$$z_m = \frac{(c_{111}-c_{110})(r-c_{000}) - (c_{101}-c_{100}) \times (r-c_{010}) + (c_{001}-c_{000})(r-c_{110}) - (c_{011}-c_{010})(r-c_{100})}{2((c_{111}-c_{110})(c_{001}-c_{000}) - (c_{101}-c_{100})(c_{011}-c_{010}))}$$

Con los valores:

$$x_p = \frac{1}{2((c_{111}-c_{011})(c_{100}-c_{000}) - (c_{110}-c_{010})(c_{101}-c_{001}))}$$

$$y_p = \frac{1}{2((c_{111}-c_{101})(c_{100}-c_{000}) - (c_{110}-c_{100})(c_{011}-c_{001}))}$$

$$z_p = \frac{1}{2((c_{111}-c_{110})(c_{001}-c_{000}) - (c_{101}-c_{100})(c_{011}-c_{010}))}$$

Así:

$$d = ar^2 + br + c \qquad (3.10)$$

Con valores representativos de:

$$a = (-c_{111} + c_{110} + c_{101} - c_{100} - c_{001} + c_{000} + c_{011} - c_{010})^2$$

$$b = 2(c_{001}\,c_{110} + c_{110}\,c_{100} + c_{101}\,c_{010} + c_{111}\,c_{000})(c_{111} + c_{000} + c_{101} + c_{110} + c_{100} + c_{010} + c_{011} + c_{001})$$
$$c_{000}\,c_{111} \quad -4(c_{000}\,c_{111}(c_{111}+c_{000}) + c_{001}\,c_{110}(c_{110}+c_{001}) + c_{010}\,c_{101}(c_{101}+c_{010}) + c_{011}\,c_{100}(c_{100}+c_{011})$$
$$-4(c_{111}\,c_{110}\,c_{101} + c_{000}\,c_{011}\,c_{110} + c_{000}\,c_{011}\,c_{101} + c_{000}\,c_{110}\,c_{101}\,c_{011}) - 4(c_{111}\,c_{001}\,c_{010} + c_{110}\,c_{101}\,c_{001} + c_{010}\,c_{100}\,c_{111} + c_{001}\,c_{010}\,c_{100})$$

$$c = -(c_{001}\,c_{110} + c_{011}\,c_{100} + c_{111}\,c_{000})^2 + 2(c_{111}^2\,c_{000}^2 + c_{101}^2\,c_{010}^2 + c_{001}^2\,c_{110}^2 + c_{011}^2\,c_{100}^2)$$

Dependiendo del valor ***d***, todos estos seis puntos son o bien puntos reales en el contorno, o todos tienen valores imaginarios. Si son reales (es decir, si $d > 0$), entonces la ecuación (3.9) da que

el polígono cerrado $(x_0, y_1, z_0, x_1, y_0, z_1)$ se encuentra en los bordes de un voxel. Tenga en cuenta que este polígono cubre completamente el contorno de la figura proyectada [34].

Ahora podemos describir un algoritmo topológicamente exacta MC:

1. Creamos los polígonos que encierran la imagen total creada y suavizada. Nosotros obtener hasta tres polígonos cerrados y los llaman anillos exteriores.
2. Calcular los puntos del anillo interior mediante la aplicación de las ecuaciones (3.9) y (3.10).
3. Si el anillo interior no es real o si el anillo interior es completamente fuera de la la imagen, entonces triangulamos los anillos externos independientemente uno de otro; de lo contrario continuará con 4.
4. Compruebe la conectividad entre el anillo interior y cada uno de los anillos exteriores. Si el anillo interior y uno de los anillos exteriores pertenecen al mismo segmento de contorno, triangular la zona entre el anillo interior y el anillo exterior.

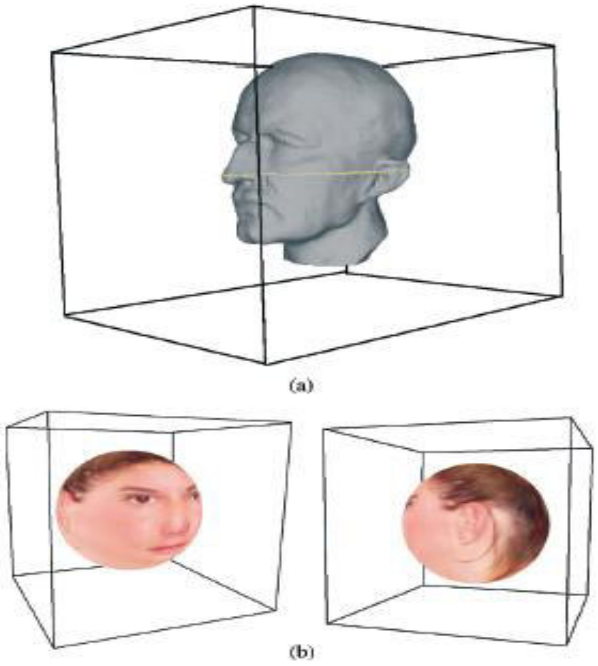


Figura N° 3. 6 *Las figuras nos dan a entender (a) Las isosuperficies representando un rostro, del lazo y de la parametrización inicial de la imagen (b) imagen utilizada como un mapa de textura [37]*

CAPÍTULO N° IV

RESULTADOS Y CONCLUSIONES

En este capítulo se explican algunas de las funciones que se han empleado para la implantación de los algoritmos en el programa MATLAB.

Si se desea más información del programa que no se trata en este capítulo se puede consultar el manual de MATLAB

4.1.- VISIÓN ARTIFICIAL

4.1.1.-OPERACIONES CON MATRICES

Para definir una matriz no hace falta establecer de antemano su tamaño. MATLAB determina el número de filas y de columnas en función del número de elementos que se proporcionan. Las matrices se definen por filas, los elementos de una misma fila están separados por espacios o comas, mientras que las filas están separadas por punto y coma (;).

Por ejemplo, el siguiente comando define una matriz A de dimensión (3x3):

>> A=[1 2 3; 4 5 6; 7 8 9]

La respuesta del programa es la siguiente:

A =

1 2 3

4 5 6

7 8 9

A partir de este momento la matriz A está disponible para hacer cualquier tipo de operación.

4.1.2.-OPERADORES

MATLAB puede operar con matrices por medio de operadores y por medio de funciones.

Los operadores matriciales de MATLAB son los siguientes:

+suma

Resta

Multiplicación

‘traspuesta

^ Potenciación

\ división-izquierda

/ división-derecha

.* Producto elemento a elemento

./ y **.** división elemento a elemento

.^ Elevar a una potencia elemento a elemento

4.1.3.- MATRICES PARTICULARES

También existen en MATLAB varias funciones orientadas a definir con gran facilidad matrices de tipos particulares. Algunas de estas funciones son las siguientes:

zeros (n,m), forma una matriz de **ceros** de tamaño (nxm) .

ones(n,m), forma una matriz de **unos** de tamaño (nxm).

4.1.4.-ACCESO A ELEMENTOS DE UNA MATRIZ

El operador: es muy importante en MATLAB y puede usarse de varias formas. Un ejemplo sería:

>> x=1:10

x =

1 2 3 4 5 6 7 8 910

El operador (:) representa un rango entre dos valores, en el caso del ejemplo sería entre 1 y 10. Por defecto el incremento es 1, pero este operador puede también utilizarse con otros valores enteros y reales, positivos o negativos.

MATLAB accede a los elementos de una matriz por medio de los índices de fila y de columna encerrados entre paréntesis y separados por una coma. Así en el ejemplo de la matriz A:

>> A(2,3)

ans =

6

Para poder acceder a más de un elemento de una matriz se emplea el operador (:).

A(n,:) devuelve la fila n entera.

A(:, m), devuelve la columna m entera.

4.2.-PROGRAMACIÓN EN MATLAB

A continuación se detallan algunos bucles, que han sido utilizados y que son de interés

Sentencia if

if condicion

sentencias

end

Para varias condiciones sería de la siguiente forma:

if condicion1

bloque1

elseif condicion2

bloque2

elseif condicion3

bloque3

else% opción por defecto para cuando no se cumplan las condiciones 1,2,3

bloque4

end

Sentencia for

La sentencia for repite un conjunto de sentencias un número predeterminado de veces.

La siguiente construcción ejecuta sentencias con valores de i de 1 a n, variando de uno en uno.

for i=1:n

sentencias

end

Sentencias `while, break`

La sentencia **`while`** tiene la siguiente estructura:

Las sentencias se siguen ejecutando mientras haya elementos distintos de cero en condición, es decir, mientras haya algún elemento true. El bucle se termina cuando todos los elementos de condición son false (es decir, cero).

La sentencia **`break`** hace que se termine la ejecución del bucle más interno de los que comprenden a dicha sentencia.

4.2.1.- GRÁFICOS

Las funciones más útiles para la representación de gráficos son `plot` y `plot3`.

plot

Crea un gráfico en dos dimensiones a partir de vectores y/o columnas de matrices.

Existen además otras funciones orientadas a añadir títulos al gráfico, a los ejes, etc.

`title('título'),` añade un título al dibujo.

`xlabel('axis x'),` añade la etiqueta "axis x" al eje de abscisas.

`ylabel('axis y'),` añade la etiqueta "axis y" al eje de ordenadas.

`text(x,y,'texto'),` introduce "texto" en el lugar especificado por las coordenadas x e y.

`legend(),` define rótulos para las distintas líneas o ejes utilizados en la figura.

`grid,` activa la inclusión de una cuadrícula en el dibujo. Si a la función plot se le pasan dos vectores como argumentos, los elementos del segundo vector se representan en ordenadas frente a los valores del primero, que se representan en abscisas

Si por el contrario se le pasa un único vector como argumento, dicha función dibuja en ordenadas el valor de los `n` elementos del vector frente a los índices `1, 2, ... n` del mismo en abscisas.

En la Tabla N°4.1 se pueden observar las distintas posibilidades que ofrece la función `plot`, el color, los marcadores y el estilo de línea que se pueden utilizar.

Memoria. Matlab aplicado a visión artificial:

Tabla N° 4. 1
Color, marcadores y estilos de línea para la función Plot

SIMBOLO	***COLOR***	***SIMBOLO***	***MARCADORES***
y	*amarillo*	.	*puntos*
m	*magenta*	*o*	*círculos*
c	*cyan*	*x*	*marcas en x*
r	*rojo*	+	*marcas en +*
g	*verde*	*	*marcas en **
b	*azul*	*s*	*marcas cuadradas*
w	*blanco*	*d*	*marcas en diamante*
k	*negro*	^	*triangulo apuntando arriba*
v	*lineal*	***v***	*triangulo apuntando abajo*
-	*línea continua*	>	*triangulo apuntando a la derecha*
:	*línea a puntos*	<	*triángulos apuntando a la izquierda*
-.	*líneas a barra – punto*	*p*	*estrella de 5 puntas*
--	*líneas a trozos*	*h*	*estrella de 6 puntas*

plot3

La función `plot3` el análogo tridimensional de la función `plot`. Esta función dibuja puntos cuyas coordenadas están contenidas en 3 vectores, uniéndolos mediante una línea continua.

Las opciones son las mismas que las descritas anteriormente para la función `plot`

4.3. FUNCIONES IMPORTANTES

En la librería se encuentran las funciones para el tratamiento de imágenes. En esta librería existen muchas funciones, por lo tanto aquí se procede a explicar sólo algunas, si se desea más información se puede consultar el manual del programa MATLAB.

Esta librería de procesamiento de imágenes soporta 4 tipos imágenes: indexadas, intensidad, binarias y RGB.

indexada: Imagen cuyos píxeles tienen valores que son índices directos a un mapa de color RGB. En MATLAB, una imagen indexada es representada por un array de clase uint8, int16, o double. El mapa de color es siempre un array m*3 de clase doublé.

intensidad: Es una imagen cuyos valores de píxeles corresponden a una escala de grises. En MATLAB, una imagen de intensidad es representada por un array de clase uint8, uint16, o double.

binaria: Una imagen binaria puede ser considerada un tipo especial de imagen de intensidad, conteniendo solamente blanco y negro. Una imagen binaria puede ser guardada en un array de clase double o uint8.

RGB: Una imagen cuyos píxeles son especificados por 3 valores, uno para cada componente de color (rojo, verde y azul) de cada píxel. En MATLAB, una imagen RGB es representada por un array m*n*3 de clase uint8, uint16, o double.

Imwrite: Función que guarda una imagen del workspace a un archivo del disco duro. Ejemplo:

imwrite (Imagen, ‘nombre_archivo.bmp’);

imshow: Función que muestra una imagen del workspace en la figura activa actual (si no hay figuras se crea una nueva). Ejemplo:

imshow(Imagen);

figure, imshow(Imagen);% crea una nueva figura

rgb2gray: Función que transforma una imagen en color RGB en una imagen en blanco y negro. Ejemplo:

ImagenBW = rgb2gray(ImagenColor);

Edge: Realiza una operación de detección de bordes sobre una imagen en blanco y negro. Acepta como argumento el tipo de algoritmo utilizado (sobel, canny…) y los parámetros específicos de cada uno de ellos. La imagen devuelta en una imagen en blanco y negro. Ejemplo:

I_bordes= edge(Imagen,'canny', 0.07);

Imresize: Función que devuelve una imagen que es 'b veces´ el tamaño de la imagen original. La imagen puede ser de cualquier tipo: escala de grises, RGB o binaria. Si 'b' está entre 0 y 1, la imagen resultante es más pequeña, mientras que si es mayor que 1, será mayor. Ejemplo:

Imagen_tam = imresize(Imagen, b)

conv2: Función que devuelve la convolución entre dos matrices. Ejemplo:

Imagen_conv=conv2(A,B)

Immultiply: Función que multiplica dos imágenes dadas. Ejemplo:

Imagen_multiplicada=immultiply(A,B)

4.4.- ANÁLISIS Y RESULTADOS

En este trabajo se ha desarrollado un algoritmo que dado un objeto real capturado por un escáner se obtenga el modelo tridimensional del mismo. Para comprobar la factibilidad de dicho algoritmo se utilizaron varios pares de imágenes. A continuación se muestra la reconstrucción de un cubo. La imagen tomada por el escáner incluía el objeto así como el resto de la escena como se verá en los siguientes programas.

4.4.1.-PROYECCIÓN DE IMÁGENES DEFINIDAS Y SIMPLES

En esta parte del capítulo nos preocupamos primero en saber y estar es factible proyectar una imagen 2D a 3D midiendo la intensidad del pixel

PROGRAMA N°1 (Apéndice B1)

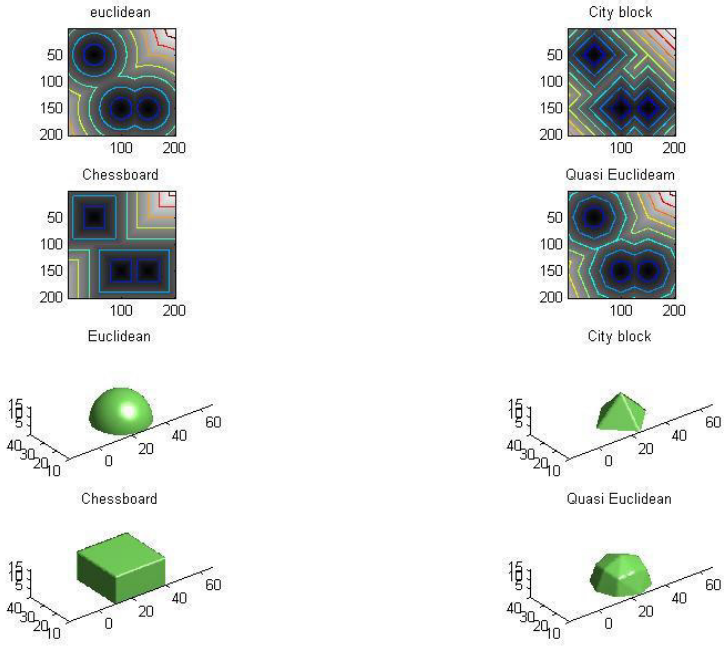


Figura N° 4. 1: *Programa de demostración de proyección de píxeles.*

En el programa anterior se observa el comportamiento de la intensidad de los pixeles al conseguir proyectar una imagen 3D, se observa que a partir de cuadrados 2D se genera cubos, y de círculos 2D genera esferas, etc . Este programa es una demostración básica de la proyección 3D a partir de una imagen 2D esto nos conlleva a afirmar que es posible medir la intensidad de cada pixel entendiéndola como una matriz de 3x3 cuya determinante nos determina su intensidad. Pero en la frontera de la imagen los pixeles se encuentran bien definidos en nuestro problema de imagen. El problema radicar cuando enfrentemos imágenes que no cuentan con esta exactitud de medida tanto en la frontera de la imagen como en la intensidad del pixel, se

puede decir que este programa se hizo solo con la finalidad de demostrar la teoría de jet del capítulo I y que no tiene nada de realismo porque la imagen 2D fue creada en computadora.

PROGRAMA N° 2 (Apéndice B2)

Este programa es una demostración que podemos separar distintos tipos de materiales tomando como base diferentes tonos de color en el pixel

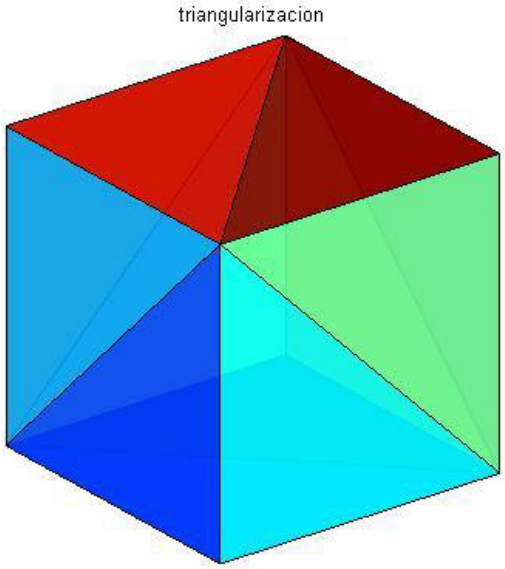


Figura N° 4. 2: *División de materiales de particiones*

En el programa anterior se demostró que es posible separar materiales en 3D si estos vienen con diferente intensidad de color, los planos de separación separan los materiales utilizando el promedio de la intensidad de los voxeles, a cada material se le asignó un diferente color para ver los diferentes materiales constituyentes de la imagen 3D.

La Figura N °4.2 es el resultado de la proyección un hexágono 2D a 3D, mostrando intensidades bien definidas en cuanto a valores del pixel se trate. Esto permite obtener una serie de prismas a partir de la proyección de las direcciones en z+ y z-.

4.4.2.-PROGRAMACIÓN AXIAL DE CABEZA

PROGRAMA N° 3 (Apéndice B3)

A continuación presentamos un programa capaz de leer una imagen de cabeza

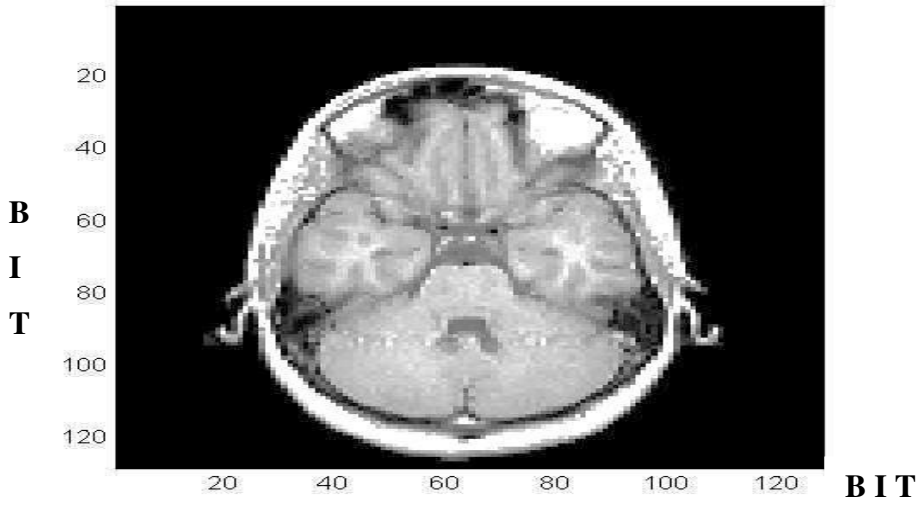


Figura N° 4. 3: *Imagen digital para su conversión matricial.*

El programa anterior es un programa con la capacidad de leer la imagen en forma de matriz, para la computadora cada tonalidad de color es representada por valores numéricos en sistema decimal, este programa tiene la capacidad de diferenciar las diferentes tonalidades de gris de la imagen y darles su respectivo valor numérico para así poder medir su intensidad y después pasar a su posible proyección en 3D. La computadora tomara este conjunto de datos como valores de una sola matriz, nuestro trabajo consiste en agrupar estos valores en matrices de 3x3 para hacer proyecciones en 3D. Las coordenadas (x,y) representan el número de datos que existe en línea recta de un punto a otro, esas coordenadas no se debe entender como dimensiones físicas, la computadora solo entiende a base de números y para ella no es muy importante las coordenadas ni las unidades físicas Pero el color negro que se encuentra alrededor de la imagen es uno de los problemas que tenemos que superar porque la computadora lo entiende como un valor homogéneo de intensidad y si este no es filtrado también será proyectado ocasionando que la imagen sea completamente cubierta y esto sería una deformidad de la imagen 3D, este problema será superado en el programa N°4

A partir de la Figura N° 4.3 se consiguió definir bien las fronteras de la imagen utilizando los principios de "organización de tejido" en la que se necesita fundamentalmente aplicar una variedad de intensidades con el fin de conseguir la expresión de la imagen en forma matricial.

PROGRAMA N° 4 (Apéndice B4)

En el siguiente programa se presenta la definición de fronteras en el interior de una imagen

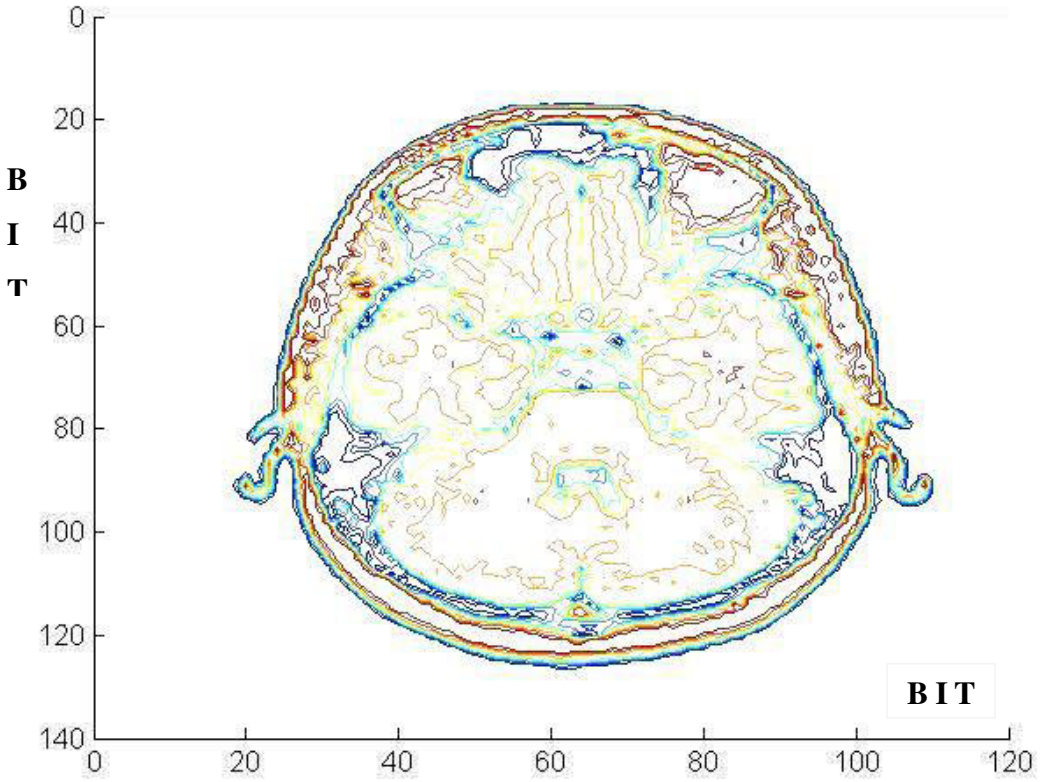


Figura N° 4. 4: *Definición de las fronteras de una imagen en el plano.*

Este programa es capaz de definir las fronteras en la superficie de una imagen en el plano siempre y cuando estas estén determinadas con alto contraste para la calidad de colores de acuerdo a la tonalidad gris que posee cada región. El programa toma un conjunto de datos del plano y define todos sus compuestos interiores mediante su intensidad promedio de pixeles así da un color respectivo en la frontera representada mediante curvas.

Al obtener un gran conjunto de datos de textura de la imagen descrito en el capítulo II de la página 50 se obtiene una lectura de planos bien definida expresada en forma discreta en el programa, como se observa en la Figura N° 4.4.

Una vez definida los materiales (órganos) interiores de nuestra imagen y determinada la frontera pasamos a filtrar el interior de esta para así conseguir la Figura N°4. 1 en donde se aprecia todos los posibles órganos internos de la imagen de cabeza 2D, aquí se obtiene diferentes intensidades de pixeles que es la información del conjunto de órganos internos que nos servirá en el siguiente programa

PROGRAMA N° 5 (Apéndice B5)

Este programa mide la pérdida de la intensidad de los pixeles a la cual hemos llamado técnica de rodajas por que poseen una visión 3D de la imagen proyectada en rodajas a partir de definir fronteras en el plano usando la teoría "lectura de planos y mallas" del capítulo II de la página 53 donde se define subcapas a partir de una imagen 2D.

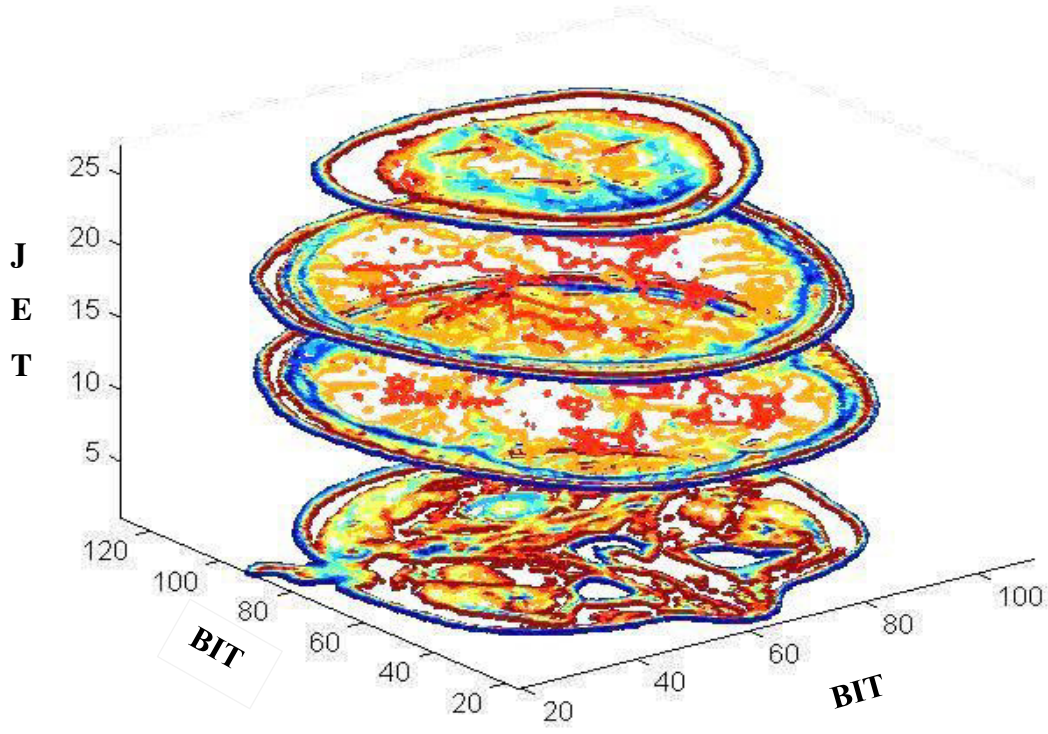


Figura N° 4. 5: *Imagen proyectada en rodajas muestra la perdida de pixeles según se vaya elevando.*

Este programa se ha diseñado utilizando “la técnica de lecturas y planos de diferentes intensidades” que consiste en la textura tridimensional de planos que ofrece una solución viable que se vio en la Figura N° 2.15, porque no hay manera de evitar la filtración entre capas necesarias para un buen rendimiento, se tiene que ser capaz de extraer una muestra debidamente filtrada en la superficie de una sub-imagen ya que las texturas individuales a menudo se repiten a través de una malla de superficies.

Aquí se consiguió definir planos de diferentes intensidades Figura N° 4.5 en la que se muestra placas de textura que contiene cuatro imágenes en una matriz de dimensión 1024x1024, proyectados de una textura única de mapa de un arreglo de subcapa 2D (matrices de 3x3) en un subconjunto de datos. En cada sub-imagen (rodaja), una octava parte de la textura perteneciente a la sub-imagen copia los datos opuestos para producir una reproducción parcial de las uniones virtuales de estos planos que nos dan como resultado final la imagen de cabeza en 3D.

PROGRAMA N°6 (Apéndice B6)

Este programa rellenaría la imagen anterior produciendo una visión 3D completa de la imagen en el espacio virtual.

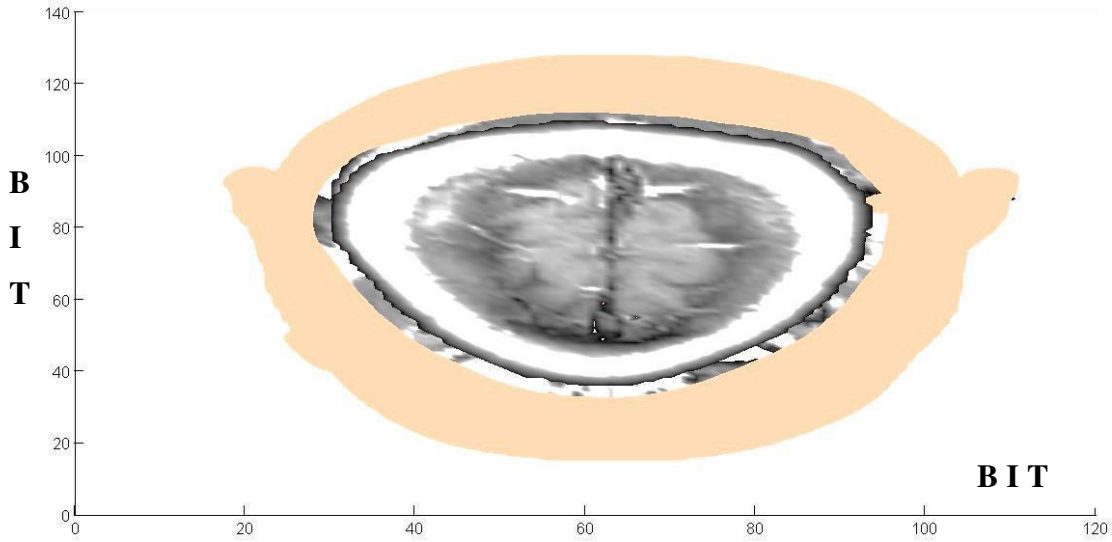


Figura N° 4. 6: *Visualización de la imagen mediante las uniones virtuales de las rodajas para observar una posible proyección en 3D.*

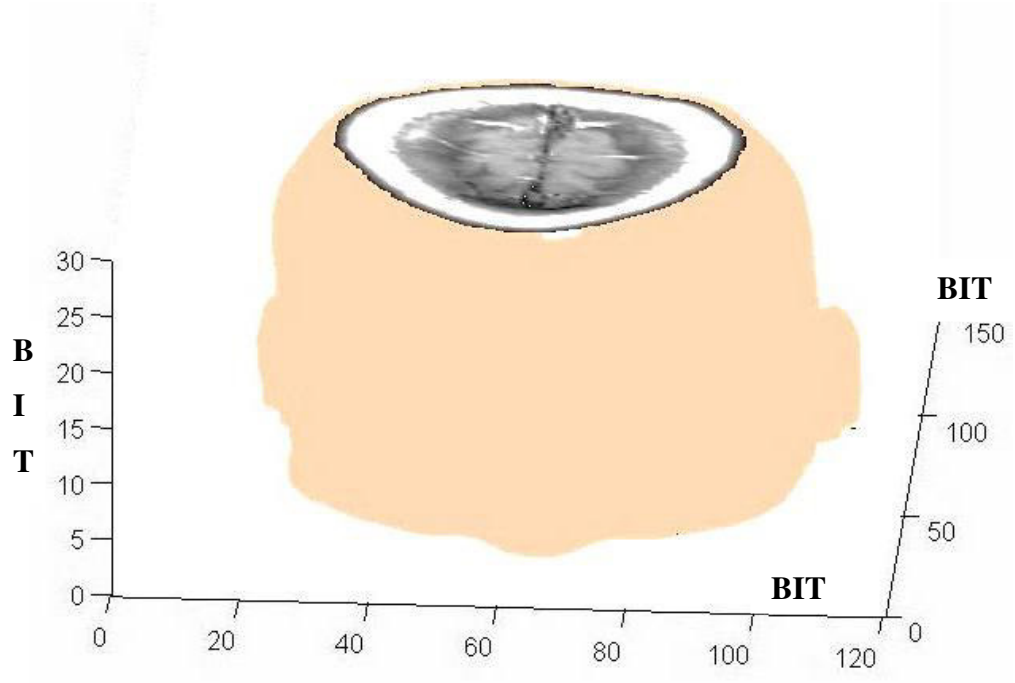


Figura N° 4. 7: *Visualización de la imagen para observar la proyección 3D de vista lateral.*

Para la realización del diseño del programa anterior y obtención de una imagen 3D obtenidas en las Figuras N° 4.6, 4.7 mediante técnicas de suavización y relleno vistas en el capítulo III en donde se presenta un conjunto de principios representadas en ecuaciones tales como (3.6), (3.8) y (3.10) utilizadas para suavizar la imagen, pero esto tiene un costo computacional disminuyendo cierta nitidez en la imagen obtenida. La malla triangular resultante es demasiado fina entonces se puede decir que el número de triángulos es demasiado alto.

Este algoritmo posee la capacidad de suavizar la imagen haciéndola más fina en su visualización artificial, con el programa de este trabajo y la implementación de algunas mejoras en cuanto a la precisión de los datos discretos utilizados, se podría proyectar imágenes médicas 3D de cualquier tipo de tumores permitiendo ofrecer un mejor diagnóstico en el planeamiento medico de estos. Permitirá también conocer la forma exacta del tumor ayudando a que se realice una incidencia más precisa de radiación en el tejido cancerígeno sin dañar los tejidos no comprometidos por el tumor, pero eso se dejara para trabajos posteriores.

4.4.3.-PROGRAMACIÓN SAGITAL DE CABEZA

PROGRAMA N°7 (Apéndice B7)

A continuación, se muestra los programas de imagen en cabeza de lado sagital. Esta imagen sagital fue obtenida por escáner manipulada por un computado por el Programa N° 7 como se muestra en la Figura N° 4.8.

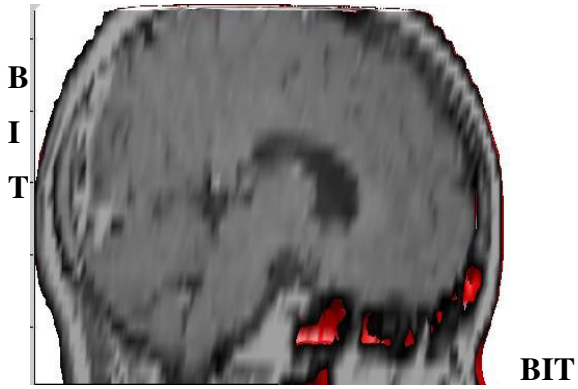


Figura N° 4. 8: *Imagen con un alto grado de manipulación computarizada para su posible proyección en 3D.*

La Figura N° 4.8 que es una figura sumamente alterada por MatLab (versión prueba de internet) en el ordenador, se tuvo que alterar cada valor de los pixeles para poder definir y promediar los pixeles de la imagen, debido a que es una imagen radiográfica escaneada que da o pierde información en cada pixel, para lograrlo se utilizó la técnica de DICOMEX y VESTA mostrada en el capítulo I que nos enseñan una técnica de como aislar y eliminar estas esporas y reemplazarlas por otro valor de pixel adecuada a la imagen mediante la técnica de "Marching Cube".

El programa anterior tiene la capacidad de reconocer la imagen para leerla en un conjunto de números agrupados en matrices en 3x3 conocidos como pixeles, la determinante de estos pixeles nos da la intensidad que es con la cual se trabaja para la obtención de una elevación representada por "Marching Cube" que fue entendida como una sucesión de cubos que a través de su elevación continua en la obtención de barras pierden intensidad como se muestra en la Figura N° 1.3

PROGRAMA N° 8 (Apéndice B8)

Este programa proyecta una elevación definida por un trozo, que muestra que puede ser elevada a toda su expresión 3D.

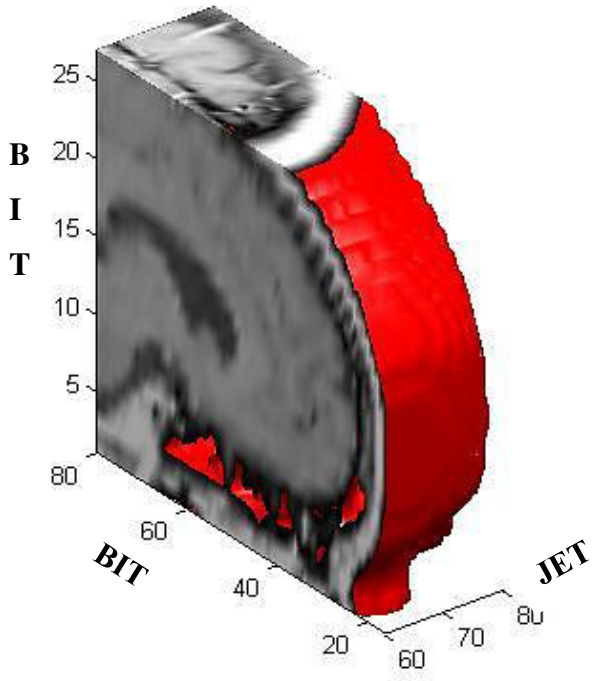


Figura N° 4. 9: *Observación de la intensidad de pixeles y elevación a una determinada altura.*

Este programa consigue la elevación de todos los pixeles, filtrando los bordes pero en esta parte del programa solo logramos elevar 10 unit mostrada en la Figura N° 4.9 aquí solo se elevó una parte de la imagen para poder observar el comportamiento interior a una determinada altura y así poder apreciar el cambio de los pixeles a través de los diferentes units de elevación, aquí se observa claramente ambas caras y se ve la perdida de contraste por intensidad al ser elevado. Sabemos que funciona porque ciertos órganos internos de cabeza pierden resolución en su visualización al ser elevados en 10 unit, aquí no ser observa con claridad pero en el programa se observa los detalles aquí mencionados.

El programa anterior muestra el comportamiento de los voxeles para la posible creación de la imagen 3D, la mayor cantidad de decimales en los pixeles permiten diferenciar los detalles de los órganos internos en una radiografía.

PROGRAMA N° 9 (Apéndice B9)

Este programa repite la técnica pero en una cara completa y midiendo la intensidad total del pixel.

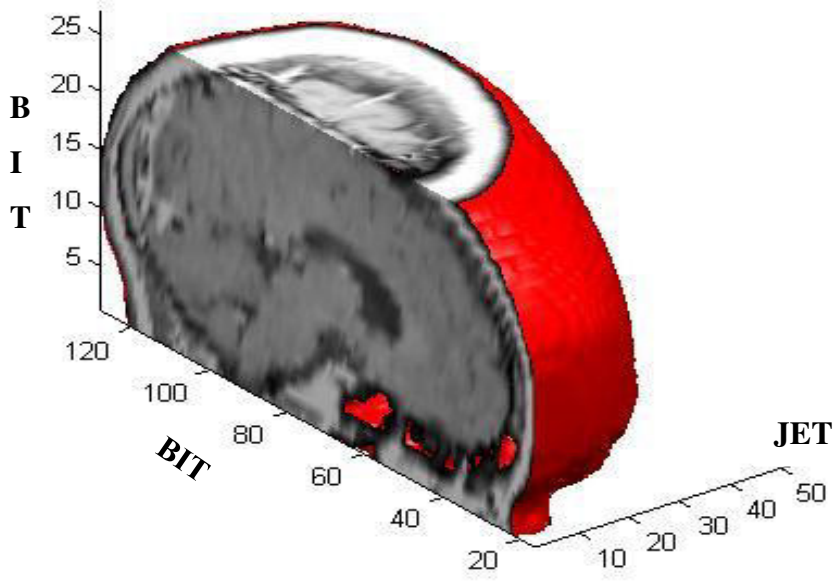


Figura N° 4. 10: *Imagen en toda una expresión 3D producida al medir toda la intensidad de los pixeles*

El programa anterior es un programa que mide toda la intensidad de los pixeles en una gran sucesión de voxeles, dando como resultado la imagen de cabeza en 3D a partir de 2D pero si observamos los alrededores de la superficie todavía falta mejorar la técnica triangulación en la frontera ocasionando la perdida de la nitidez en la imagen.

Las esporas originadas debido a la diferencia de intensidades en la técnica de DICOMEX fueron aisladas y reemplazadas por valores que no perjudica la imagen permitiendo su

visualización en forma sagital, en la frontera se ve la aplicación de la técnica de triangulación con un rango de intensidades desde 10 unit hasta 40 unit, se realizó algunos rellenos que deformaban la imagen, con la finalidad de obtener una adecuada imagen de cabeza 3D. Se trató de disminuir el espaciamiento entre las barras de intensidades porque deforman la imagen.

La técnica de barras aplicada en la Figura N° 4.10 comprada con la imagen de rodajas de Programa N° 5 consigue obtener más detalles de la imagen en el interior de la cabeza mientras que la técnica de rodajas de la Figura N°4. 2 no se observa detalles precisos.

En la forma de la frontera de la imagen total (capa roja) el número de triangulaciones es grande por lo que requirió un gran costo computacional (memoria de procesador) y tiempo en dar resultados de los detalles observados en la imagen que no son muy exactos pero podría servir para conocer su forma real.

Uno de los defectos que posee esta técnica utilizada es la limitación en cuanto a la resolución de los pixeles por eso es necesario filtrar los pixeles que ocasionan deformación, en cambio la técnica rodajas solo trabaja con fronteras de órganos interno proyectando en 3D.

PROGRAMA N° 10 (Apéndice B10)

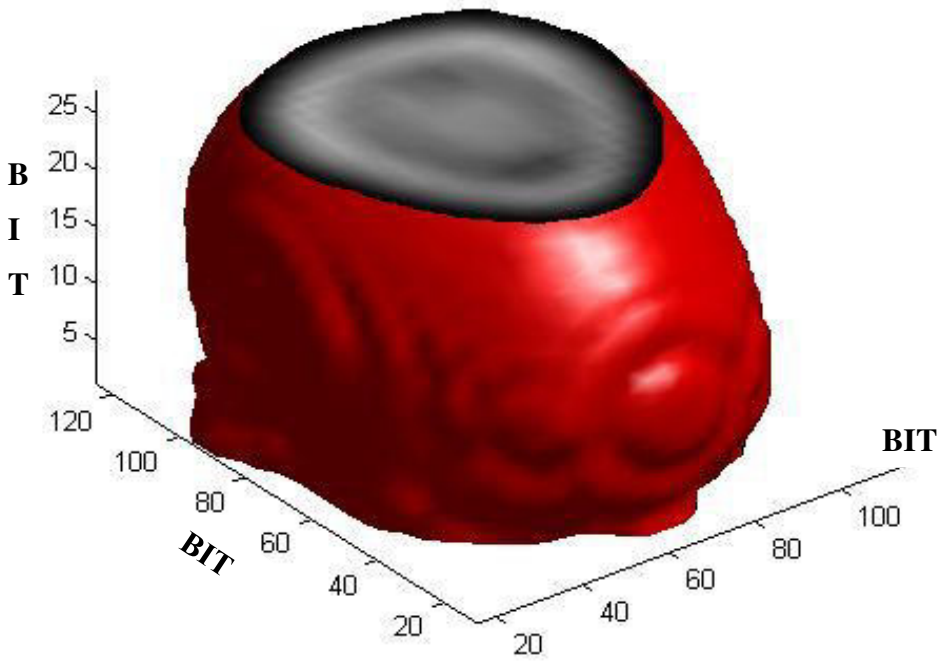


Figura N° 4. 11: *Imagen producida al combinar todas las técnicas sagitales.*

Haciendo lo anterior (programa 9) en el lado anterior y posterior de la cabeza, uniendo ambas partes se obtiene la imagen total de cabeza aplicando la teoría del capítulo III para suavizar deformaciones de frontera.

Finalmente utilizando las ecuaciones (3.5),(3.8),(3.9) y (3.10) del capítulo III se logró mejoras de suavidad como se muestra en la Figura N° 4.11 en donde se observa correcciones de suavidad y no se aprecia los triángulos en la superficie como en la Figura N° 4.10.

Sin embargo, cuando proyectamos en 3D el conjunto de pixeles en forma elevada Figura N° 3.1. (b), la imagen tiende a deformarse, el problema radica en determinar una proyección de pixeles en forma óptima que permita obtener una buena imagen 3D. En este trabajo se logró obtener una proyección de vóxel adecuada Figura N° 3.1. (a).

A partir de la Figura N° 4.10 el programa anterior consigue definir bien las fronteras de la imagen utilizando los principios de "organización de tejido" en la que se necesita

fundamentalmente aplicar una variedad de triangulaciones más intensas con el fin de conseguir la expresión de la imagen en forma matricial y ser refinada en 3D.

PROGRAMA N° 11 (Apéndice B11)

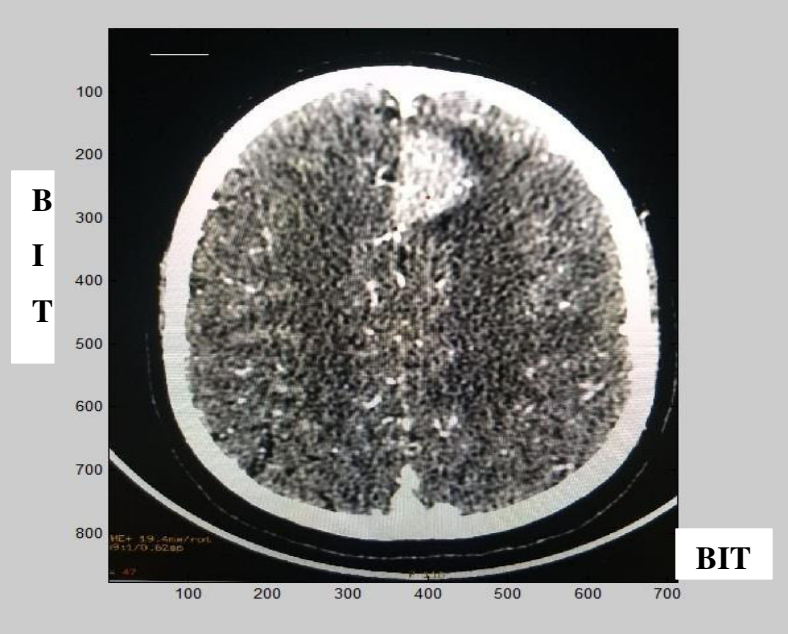


Figura N° 4. 12: *Imagen extraída por una tomografía compurizada del IMSS UMAE*

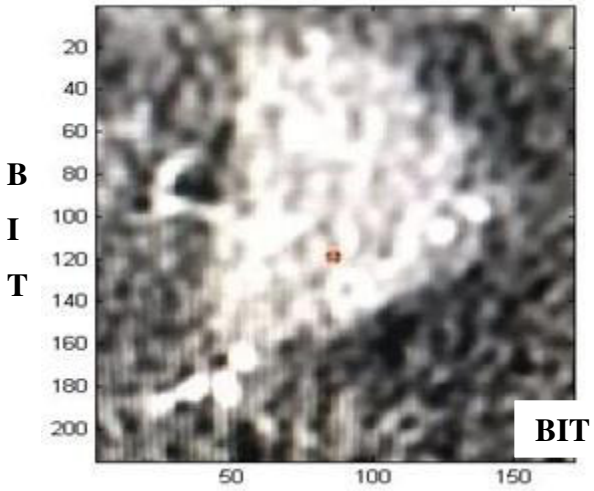


Figura N° 4. 13*:Parte del tumor en el cerebro detectado por la tomografía axial computarizada*

En la Figura N°4.12 nos muestra una imagen de cabeza en 2D con una medida de 700x800 pixeles este procesamiento de pixeles es dificultoso para la computadora debido que nuestra PC solo puede procesar hasta 3 millones de datos por segundo estos datos al sufrir lecturas de pixeles en 3 canales se convierten 1.47 millones de datos esto ya satura la capacidad de programación, al producirse condiciones de elevación y condiciones de frontera, además no nos serviría de mucho debido a que estamos interesados en proyectar los tumores y para ello debemos de recortar la imagen.

En la Figura N°4.13 nos muestra una reducción de datos que seria 200x200 dando un total de 0.04 millones de datos siendo mas accesible a nuestro recurso computacional además esta lectura se podría procesar bajo condiciones de frontera sin demasiada demora debido a que los datos ocuparían cerca de 2 núcleo de los 5 núcleos que posee nuestra computadora

PROGRAMA N° 12 (Apéndice B12)

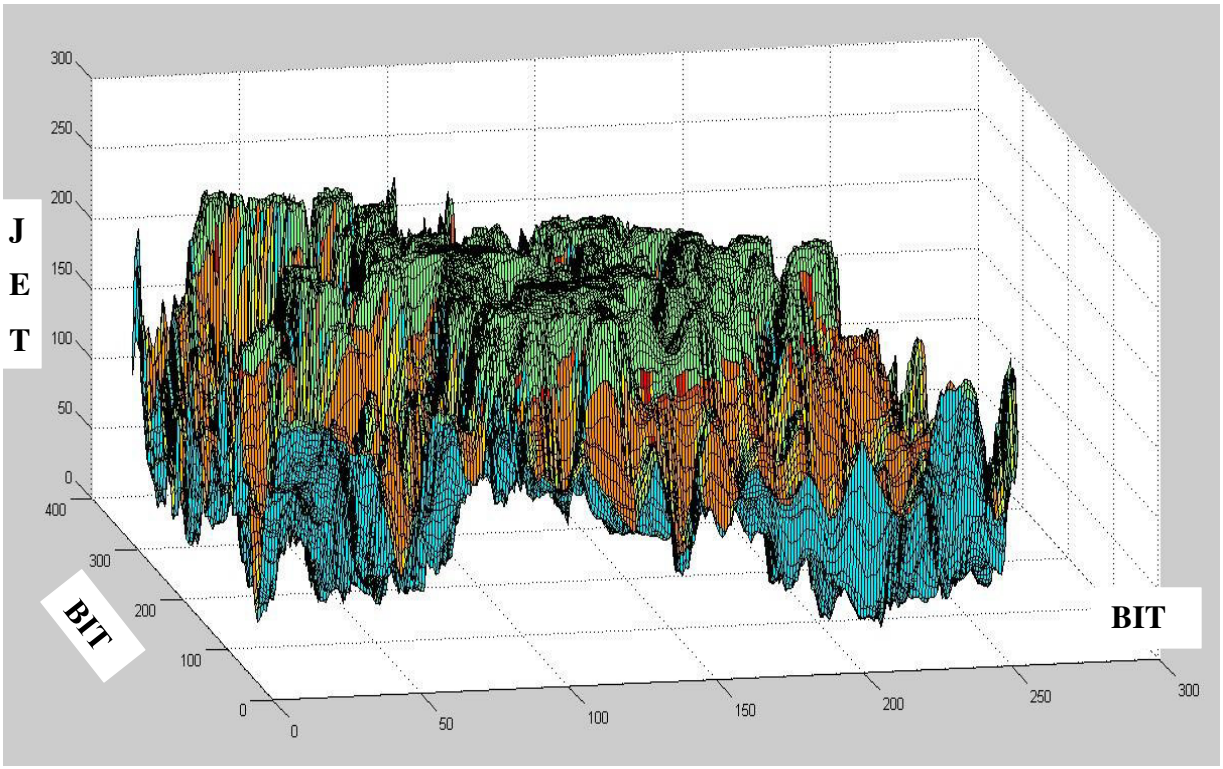


Figura N° 4. 14:*Proyección de tumor cerebral a partir de la Figura N°4.13*

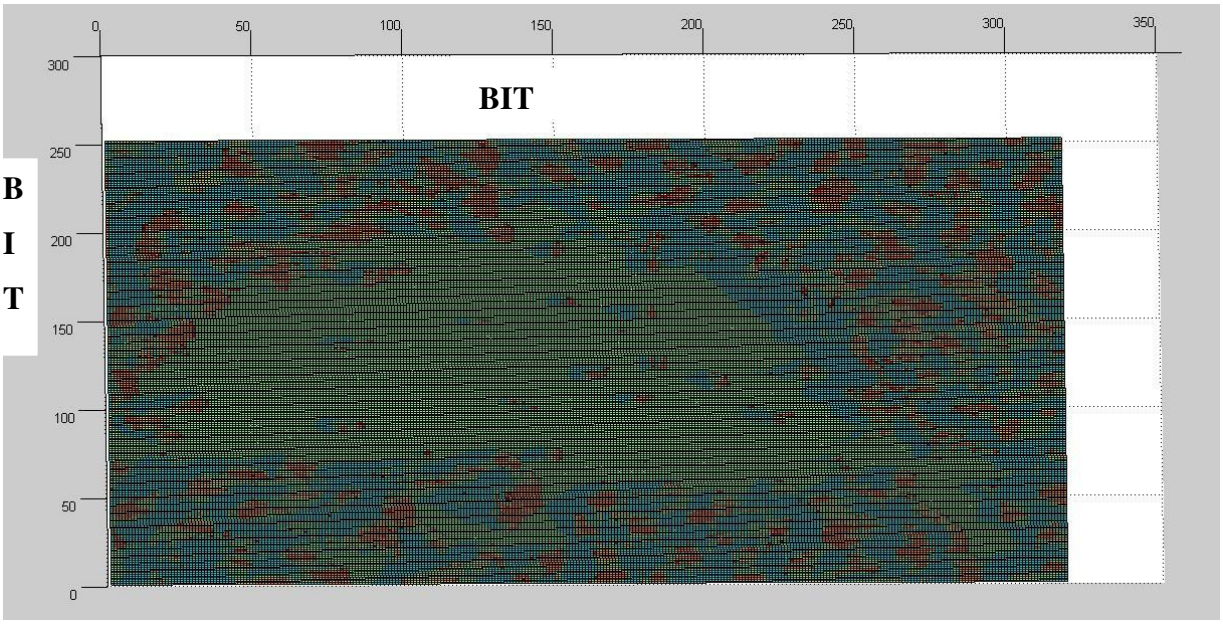


Figura N° 4. 15: *Esta figura es una representación de la Figura N°4.14 solo que aquí esta la representación con intensidades desde una vista en z+(arriba)*

La Figura N°4.14 es la aproximación 3D de la Figura N°4.13 que utiliza la técnica de Marching Cube y la ecuación (1.11) que nos habla de la elevación de un jet esto determina la intenciada de cada pixel mediante una elevación de voxeles; de esta figura podemos decir que los tumores en el cerebro no tienen una forma uniforme y determinada, de lo contrario son demasiado deformes o amorfas en toda su estructura; este tumor lejos de estar confiormado por solamente células cancerígenas en su interior también existen células sanas; bueno esto personalmente sorprende debido a que los tumores cancerígenos son acumulaciones de células sin células sanas en su interior.

La Figura N°4.15 es una visión de la base de la Figura N°4.14 si hacemos una comparación visual con la Figura N°4.13 es la misma solo que esta figura tiene demasiados problemas de pimienta la escala a aumentado de 250x325 debido a que estamos cambiando el espacio de visión virtual en la proyección 3D esto debido a que los pixeles al ser elevados a voxeles con se combinas unos con otros al corregir su estructura o se disminuyen al deformar su estructura en nuestro caso y todavía no se entiende la razón corregir su estructura mediante la elevación la imagen en el plano a aumentado ; esto sabre responder mas adelante..

PROGRAMA N° 13 (Apéndice B13)

Figura N° 4. 16*: Creación de lectura de pixeles en canal rojo usando una cámara simple*

Este programa tiene la capacidad de ver pixeles de color rojo desde una mínima cantidad 0 a una gran cantidad de 300 la fijación exentrica del programa todavía es manual debido a que no podemos y de momento resolver la automatización de la fijación exentrica este programa podría ser utilizado para determinar la velocidad de objetos acercándose debido aque nosotros apreciamos el aumento de pixeles a manera que un objeto se acerca. En el campo medico este programa sirve para determinar flujo sanguíneo en determinadas partes del cuerpo debido a que la piel esta roja sabemos cuando hay una precion de flujo sanguíneo y cuando no la piel conserva su color determinado, otras de las aplicaciones que podríamos aplicar es el auumneto de temperatura en las personas debido a que generalmente en las pupilas de la cara se nota el cambio de color a rojo cuando una persona siente colera o miedo.Otra aplicación seria el de seguir una bacteria de color rojo de un tamaño dado en milímetros; son aplicaciones que se me ocurren de momento pero en otros trabajos futuros se que a otras personas se les ocurrirá.

NOTA:¿Cuál es el tamaño de un píxel?. Esta es una de las clásicas preguntas que se hace todo el mundo. Sin embargo, un píxel no tiene una medida fija, es decir, 1 píxel no mide X cm siempre, sino que depende de la resolución. Por ejemplo, si una imagen posee 500×200 píxeles sabemos que está divida en 100.000 píxeles. Sin embargo, no sabemos cuánto mide cada píxel. Todo cambia cuando sabemos la resolución por pulgada. La resolución indica la cantidad de píxeles por pulgada que posee una imagen. Si decimos que una imagen tiene 100 píxeles por pulgada podemos deducir que cada píxel equivale 2,54 milímetros, ya que cada pulgada que son 2,54 centímetros está dividida en 100 píxeles.

4.5.- CONCLUSIONES

Al tener una imagen 2D bien definida en el plano por los píxeles, se pudo observar que se pueden proyectar imágenes en 3D en distintas direcciones del octante y tales observaciones se puede ver en el código Matlab versión 9 (de prueba), el problema radica cuando se tiene una imagen en 2D con sus bordes coloreados o perturbados comúnmente llamado por programadores “efectos de pimienta”, estos se proyectan también en 3D y la imagen pierde contraste de definición, para superar este problema de resolución usamos la técnica de triangulación que todavía no se pudo mejorar en este trabajo.

Otro problema que se presenta en una imagen proyectada en 3D a partir de 2D es la gran variedad de bits en código decimal en la frontera de la imagen que perjudica la obtención del núcleo matricial computacional de calidad de imagen.

Se pudo aumentar las intensidades de los píxeles para poder tener una observación clara de las imágenes escaneadas así como también se alcanzó a observar que ocurriría si los píxeles son aumentados a una intensidad demasiada alta, en este punto se pudo ver que pierde resolución de la imagen.

Al proyectar la imagen a vóxel se puede ver dos casos, el primero es cuando se proyecta una intensidad demasiado baja, la imagen es elevada, podemos observar la superficie con algunas protuberancias que disminuyen la calidad de imagen 3D deseada, segundo es cuando se proyecta una intensidad demasiado alta, podemos observar picos que deforman la imagen.

Como las células de nuestro cuerpo son dinámicas incluso el mas mínimo error en la reconstrucción del temor tendría consecuencias desastrosas; a mi apreciación personal nuestro trabajo tiene un error de 80% a mas para ser mas precisos tendría que trabajar con sujetos vivos en tiempo real.

Y por último se puede concluir que se han obtenido imágenes en las cuales no se a logrado aclarar de manera limpia y sin ruido dificultando la ostensión de imágenes 3D con alta resolución. Todas las técnicas aplicados en este trabajo presentan situaciones en las que no se obtiene muy buenos resultados esto debido a las limitaciones de los recursos computacionales por que se trabajó con 1.2 o casi cerca de 1.3 millones de datos de cada imagen ya sea en el plano o en distintas intensidades 3D de la imagen

TRABAJOS A FUTURO

En este trabajo aún existe un grandes problemas por superar y mejorar imágenes médicas de cualquier índole, el mayor problema que tenemos es superar las condiciones de frontera cuando se escanea una imagen se pierde información debido a la lectura del escáner y además que el píxeleado no es tan exacto en la frontera y aun peor en medio de la imagen existe perdidas muy definidas de la figura ya sean por factor contaminante o por cualquier otra índole producen representación de píxeles en decimales, que es el mayor problema, debido a que estos valores decimales producen gran consumo de recurso computacional, porque esto a su vez produce gran cantidad de datos binomiales, aquí que debemos recortar que el "símbolo de sistemas" de la computadora opera a través de ceros y unos, pero nosotros no hemos trabajado desde el "símbolo de sistemas" nosotros aquí hemos trabajado en Matlab que está dentro de Windows 7, que a su vez está dentro del "símbolo de sistemas" .

Para poder explicar mejor la solución que estamos planteando para resolver el problema de decimales nosotros en este trabajo seguimos los siguientes pasos:

1. Nuestros datos llegan al símbolo de sistemas en lenguaje binario.
2. Después pasan al Windows y lo lee en sistema decimal de píxeleados pero en Windows 7 no se puede operar.
3. Por último los datos obtenidos para cada imagen pasan a Matlab 2009 (de prueba) en donde se mostraron nuestros resultados.

Estos tres pasos demanda bastante recurso computacional y a su vez gran pérdida de datos para superar esto en un trabajo a futuro tenemos que usar solo el "símbolo de sistemas" y después ir directamente a Matlab sin pasar por Windows y tomar solo los comandos que necesitamos del Matlab 2009 (de prueba) y adaptarlos directamente al "símbolo de sistemas" con esta teoría como planteamiento ya se podría superar la gran cantidad de datos decimales, porque ya se trabajaría solo con lenguaje binario , y a su vez ahorrar bastante recurso computacional. Por ultimo una vez superada la pérdida de datos y la gran cantidad de decimales. Así proyectaremos imagenes en 3D a partir de 2D.

REFERENCIAS BIBLIOGRÁFICAS

[1] B. S. C. M. Elliott, Modeling and computation of two phase geometric biomembranes using surface, Journal of Computational Physics , 2010.

[2] A. R. S. Gross, Numerical methods for two-phase incompressible flows, Vol 40, Springer-Verlag,, 2011.

[3] T. P. M. R. U. Diewald, Anisotropic diffusion in vector field vizualization on Euclidean domains and surfaces, IEEE Trans. : Visualization Comput. Graphics 6, 2000.

[4] B. E. C. a. J. R. F. J. M. Butterworth, http://arxiv.org/abs/hep-ph/0201098, Phys. Rev. D 65, 096014, 2002.

[5] M. CACCIARI, FASTJET: A CODE FOR FAST kt CLUSTERING, AND MORE, Paris 6, France: LPTHE, Universit´e P. et M. Curie , 6 jul 2006.

[6] A. G. J.-P. Thirion, The 3D Marching Lines Algorithm, Graphical Models and Image Processing, 58,pp. 503 – 509,, 1996.

[7] H. L. A. V. G. T. T. Lewiner, Efficient implementation of marching cubes' cases with topological guarantees, ACM Press: Journal of Graphics Tools 8(2), pp. 1 – 15,, 2003.

[8] B. R. Schlei, Hyper-Surface Extraction in Four Dimensions Theoretical Division - Self Assessment, Special Feature, Los Alamos : a portion of LA-UR-04-2143-168, 2004.

[9] Courtesy of G. T. Seidler, Physics Department, University of Washington, Seattle.

[10] B. Schlei, Volume-Enclosing Surface Extraction, GSI Helmholtz Centre for Heavy Ion Research GmbH, Planckstraße, 8 Jun 2012.

[11] S. Fortune, in Proceedings of the second annual symposium on Computational geometry, p. 312, 1986.

[12] A. R. M. Olshanskii, A finite element method for surface PDEs: Matrix properties, Numer. Math. 114 491-520., 2010.

[13] A. R. X. M. Olshanskii, An eulerian space-time finite element method for diffusion problems on evolving surfaces, SIAM J. Numer. Anal. 52 1354-1377., 2014.

[14] E. V. Chernyaev, Marching Cubes 33: Construction of Topologically Correct Isosurfaces, Technical Report : CERN CN 95-17, CERN., 1995..

[15] C. Crassin, F. Neyret, S. Lefebvre, M. Sainz y E. and Eisemann, Beyond Triangles : Gigavoxels Effects In Video Games., SIGGRAPH 2009: Technical Talk., 2009.

[16] W. E. C. H. E. Lorensen, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics (Proceedings of SIGGRAPH 87): Volume 21, Issue 4, pp. 163–169., 1987.

[17] G. M. H. B. Nielson, The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes., Proceedings of the 2nd Conference on Visualization '91,: IEEE Computer Society Press, pp. 83–91., 1991.

[18] A. a. W. J. Van Gelder, Topological Considerations in Isosurface Generation, ACM Transactions on Graphics: Volume 13, Issue 4, pp. 337–375, 1994.

[19] U. S. G. Survey, National Elevation Dataset. http://ned. usgs.gov/, 2006.

[20] P. A. B. J. NING, An Evaluation of Implicit Surface Tilers, IEEE Computer Graphics and Applications, Volume 13, Number 6, pp. 33–34., 1993.

[21] C. C. Tanner, C. J. Migdal y M. T. and Jones, The Clipmap: A Virtual Mipmap, Proceedings of SIGGRAPH 1998, pp. 151–158., 1998.

[22] R. Shu, C. Zhou y M. S. and Kankanhalli, Adaptive Marching Cubes, The Visual Computer: Volume 11, pp. 202–217, 1995.

[23] G. M. NIELSON, Dual Marching Cubes". Proceedings of the Conference on Visualization, 04, pp. 489–496., 2004.

[24] T. Ulrich, Super-size it! Scaling up to Massive Virtual Worlds, SIGGRAPH: 2002 Courses., 2002.

[25] R. a. G. E. Pajarola, Survey on Semi-Regular Multiresolution Models for Interactive Terrain Rendering, The Visual Computer: International Journal of Computer Graphics: Volume 23, Number 8, pp. 583–605., 2007.

[26] Y. Livny, Z. Kogan y J. and EL-SANA, Seamless Patches for GPU-Based Terrain Rendering, The Visual Computer: International Journal of Computer Graphics: Volume 25, Number 3, pp. 197–208, 2009.

[27] J. F. Blinn, Models of Light Reflection for Computer Synthesized Pictures, Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques: pp. 192–198., 1977.

[28] R. S. G. a. J. C. Nagtegaal., An efficient 3D visualization technique for finite element models and other coarse volumes., In Proceeding of the Siggraph 89. , 1989..

[29] B. Hamann., Modelling contours of trivariate data., Modelisation Mathematique et Analysis Numerique 26:51–75, 1992..

[30] A. v. G. a. J. Wilhelms., Topological considerations in isosurface generation., ACM Transactions on Graphics, 13(4):337–375, , 1994..

[31] G. Farin., Curves and Surfaces for Computer Aided, Geometric Design: 4th edn. Academic Press, Boston, 1997..

[32] A. v. G. a. J. Wilhelms., Topological considerations in isosurface generation., ACM Transactions on Graphics, 13(4):337–375, , 1994.

[33] R. S. a. R. S. C. Montani, A modified look-up table for implicit disambiguities of marching cubes., The Visual Computer, 10(6):353–355, , 1994..

[34] W. E. L. a. H. E. Cline., Marching cubes: a high resolution 3d surface reconstruction algorithm, puter Graphics, 21:163–169., 1987.

[35] G. M. N. a. B. Hamann., The asymptotic decider: resolving the ambiguity in marching cubes., In Proceedings of the IEEE Visualization ’91,: Los Alamitos: IEEE Computer Society Press, pp. 83–91., 1991..

[36] A. v. G. a. J. Wilhelms., Topological considerations in isosurface generation, ACM Transactions on Graphics, 13(4):337–375,, 1994..

[37] G. G. H. A. P. M. H. R. S. Nielson G M, Optimal separating surface for enumerated volumes., In Proc. VisSym'03, Grenoble, France, Eurographics: Association pp.75-84., 2003.

[38] L. C. S. O. G. S. M. Bertalmio, Variational problems and partial differential equation on implicit surfaces: The framework and examples in image processing and pattern formation, J. Comput.Phys. 174-759-780, 2001.

[39] C. M. E. G. Dziuk, Finite element methods for surface PDEs, Acta Numerica 289-396., 2013.

[40] A. R. J. G. M. Olshanskii, A finite element method for elliptic equations on surface, SIAM, J. Numer. Anal. 47 3339-3358, 2009.

[41] M. A. O. Alexey Y. Chernyshenko, An adaptive octree finite element method for PDEs posed on surface, August 19, 2014.

[42] R. N. M. P. A. Bonito, Dynamics of biomembranes: effect to the bulk fluid, Math. Model. Nat. Phenom. 6 25-43., 2011.

[43] J. Grande, Eulerian finite element methods for parabolic equations on moving surface, SIAM Journal on Scientific Computing 36 248-271, 2014.

[44] M. G. L. S. Z. P. Hansbo, Characteristic cut finite element methods for convection-diffusion problems on time dependent surfaces, Uppsala University.: Tech. rep., April 2013.

[45] M. C. a. G. P. Salam, http://arxiv.org/abs/hep-ph/0512210.

[46] B. K. Natarajan., On generating topologically consistent isosurfaces from uniform samples., The Visual Computer 11(1):52–62, , 1994.

[47] Z. L. L. K. H. A. Nielson G M, Parameterizing marching cubes isosurfaces with natural neighbor coordinates., Advances in Geometric Modeling and Processing: Chen FL, JÄuttler B (eds.), Springer pp.315-328., 2008.

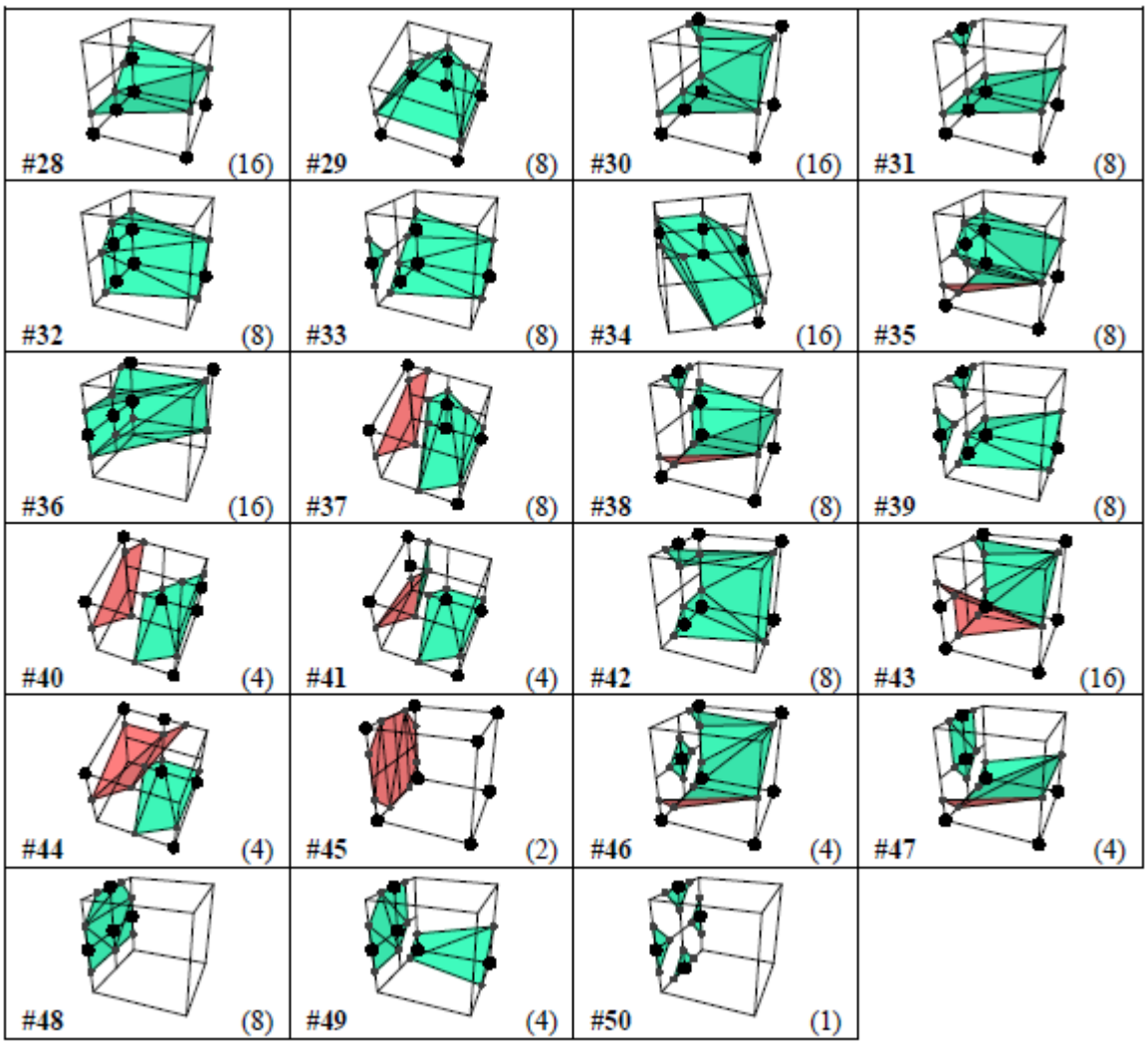
[48] A. P. Meyer D M, Intrinsic parameterizations of surface meshes., Eurographics, Computer Graphics Forum, 21(2): 209-218., 2002.

[49] K. M. J. F. A. K. R. Warfield S K, Adaptive, template moderated, spatially varying statistical classification., Med Image Anal, 4(1): 43-55., 2000.

APÉNDICE A

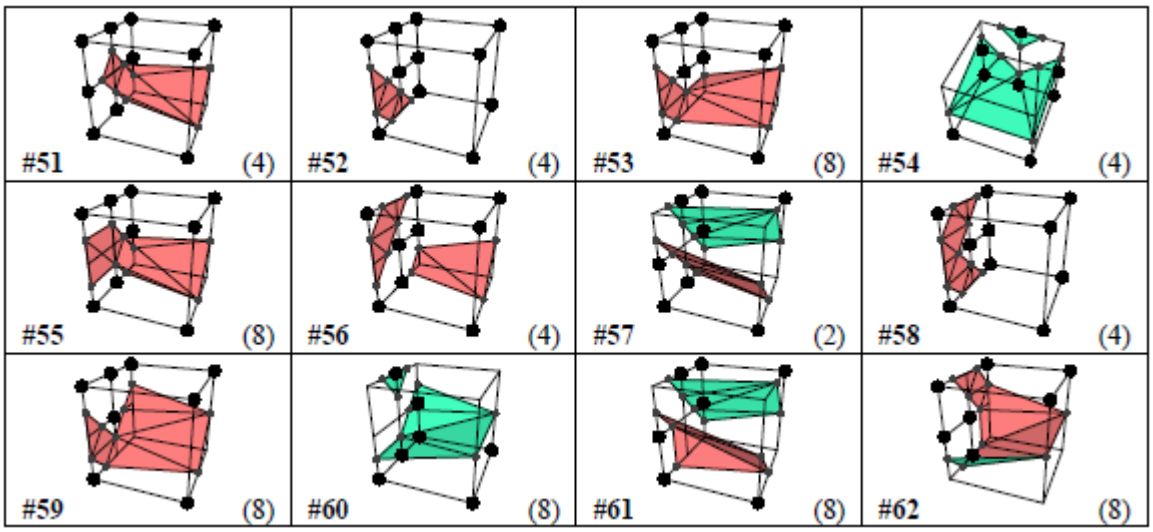
APÉNDICE A.1

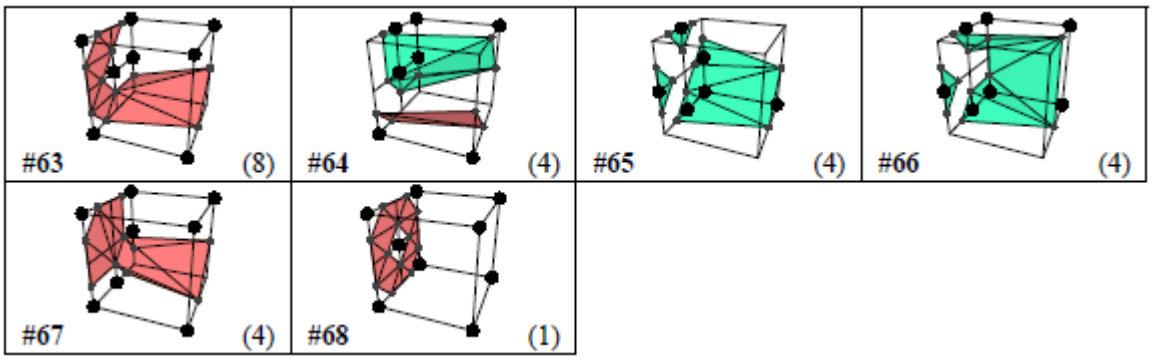
Estas son las clases de equivalencia de cubos 16 de transición para que exactamente tres muestras los valores están en el interior del espacio sólido. Estas clases representan 130 de los 512 de casos distintos [15].

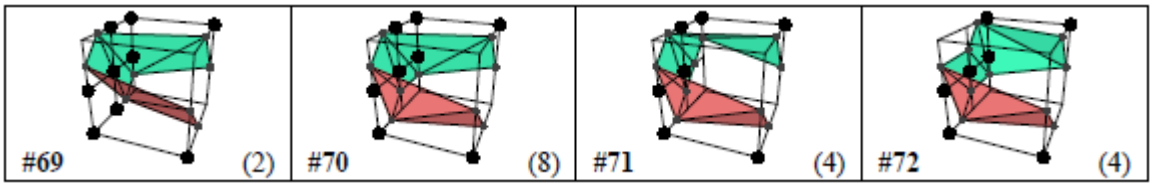


APÉNDICE A.2

Estas son las clases de equivalencia de cubos de transición 18 que corresponden a las inversas de clases que aparecen fuera del material que tienen al menos un cuadrante ambiguo en la cara de resolución completa. Estas clases representan 95 de los 512 casos distintos [15].







APÉNDICE A.3

Esta tabla muestra todas las clases de equivalencia de cubos de transición 73 e identifica el estabilizador subgrupo H para el elemento representativo mostrado para cada clase. Para las clases dentro y fuera del material que no incluyen inversas geométricas, el inverso de clase de equivalencia asociado que se desea aplicar.

Class	Inverse class	Stabilizer subgroup H	Class size
#0 		D_8	2
#1 		$\langle r^3 f \rangle$	8
#2 		$\langle r^2 f \rangle$	8
#3 		D_8	2
#4 		$\{e\}$	16
#5 		$\langle r^2 f \rangle$	8
#6 	#51 	$\langle r^3 f \rangle$	4
#7 		$\{e\}$	16
#8 	#69 	$\langle r^2, rf \rangle$	2
#9 	#52 	$\langle rf \rangle$	4
#10 		$\langle f \rangle$	8
#11 		$\langle r^2 f \rangle$	4
#12 		$\langle r^2 f \rangle$	8
#13 		$\langle r^3 f \rangle$	8
#14 		$\{e\}$	16
#15 	#53 	$\{e\}$	8
#16 		$\{e\}$	16
#17 		$\{e\}$	16
#18 	#70 	$\{e\}$	8

Class	Inverse class	Stabilizer subgroup H	Class size
#19 	#54 	$\langle r^2 f \rangle$	4
#20 		$\langle r^3 f \rangle$	8
#21 		$\langle r^2 f \rangle$	8
#22 	#55 	$\{e\}$	8
#23 	#56 	$\langle r^3 f \rangle$	4
#24 	#57 	$\langle r^2, rf \rangle$	2
#25 		$\langle rf \rangle$	8
#26 	#58 	$\langle f \rangle$	4
#27 		$\langle r^2, f \rangle$	4
#28 		$\{e\}$	16
#29 		$\langle r^2 f \rangle$	8
#30		$\{e\}$	16
#31		$\langle r^2 f \rangle$	8
#32		$\langle r^3 f \rangle$	8
#33	#59	$\{e\}$	8
#34		$\{e\}$	16
#35	#60	$\{e\}$	8
#36		$\{e\}$	16
#37	#61	$\{e\}$	8

APÉNDICE B

APÉNDICE B.1

```
%% Plot Any Four Bar Linkage
%% Mohammad Y. Saadeh May, 10, 2010, University of Nevada Las Vegas
clc;clear;close all
X =   [150 110 100 90 40 120];
% X =   [180 100 185 220 55 0];
% X = [r1  r2  r3  r4  Cx  Cy ];
% r1: Crank (make sure its always the smallest, also r3+r4>=r1+r2)
% r2: Coupler
% r3: Lever (Rocker)
% r4: Frame
% Cu: x coordinate for coupler point wrt crank-coupler point
% Cv: y coordinate for coupler point wrt crank-coupler point

cycles = 2;% number of crank rotations
INCREMENTS = 100;% divide a rotation into this number
%% check the geometry
P = X(1:4);
check = P;
[L locL] = max(check);
check(locL) = [];
[S locS] = min(check);
check(locS) = [];
R = check;
flag = 0;

if S==X(4) & sum(check)>(L+S)
    TITLE = 'This is a Double-Crank Mechanism';
elseif (S==X(1)|S==X(3)) & sum(check)>(L+S)
    TITLE = 'This is a Rocker-Crank Mechanism';
elseif S==X(2) & sum(check)>(L+S)
    TITLE = 'This is a Double-Rocker Mechanism';
    flag  = 1;
elseif sum(check)==(L+S)
    TITLE = 'This is a Change Point Mechanism';
elseif sum(check)<(L+S)
    flag  = 1;
    TITLE = 'This is a Double-Rocker Mechanism';
end
%%
TH1 = linspace(0,2*pi,INCREMENTS);% Input angle theta1
dig = 10;% divide links into this number
R1 = X(1); r1 = linspace(0,R1,dig);
R2 = X(2); r2 = linspace(0,R2,dig);
R3 = X(3); r3 = linspace(0,R3,dig);
R4 = X(4); r4 = linspace(0,R4,dig);
Cu = X(5); cu = linspace(0,Cu,dig);
Cv = X(6); cv = linspace(0,Cv,dig);

%% check valid region
D = sqrt(R1^2 + R4^2 - 2*R1*R4*cos(TH1));% diagonal distance between
%                           crank-coupler point and rocker-frame point
```

```
TH5 = acos((R3^2+D.^2-R2^2)./(2*R3*D));% angle between rocker and diagonal
%                                       link (d)
IMAG = imag(TH5);
[VALUES LOCATION] = find(IMAG==0);%%
```

APÉNDICE B.2

```
IMAG = imag(TH5);
LOCATION = IMAG==0;
LOCATION1 = find(IMAG==0);
LOC = LOCATION;
n = length(LOCATION);
n1 = length(LOCATION1);
Check = 0;
direction = 1;
for i=1:n-1
    if LOC(i+1)~=LOC(i)
        if Check==0
            direction = LOC(i);
        end
        Check = Check+1;
    end
end
%%
Rotate = 0;
if isempty(LOCATION1)
    error('This is not a valid linkage');
elseif direction==0 & Check==2
    LOC1 = find(LOCATION==1);
    th1 = [TH1(LOC1) TH1(fliplr(LOC1))];
n1==n
    th1 = TH1;
elseif direction==1 & Check==2
    Rotate = 1;
    loc1 = LOC(1:end-1);
    loc2 = LOC(2:end);
    [Value deadpoint] = find((loc2-loc1)~=0);
    deadp = deadpoint + [0 1];
    LOC2 = [deadp(2):n 1:deadp(1)];
    th1 = [TH1(LOC2) TH1(fliplr(LOC2))];
elseif Check==4
    Rotate = 1;
    loc1 = LOC(1:end-1);
    loc2 = LOC(2:end);
    [Value deadpoint] = find((loc2-loc1)~=0);
    deadp1 = deadpoint(1:2) + [1 0];
    deadp2 = deadpoint(3:4) + [1 0];
    fprintf('This mechanism has two disconnected upper and lower
regions\n');
    DIREC = 1;
    DIREC = input('Select [1] for upper, [2] for lower    Default = [1]
');
    if DIREC == 1
        LOC3 = [deadp1(1):deadp1(2)];
    else
        LOC3 = [deadp2(1):deadp2(2)];
    end
```

```
    th1 = [TH1(LOC3) TH1(fliplr(LOC3))];
end
```

APÉNDICE B.3

```
d = sqrt(R1^2 + R4^2 - 2*R1*R4*cos(th1));
th5 = acos((R3^2+d.^2-R2^2)./(2*R3*d));% angle between rocker and

%%
if Rotate == 1
    d = sqrt(R1^2 + R4^2 - 2*R1*R4*cos(th1));
    th5 = acos((R3^2+d.^2-R2^2)./(2*R3*d));% angle between rocker and
diagonal link (d)
    th5 = [th5(1:end/2) -th5(end/2+1:end)];
end
Ax = R1*cos(th1);% x coordinate for the crank-coupler point
Ay = R1*sin(th1);% y coordinate for the crank-coupler point
a = R4 - R1*cos(th1);% horizontal distance between rocker-frame point and
%                       projection of crank-coupler point
b = Ay;% vertical projection of crank-coupler point
th6 = atan2(b,a);% angle between frame and diagonal link (d)
th4 = pi - th5 - th6;% angle the rocker makes with horizon
Bx = R3*cos(th4) + R4;% horizontal distance between frame-crank point and
%                        projection of coupler-rocker point
By = R3*sin(th4);% vertical projection of coupler-rocker point
th2 = atan2((By-Ay),(Bx-Ax));% angle the coupler makes with the horizon
Cx = Ax + Cu*cos(th2) - Cv*sin(th2);% horizontal projection of coupler
%                                    point wrt coupler
Cy = Ay + Cu*sin(th2) + Cv*cos(th2);% vertical projection of coupler
%                                    point wrt coupler
% calculate display (figure) limits
xmin = 1.2*min([min(Cx) -R1 -R3]);
xmax = 1.2*max([max(Cx) R4+max([R3 max(R3*cos(th4))])]);
ymin = 1.2*min([min(Cy) -R1 -R3]);
ymax = 1.2*max([max(Cy) max([R1 R3 R3+Cv])]);
for t=0:h:tfin
n=n+1;
k1=feval('amortiguado',x,k,m,c,v);
k2=feval('amortiguado',x,k,m,c,v+k1*h/2);
k3=feval('amortiguado',x,k,m,c,v-k1*h+2*k2*h);
v=v+(k1+4*k2+k3)*h/6;
x=x+h*v;
E=(m*v^2)/2;
ppt(n+1)=t;
ppx(n+1)=x;
ppv(n+1)=v;
ppa(n+1)=k1;
ppE(n+1)=E;
end
subplot(3,1,2), plot(ppx,ppE); grid on;
xlabel('desplazamiento(m)');
ylabel('energia cinetica(J)
%%
increments = length(th1);
for i=1:increments
    link1x(i,:) = r1*cos(th1(i));
    link1y(i,:) = r1*sin(th1(i));
```

```
    link2x(i,:) = linspace(Ax(i),Bx(i),dig);
    link2y(i,:) = linspace(Ay(i),By(i),dig);
    link3x(i,:) = R4 + r3*cos(th4(i));
    link3y(i,:) = r3*sin(th4(i));
    Couplx1(i,:) = linspace(Ax(i),Cx(i),dig);
    Couply1(i,:) = linspace(Ay(i),Cy(i),dig);
    Couplx2(i,:) = linspace(Cx(i),Bx(i),dig);
    Couply2(i,:) = linspace(Cy(i),By(i),dig);
end
for k=1:cycles
    for i = 1:increments
        plot(link1x(i,:),link1y(i,:),'b',link2x(i,:),link2y(i,:),'r',...
            link3x(i,:),link3y(i,:),'k',Couplx1(i,:),Couply1(i,:),'r',...
            Couplx2(i,:),Couply2(i,:),'r')
        hold on
        plot([link2x(i,:) ;Couplx1(i,:)],[link2y(i,:);
Couply1(i,:)],'g','linewidth',2)
        plot([link2x(i,:) ;Couplx2(i,:)],[link2y(i,:);
Couply2(i,:)],'g','linewidth',2)
        plot(0,0,'sk',R4,0,'sk','MarkerSize',12)
        plot(0,0,'ok',R4,0,'ok')
        plot(Couplx1(i,end),Couply1(i,end),'ok','MarkerSize',6,...
            'MarkerFaceColor','g')
                axis([xmin xmax ymin ymax])
        if Rotate == 1 & i<=increments/2
            plot(Couplx1(1:i,end),Couply1(1:i,end),'--g','linewidth',2)
        elseif Rotate == 1
            plot(Couplx1(1:increments/2,end),Couply1(1:increments/2,end),'-
-g','linewidth',2)
            plot(Couplx1(increments/2:i,end),Couply1(increments/2:i,end),'-
-r','linewidth',2)
        else
            plot(Couplx1(1:i,end),Couply1(1:i,end),'--g','linewidth',2)
        end
        clc
        title(['\bf',TITLE])
        fprintf('Th1 = %5.2f, th5 = %5.2f, D = %7.2f\n',th1(i),th5(i),d(i))
        YY = input('Hit Enter    ');
        hold off
    end
end
if Rotate==1
    hold on
    plot(Couplx1([1 end/2],end),Couply1([1
end/2],end),'hr','MarkerSize',10)
end
```

APÉNDICE B.4

```
%function data=dataset11
clc; clear; close all
dmax=max(data);
dmin=min(data);
%hallando el rango
xrango=range(data);
```

```
xmedia=mean(data);
xmedian=median(data);
%xmad=mad(data);
xstd=std(data);
xcvar=var(data);
xvar1=var(data);
xmoda=mode(data);
curtosis = kurtosis(data);
fprintf('la curtosis es  =');
disp(curtosis);
fprintf('calculo de la varianza =');
disp(xvar1);
fprintf('el calculo de la moda es  =');
disp(xmoda);
fprintf('el calculo de la varianza es cvar =');
disp(xcvar);
fprintf('la desviacion estandar stad =');
disp(xstd);
%fprintf('la desviacion media adsoluta mad =');
%&disp(xmad);
fprintf('la mediana es mediana =');
disp(xmedian);
fprintf('la media aritmetica es media =');
disp(xmedia);
fprintf('valor maximo de datos dmax=');
disp(dmax);
fprintf('valor minimo de datos dmin=');
disp(dmin);
fprintf('el rango del conjunto de datos es =');
disp(xrango);
fprintf('tabla de distribucion de frecuencia')
tb_valores____conteo___porcentages=tabulate(data)
 increments = length(th1);
for i=1:increments
    link1x(i,:) = r1*cos(th1(i));
    link1y(i,:) = r1*sin(th1(i));
    link2x(i,:) = linspace(Ax(i),Bx(i),dig);
    link2y(i,:) = linspace(Ay(i),By(i),dig);
    link3x(i,:) = R4 + r3*cos(th4(i));
    link3y(i,:) = r3*sin(th4(i));
    Couplx1(i,:) = linspace(Ax(i),Cx(i),dig);
    Couply1(i,:) = linspace(Ay(i),Cy(i),dig);
    Couplx2(i,:) = linspace(Cx(i),Bx(i),dig);
    Couply2(i,:) = linspace(Cy(i),By(i),dig);
end
for k=1:cycles
    for i = 1:increments
        plot(link1x(i,:),link1y(i,:),'b',link2x(i,:),link2y(i,:),'r',...
            link3x(i,:),link3y(i,:),'k',Couplx1(i,:),Couply1(i,:),'r',...
            Couplx2(i,:),Couply2(i,:),'r')
        hold on
        plot([link2x(i,:) ;Couplx1(i,:)],[link2y(i,:);
Couply1(i,:)],'g','linewidth',2)
        plot([link2x(i,:) ;Couplx2(i,:)],[link2y(i,:);
Couply2(i,:)],'g','linewidth'
```

```
subplot(2,2,1),hist(data);grid on; title('grafica de histogramas'
);xlabel('datos'); ylabel('frecuencia');
subplot(2,2,2),boxplot(data);grid on; title('diagrama o grafica de caja'
);xlabel('columna de numeros'); ylabel('valores');
subplot(2,2,3),pie(data);grid on;
%subplot(2,2,4),plotedit(data);grid on;
subplot(2,2,4),cdfplot(data);grid on;
        plot(0,0,'sk',R4,0,'sk','MarkerSize',12)
        plot(0,0,'ok',R4,0,'ok')
        plot(Couplx1(i,end),Couply1(i,end),'ok','MarkerSize',6,...
            'MarkerFaceColor','g')
                axis([xmin xmax ymin ymax])
        if Rotate == 1 & i<=increments/2
            plot(Couplx1(1:i,end),Couply1(1:i,end),'--g','linewidth',2)
        elseif Rotate == 1
            plot(Couplx1(1:increments/2,end),Couply1(1:increments/2,end),'-
-g','linewidth',2)
            plot(Couplx1(increments/2:i,end),Couply1(increments/2:i,end),'-
-r','linewidth',2)
```

APÉNDICE B.5

```
clear, clf, hold on;        %pregunta # 7 capitulo 4
n=0; h=0.1;
% Constantes del Sistema
k=10; m=3;
% Condiciones Iniciales
t=0; x=1; v=0; tfin=100;
%Inicio de la Simulacion
pt(1)=t; pv(1)=v; px(1)=x; pE(1)=0;
for t=0:h:tfin
n=n+1;
k1=feval('armonico',x,k,m);
k2=feval('armonico',x,k,m);
k3=feval('armonico',x,k,m);
v=v+(k1+4*k2+k3)*h/6;
x=x+h*v;
E=(m*v^2)/2;
subplot(2,2,2),boxplot(data);grid on; title('diagrama o grafica de caja'
);xlabel('columna de numeros'); ylabel('valores');
subplot(2,2,3),pie(data);grid on;
%subplot(2,2,4),plotedit(data);grid on;
subplot(2,2,4),cdfplot(data);grid on;
        plot(0,0,'sk',R4,0,'sk','MarkerSize',12)
        plot(0,0,'ok',R4,0,'ok')
        plot(Couplx1
pt(n+1)=t;
px(n+1)=x;
pv(n+1)=v;
pa(n+1)=k1;
pE(n+1)=E;
end
subplot(3,1,1), plot(px,pE); grid on;
xlabel('desplazamiento(m)');
ylabel('energia cinetica(J)');
```

```
hold on;
n=0; h=0.01;
% Constantes del Sistema
k=10; m=3; c=5;
% Condiciones Iniciales
t=0; x=1; v=0; tfin=100;
%Inicio de la Simulacion
ppt(1)=t; ppv(1)=v; ppx(1)=x; ppE(1)=0');
 subplot(2,2,2),boxplot(data);grid on; title('diagrama o grafica de caja'
);xlabel('columna de numeros'); ylabel('valores');
subplot(2,2,3),pie(data);grid on;
%subplot(2,2,4),plotedit(data);grid on;
subplot(2,2,4),cdfplot(data);grid on;
        plot(0,0,'sk',R4,0,'sk','MarkerSize',12)
        plot(0,0,'ok',R4,0,'ok')
        plot(Couplx1(i,end),Couply1(i,end),'ok','MarkerSize',6,...
            'MarkerFaceColor','g')
                axis([xmin xmax ymin ymax])
        if Rotate == 1 & i<=increments/2
            plot(Couplx1(1:i,end),Couply1(1:i,end),'--g','linewidth',2)
        elseif Rotate == 1
            plot(Couplx1(1:increments/2,end),Couply1(1:increments/2,end),'-
-g','linewidth',2)
            plot(Couplx1(increments/2:i,end),Couply1(increments/2:i,end),'-
-r','linewidth',2)

hold on;
n=0; h=0.01;
% Constantes del Sistema
k=10; m=3; c=8; Fo=3; w=1
% Condiciones Iniciales
t=0; x=1; v=0; tfin=100;
%Inicio de la Simulacion
pppt(1)=t; pppv(1)=v; pppx(1)=x; pppE(1)=0;
for t=0:h:tfin
n=n+1;
k1=feval('amortiguadoforzado',x,k,m,c,v,Fo,w,t);
k2=feval('amortiguadoforzado',x,k,m,c,v+k1*h/2,Fo,w,t+h/2);
k3=feval('amortiguadoforzado',x,k,m,c,v-k1*h+2*k2*h,Fo,w,t+h)
v=v+(k1+4*k2+k3)*h/6;
x=x+h*v;
E=(m*v^2)/2;
pppt(n+1)=t;
pppx(n+1)=x;
pppv(n+1)=v;
pppa(n+1)=k1;
pppE(n+1)=E;
end
```

APÉNDICE B.6

```
%Se muestra la imagen para que el usuario seleccione la semilla en el color
deseado
scrsz = get(0,'ScreenSize');
```

```
xs=[];
ys=[];
%Ciclo para obtener las semillas para cada color
for n=1:ncolors
fig1=figure('Name','Imagen para crecimiento','Position',[1 scrsz(4)/2
scrsz(3)/2 scrsz(4)/2]);
texto=['Selecciona la semilla con el botón derecho del mouse de color #
',num2str(n)];
imshow(img),title(texto)
[x{n} y{n}]=getpts; %Con el boton derecho del ratón la selecciono %x =
columnas %y=renglones
x{n}=int32(x{n});
y{n}=int32(y{n});
xs=[xs;x{n}];
ys=[ys;y{n}];
end
npropiedades=[]; %Crear matriz de propiedades
%Tabla en la cual buscaré el color
for n=1:ncolors tic
    x=xs(n);
    y=ys(n);

matrizaux = find(tablacolors(:,1) == n);
tablacolor=tablacolors(matrizaux,:);

%Inciialización de variables para cada color

mat_aux=zeros(nR, nC); %En esta matriz verificaré que las posiciones de los
pixeles encontrados se sustituyan
mat_aux(y,x)=1; %Primer valo sustituido que es el pixel de la primera
posición
%Inicio de la Simulacion
pt(1)=t; pv(1)=v; px(1)=x; pE(1)=0;
for t=0:h:tfin
n=n+1;
k1=feval('armonico',x,k,m);
k2=feval('armonico',x,k,m);
k3=feval('armonico',x,k,m);
v=v+(k1+4*k2+k3)*h/6;
x=x+h*v;
E=(m*v^2)/2;
pt(n+1)=t;
px(n+1)=x;

reg_size=1; %El tamaño de la región hasta el momento

free_mem=10000; %Libero memoria para almacenar las regiones encontradas
vec_list=zeros(free_mem,5); %Guardará la información de posición y valor
del pixel encontrado para después ser evaluado
list_reg=zeros(free_mem,5);
list_reg(1,:)=[x y pixel(y,x,1) pixel(y,x,2) pixel(y,x,3)];

vec_pos=0; %Contador inicial de la posición
bandera=1; %Condición que me permitirá continuar con while, la bandera 1,
significa que al menos se encontro 1 vecino con ese pixel

vec=[-1 0; 1 0; 0 -1; 0 1]; %Posiciones de los vecinos

while bandera==1     %Esta condición significa que se han encontrado
vecinos para el pixel, pues se encuentran dentro de la tabla de evaluación

    for i=1:4 167
```

```
                xn = x +vec(i,1); %Columnas
        yn = y +vec(i,2); %Renglones
        %Para revisar si el vecino está dentro de la región
                ins=(xn>=1)&&(yn>=1)&&(xn<=nC)&&(yn<=nR);

        %Añadir el pixel si no es parte de la region aun y si está dentro
de los límites
        if(ins&&(mat_aux(yn,xn)==0))
                vec_pos =vec_pos+1;
                vec_list(vec_pos,:) = [xn yn pixel(yn,xn,1) pixel(yn,xn,2)
pixel(yn,xn,3)]; %Guardará los valores de los pixeles vecinos, sin importar
si son del mismo color
                mat_aux(yn,xn)=1;
        end
    end
       %Anadir un nuevo bloque de memoria, en caso de ser necesario
     if(vec_pos+10>free_mem), free_mem=free_mem+10000;
vec_list((vec_pos+1):free_mem,:)=0; end

     %Criterios de evaluación y cómo obtener el nuevo x y y;
     if vec_pos>0
     pix=vec_list(1,3:5);
     %Evalúo el pixel en la función evalpixel
     [val]=evalpixel_ncolor(pix,tablacolor,ord(n,:));

     if val==1  %significa que si encontró al pixel en la lista, por lo
```

APÉNDICE B.7

```
tanto se agrega a la región, a la lista de regiones, se establece una nueva
X y Y y se elimina el pixel de vec_list
         reg_size=reg_size+1;
         list_reg(reg_size,:)=vec_list(1,:);
         x= vec_list(1,1); y= vec_list(1,2); %Los nuevos valores que tomara
x
         vec_list(1,:)=vec_list(vec_pos,:);  %Como eliminare el primer
valor, para que ya no sea evaluado nuevamente, hago que el último valor,
sea el primero
         vec_pos=vec_pos-1; %Después reduzco el índice para eliminar el
último elemento que ya tomó el lugar del primero
         bandera=1;
     else
         for m=2:vec_pos
             pix=vec_list(m,3:5);
             val=evalpixel_ncolor(pix,tablacolor,ord(n,:));
             pos_enlista=m;
             if val==1
                 break
             end
         end
         %Lo que me dará como resultado el ciclo anterior es el índice de
vec_list en el cual encontró el siguiente pixel que pertenece, si
         %el primero al ser evaluado es rechazado
         reg_size=reg_size+1;
         list_reg(reg_size,:)=vec_list(pos_enlista,:);
         x= vec_list(pos_enlista,1); y = vec_list(pos_enlista,2);
         %Para eliminar los pixeles que ya no interesan, porque no se
         %encontró su valor en la tabla, se realiza lo siguiente
         num_eliminados=pos_enlista-1;
         vec_list=vec_list(pos_enlista:vec_pos,:);  %Elimino todos aquellos
que no se encontraron dentro de mi tabla 168
```

```
         vec_pos=vec_pos-num_eliminados; %Después reduzco el índice
vec_pos, quitando el número de eliminados, (los cuales son todos los
anteriores al que encontré)
         bandera=1;
     end
     if val==0          bandera=0;
     else      end
     else          bandera=0;
     end end
%Eliminó de la matriz aquellos que sean igual a 0

valdifzero=find(list_reg(:,1)>0);
list_reg=list_reg(valdifzero,:);

     %La matriz que utilizaré es list_reg la cual tiene todos los valores
que pertenecen a la región en 1--> columnas x  2--->renglones y

pixpertenece=zeros(size(list_reg,1),3);

%Cliclo para generar los pixeles que pertenecen a la lista y la imagen
for k=1: size(list_reg,1)
    x=list_reg(k,1);
    y=list_reg(k,2);
    R=img(y,x,1);
    G=img(y,x,2);
    B=img(y,x,3);
    pixpertenece(k,:)=[R G B];
```

APÉNDICE B.8

```
Los vecinos para el pixel, pues se encuentran dentro de la tabla de evaluación

    for i=1:4 167
                 xn = x +vec(i,1); %Columnas
        yn = y +vec(i,2); %Renglones
        %Para revisar si el vecino está dentro de la región
                 ins=(xn>=1)&&(yn>=1)&&(xn<=nC)&&(yn<=nR);

        %Añadir el pixel si no es parte de la region aun y si está dentro
de los límites
        if(ins&&(mat_aux(yn,xn)==0))
                vec_pos =vec_pos+1;
                vec_list(vec_pos,:) = [xn yn pixel(yn,xn,1) pixel(yn,xn,2)
pixel(yn,xn,3)]; %Guardará los valores de los pixeles vecinos, sin importar
si son del mismo
    J(y,x,:)=[R G B];
    L(y,x,:)=[1 1 1];
end
pertenece=[pixpertenece list_reg(:,2) list_reg(:,1)];
if ~isempty(pertenece)
[propiedades]=[regiones(pertenece) etiqueta(1,n)];
npropiedades=[npropiedades;propiedades];
else
end
end %Cierro el ciclo de los colores


image(J),title('Imagen segmentada')
%Para mostrar ambas imágenes en la pantalla
texto=['Comparación de imáges normal vs. resultado de segmentación'];
figure('Name',texto,'Position',[1 scrsz(4)/2 scrsz(3) (scrsz(4)/2)+20]);
subplot(1,2,1)
imshow(img),title('Imagen seleccionada')
```

```
subplot(1,2,2)
image(J),title('Imagen segmentada')

%figure('Name','Imagen seccionada');
%imshow(img),title('Esta es la imagen, junto con su rectángulo')
hold on
int=1;
matiniregiones=zeros(150,9);
%Para mostrar los contornos
for r=1:size(npropiedades,1)
    Xi=npropiedades(r,2);
    Xf=npropiedades(r,4);
    Yi=npropiedades(r,3);
    Yf=npropiedades(r,5);
    ancho=(Xf-Xi)+1;
    alto=(Yf-Yi)+1;
    matiniregiones(int,:)=[int npropiedades(r,7) npropiedades(r,2:6) ancho
alto];
    rectangle('Position',[Xi,Yi,ancho,alto],'EdgeColor',[1,1,1])
    int=int+1;
end


hold off
mataux=matiniregiones(matiniregiones(:,1)>0); matregiones=matiniregiones(mataux,:);
tf=toc;     for i = 1:increments
        plot(link1x(i,:),link1y(i,:),'b',link2x(i,:),link2y(i,:),'r',...
            link3x(i,:),link3y(i,:),'k',Couplx1(i,:),Couply1(i,:),'r',...
            Couplx2(i,:),Couply2(i,:),'r')
        hold on
        plot([link2x(i,:) ;Couplx1(i,:)],[link2y(i,:);
Couply1(i,:)],'g','linewidth',2)
        plot([link2x(i,:) ;Couplx2(i,:)],[link2y(i,:);
Couply2(i,:)],'g','linewidth'

subplot(2,2,1),hist(data);grid on; title('grafica de histogramas'
);xlabel('datos'); ylabel('frecuencia');

subplot(3,1,3), plot(pppx,pppE); grid on;
xlabel('desplazamiento(m)');
ylabel('energia cinetica(J)');
hold off
```

APÉNDICE B.9

```
end
pertenece=[pixpertenece list_reg(:,2) list_reg(:,1)];
if ~isempty(pertenece)
[propiedades]=[regiones(pertenece) etiqueta(1,n)];
npropiedades=[npropiedades;propiedades];
else
```

```
end
end %Cierro el ciclo de los colores


image(J),title('Imagen segmentada')
%Para mostrar ambas imágenes en la pantalla
texto=['Comparación de imáges normal vs. resultado de segmentación'];
figure('Name',texto,'Position',[1 scrsz(4)/2 scrsz(3) (scrsz(4)/2)+20]);
subplot(1,2,1)
imshow(img),title('Imagen seleccionada')
subplot(1,2,2)
image(J),title('Imagen segmentada')

%figure('Name','Imagen seccionada');
%imshow(img),title('Esta es la imagen, junto con su rectángulo')
hold on
int=1;
matiniregiones=zeros

 subplot(2,2,2),boxplot(data);grid on; title('diagrama o grafica de caja'
);xlabel('columna de numeros'); ylabel('valores');
subplot(2,2,3),pie(data);grid on;
%subplot(2,2,4),plotedit(data);grid on;
subplot(2,2,4),cdfplot(data);grid on;
        plot(0,0,'sk',R4,0,'sk','MarkerSize',12)
        plot(0,0,'ok',R4,0,'ok')
        plot(Couplx1(i,end),Couply1(i,end),'ok','MarkerSize',6,...
            'MarkerFaceColor','g')
                axis([xmin xmax ymin ymax])
        if Rotate == 1 & i<=increments/2
            plot(Couplx1(1:i,end),Couply1(1:i,end),'--g','linewidth',2)
        elseif Rotate == 1
            plot(Couplx1(1:increments/2,end),Couply1(1:increments/2,end),'-
-g','linewidth',2)
            plot(Couplx1(increments/2:i,end),Couply1(increments/2:i,end),'-
-r','linewidth',2)
fprintf('la curtosis es  =');
disp(curtosis);
fprintf('calculo de la varianza =');
disp(xvar1);
fprintf('el calculo de la moda es  =');
disp(xmoda);
fprintf('el calculo de la varianza es cvar =');
disp(xcvar);
fprintf('la desviacion estandar stad =');
disp(xstd);
%fprintf('la desviacion media adsoluta mad =');
%&disp(xmad);
fprintf('la mediana es mediana =');
disp(xmedian);


%K=x^2*k/2

%subplot(3,1,1), plot
```

APÉNDICE B.10

```
tanto se agrega a la región, a la lista de regiones, se establece una nueva
X y Y y se elimina el pixel de vec_list
         reg_size=reg_size+1;
         list_reg(reg_size,:)=vec_list(1,:);
         x= vec_list(1,1); y= vec_list(1,2); %Los nuevos valores que tomara
x
         vec_list(1,:)=vec_list(vec_pos,:);  %Como eliminare el primer
valor, para que ya no sea evaluado nuevamente, hago que el último valor,
sea el primero
         vec_pos=vec_pos-1; %Después reduzco el índice para eliminar el
último elemento que ya tomó el lugar del primero
         bandera=1;
     else
         for m=2:vec_pos
             pix=vec_list(m,3:5);
             val=evalpixel_ncolor(pix,tablacolor,ord(n,:));
             pos_enlista=m;
             if val==1
                 break
             end
         end
pt(n+1)=t;
px(n+1)=x;
pv(n+1)=v;
pa(n+1)=k1;
pE(n+1)=E;

         %Lo que me dará como resultado el ciclo anterior es el índice de
vec_list en el cual encontró el siguiente pixel que pertenece, si
         %el primero al ser evaluado es rechazado
         reg_size=reg_size+1;
         list_reg(reg_size,:)=vec_list(pos_enlista,:);
         x= vec_list(pos_enlista,1); y = vec_list(pos_enlista,2);
         %Para eliminar los pixeles que ya no interesan, porque no se
         %encontró su valor en la tabla, se realiza lo siguiente
         num_eliminados=pos_enlista-1;
         vec_list=vec_list(pos_enlista:vec_pos,:);  %Elimino todos aquellos
que no se encontraron dentro de

pt(1)=t; pv(1)=v; px(1)=x; pE(1)=0;
for t=0:h:tfin
n=n+1;
k1=feval('armonico',x,k,m);
k2=feval('armonico',x,k,m);
k3=feval('armonico',x,k,m);
v=v+(k1+4*k2+k3)*h/6;
x=x+h*v;
E=(m*v^2)/2;
subplot(2,2,2),boxplot(data);grid on; title('diagrama o grafica de caja'
);xlabel('columna de numeros'); ylabel('valores');
subplot(2,2,3),pie(data);grid on;
%subplot(2,2,4),plotedit(data);grid on;
subplot(2,2,4),cdfplot(data);grid on;
         plot(0,0,'sk',R4,0,'sk','MarkerSize',12)
         plot(0,0,'ok',R4,0,'ok')
         plot(Couplx1
pt(n+1)=t;
px(n+1)=x;
```

APÉNDICE B.11

```
function recvoc_OpeningFcn(hObject, eventdata, handles, varargin)
load geno.mat;
load rosa.mat;
load flavio.mat;
load andres.mat;
load 'handel';
wavwrite(y,'handel');
rec=reconociendovoz(y0);
handles.voz_grabada=y;
handles.feature_datos=rec
handles.feature_f0=f0;
handles.feature_f1=f1;
handles.feature_f2=f2;
handles.feature_f3=f3;
handles.feature_y0=y0;
handles.feature_y1=y1;
handles.feature_y2=y2;
handles.feature_y3=y3;
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% --- Outputs from this function are returned to the command line.
function varargout = recvoc_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
67
varargout{1} = handles.output;
% --- Executes on button press in Grabar.
function Grabar_Callback(hObject, eventdata, handles)
% hObject handle to Grabar (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% obtener la palabra del microfono
Fs = 11025; % Sampling Frequency (Hz)
n = 20;
Nseconds = 2; % largo de la señal de voz
y = wavrecord(Nseconds*Fs, Fs, 1)';
rec1=reconociendovoz(y);
handles.voz_grabada = y;
handles.feature_datos=rec1
plot(y);
msgbox('Grabacion terminada');
guidata(hObject, handles);
%load rec.mat;
%handles.feature_voz=f;
% --- Executes on button press in identificar.
function identificar_Callback(hObject, eventdata, handles)
% hObject handle to identificar (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
rec1=handles.feature_datos;
y0=handles.feature_y0;
```

```
y1=handles.feature_y1;
y2=handles.feature_y2;
y3=handles.feature_y3;
f0=handles.feature_f0;
f1=handles.feature_f1;
f2=handles.feature_f2;
f3=handles.feature_f3;
68
d1=distancia(f0,rec1)
d2=distancia(f1,rec1)
d3=distancia(f2,rec1)
d4=distancia(f3,rec1)
if (d1 < d2) & (d1 < d3) & (d1 < d4)
 plot(abs(fft(y0)));
 soundsc(y0);
 msgbox('Usted dijo: Geno');
 else if (d2 < d1) & (d2 < d3)& (d2 < d4)
 plot(abs(fft(y1)));
 soundsc(y1);
 msgbox('Identificado: Rosa')
 else if (d3 < d1) & (d3 < d2) & (d3 < d4)
 plot(abs(fft(y2)));
 soundsc(y2);
 msgbox('Identificado: Flavio');
 else if (d4 < d1) & (d4 < d2) & (d4 < d3)
 plot(abs(fft(y3)));
 soundsc(y3);
 msgbox('Identificado: Andres');
 end
 end
 end
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
exit;
```

APÉNDICE B.12

```
clc
Fs=11025;
y0=wavrecord(1*Fs,Fs,1)';
soundsc(y0);
word0=['geno'];
y0=chop_silencio(y0);
z0=enfasis(y0);
seg0=segmentos(z0);
c0=cepstrum(seg0);
f0=features(c0);
save geno.mat word0 f0 y0;
y1=wavrecord(1*Fs,Fs,1)';
soundsc(y1);
word1=['rosa'];
y1=chop_silencio(y1);
z1=enfasis(y1);
```

```
seg1=segmentos(z1);
c1=cepstrum(seg1);
f1=features(c1);
save rosa.mat word1 f1 y1;
y2=wavrecord(1*Fs,Fs,1)';
soundsc(y2);
word2=['flavio'];
y2=chop_silencio(y2);
z2=enfasis(y2);
seg2=segmentos(z2);
c2=cepstrum(seg2);
62
f2=features(c2);
save flavio.mat word2 f2 y2;
y3=wavrecord(1*Fs,Fs,1)';
soundsc(y3);
word3=['andres'];
y3=chop_silencio(y3);
z3=enfasis(y3);
seg3=segmentos(z3);
c3=cepstrum(seg3);
f3=features(c3);
save andres.mat word3 f3 y3;
%chop_silencio
%Corta el silencio en la señal completa
%y=chop_silencio(s)
function y = chop_silencio(s)
 len = length(s); % length del vector
 avg_e = sum(s.*s)/len; %promedio señal entera
 THRES = 0.2;

 y = [0];
 for i = 1:80:len-80 % cada 10ms
 seg = s(i:i+79); % segmentos
 e = sum(seg.*seg)/80; % promedio de cada segmento
 if( e> THRES*avg_e) % si el promedio energetico es mayor que la señal
%completa por el valor umbral
 y=[y,seg(1:end)]; % almacena en y sino es eliminado como espacio en
blanco
 end;
 end
```

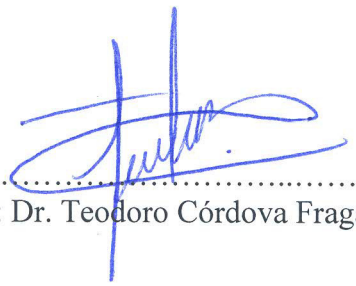
APÉNDICE B.13

```
%features
%Encuentra los coeficientes polinomiales de la expansion correspondientes al
los
%coeficientes cepstrum
%[ftr]=features(c)
function ftr=features(c)
 [M,N] = size(c);
 ftr = zeros(M-8,20);

 j = (1:9); %col vector
 p1 = repmat((j-5)',1,10); %crea una matriz llena de P1j=j-5
 sum_p1 = sum((j-5).*(j-5)); % suma total
 for i = 1:M-8
 %caracteristicas de orden cero (coeficientes cepstrales)
 ftr(i,1:10)=c(i,1:10);
 %caracteristicas de primer orden
```

```
 b = zeros(1,10);
 for j = 0:8 % aqui se sacan los coeficientes polinomiales para cada
segmento
 b = b+ p1(j+1,:).*c(i+j,:); % numerador de la ecuacion
 end;
 ftr(i,11:20)=b/sum_p1; % dividiendolo en el denominador
 end; %se obtienen los valores de b
cubeindex = 0;
   if (grid.val[0] < isolevel) cubeindex |= 1;
   if (grid.val[1] < isolevel) cubeindex |= 2;
   if (grid.val[2] < isolevel) cubeindex |= 4;
   if (grid.val[3] < isolevel) cubeindex |= 8;
   if (grid.val[4] < isolevel) cubeindex |= 16;
   if (grid.val[5] < isolevel) cubeindex |= 32;
   if (grid.val[6] < isolevel) cubeindex |= 64;
   if (grid.val[7] < isolevel) cubeindex |= 128;
/* Find the vertices where the surface intersects the cube */
   if (edgeTable[cubeindex] & 1)
      vertlist[0] =
         VertexInterp(isolevel,grid.p[0],grid.p[1],grid.val[0],grid.val[1]);
   if (edgeTable[cubeindex] & 2)
      vertlist[1] =
         VertexInterp(isolevel,grid.p[1],grid.p[2],grid.val[1],grid.val[2]);
   if (edgeTable[cubeindex] & 4)
      vertlist[2] =
         VertexInterp(isolevel,grid.p[2],grid.p[3],grid.val[2],grid.val[3]);
   if (edgeTable[cubeindex] & 8)
      vertlist[3] =
         VertexInterp(isolevel,grid.p[3],grid.p[0],grid.val[3],grid.val[0]);
   if (edgeTable[cubeindex] & 16)
      vertlist[4] =
         VertexInterp(isolevel,grid.p[4],grid.p[5],grid.val[4],grid.val[5]);
   if (edgeTable[cubeindex] & 32)
      vertlist[5] =
         VertexInterp(isolevel,grid.p[5],grid.p[6],grid.val[5],grid.val[6]);
   if (edgeTable[cubeindex] & 64)
      vertlist[6] =
         VertexInterp(isolevel,grid.p[6],grid.p[7],grid.val[6],grid.val[7]);
   if (edgeTable[cubeindex] & 128)
      vertlist[7] =
         VertexInterp(isolevel,grid.p[7],grid.p[4],grid.val[7],grid.val[4]);
   if (edgeTable[cubeindex] & 256)
      vertlist[8] =
         VertexInterp(isolevel,grid.p[0],grid.p[4],grid.val[0],grid.val[4]);
   if (edgeTable[cubeindex] & 512)
      vertlist[9] =
         VertexInterp(isolevel,grid.p[1],grid.p[5],grid.val[1],grid.val[5]);
   if (edgeTable[cubeindex] & 1024)
      vertlist[10] =
         VertexInterp(isolevel,grid.p[2],grid.p[6],grid.val[2],grid.val[6]);
            VertexInterp(isolevel,grid.p[3],grid.p[7],grid.val[3],grid.val[7]);
```

AUTORES



.....

ASESOR: Dr. Teodoro Córdova Fraga

.....

AUTOR: Lic. José Zacarías Huamaní Luna

UNIVERSIDAD DE GUANAJUATO

León, GTO; 26 de septiembre de 2019

Dr. Delepine, David Yves Ghislain

Estimado Professor Delepine,

Por este medio pongo a su consideración el jurado para evaluar el trabajo de José Zacarías Huamani Luna, estudiante de la maestría en Física, cuyo título de trabajo es: Procesamiento de Imágenes Digitales: 2D a 3D.

Presidente:	Dr. Cristian Gómez Solís	DCI	lienquidremo@hotmail.com
Secretario:	Dr. Rafael Guzmán Cabrera	CISIS	guzmanc81@gmail.com
Vocal:	Dr. Juan Carlos Martínez Espinosa	IPN	jcmartineze@ipn.mx
Suplente:	Dr. José Alfredo Soto Álvarez	DCI	jasa@fisica.ugto.mx

Agradezco de antemano su fina atención a la presente

Dr. Teodoro CORDOVA-FRAGA
Profesor Titular
Perfil PRODEP
SNI Nivel II

C.c.p. Archivo

División de Ciencias e Ingenierías, Campus León
Loma del Bosque 103, Fracc. Lomas del Campestre, 37150 León, GTO, Mx
Tel. 477 788-5100 Ext. 8454, . theo@fisica.ugto.mx . http://www.dci.uGTO,mx

UNIVERSIDAD DE GUANAJUATO

Asunto: Carta
León, GTO, 27 Febrero 2020

DR. David Yves Ghislain Delepine
Director
División de Ciencias e Ingenierías,
Universidad de Guanajuato campus León

Estimado Dr. David

Por este medio le notifico que he leído y dado mis comentarios al C. José Zacarias Huamani Luna sobre su trabajo de tesis. El C. José Zacarias Huamani Luna realizo los cambios sugeridos y tomando en cuenta cada detalle sobre su trabajo de tesis.

Por lo tanto, confirmo que el C. José Zacarias Huamani Luna puede realizar los tramites para presentar su trabajo de tesis en las fechas que se establezcan.

Sin más por el momento me despido de usted agradeciendo su atención y quedo a sus órdenes.

ATENTAMENTE
"LA VERDAD OS HARÁ LIBRES"

Dr. Christian Gómez Sólis
Profesor Titular A
Departamento de Ingeniería Física
Universidad de Guanajuato

C.c.p. Archivo

UNIVERSIDAD DE
GUANAJUATO

Salamanca, Gto. 7 de febrero del 2020.

Dr. Delepine David Yves Ghislain
Director de la División de ciencias e ingenierías
Campus León, Universidad de Guanajuato.
Presente.

Distinguido Dr. Delepine,

Por medio de la presente le envío un cordial saludo y le comento que el estudiante de maestría en Física: José Zacarias Humani Luna, atendió las observaciones realizadas en la revisión de su trabajo de tesis. También le comento que, en mi opinión, dicha tesis cuenta con el nivel y calidad necesarios para ser de maestría.

Sin mas por el momento, agradezco su amable atención y me despido reiterándole la mas distinguida de mis consideraciones.

Atentamente:

Rafael Guzmán Cabrera
Profesor Titular B
DICIS, IGTO

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"
175 Aniversario de la Escuela Superior de Comercio y Administración
125 Aniversario de la Escuela Nacional de Medicina y Homeopatía
80 Aniversario del CECyT 6 "Miguel Othón de Mendizábal"
75 Aniversario de la Escuela Nacional de Biblioteconomía y Archivonomía

Silao de la Victoria Guanajuato, 12 de febrero de 2020

DR. DELEPINE DAVID YVES GHISLAIN
DIRECTOR DE LA DIVISION DE CIENCIAS E INGENIERIAS
CAMPUS LEON, UNIVERSIDAD DE GUANAJUATO

P R E S E N T E

Me permito informarle que el estudiante de la maestría en física José Zacarías Humani Luna, atendió en tiempo y forma las observaciones que un servidor hizo sobre su trabajo de tesis. Finalmente le comento que dicho trabajo de tesis cuenta con los elementos necesarios para tener el nivel y calidad de la maestría en física.

Sin más por el momento aprovecho la ocasión para enviarle un cordial saludo y quedo a sus ordenes para cualquier duda.

ATENTAMENTE

"LA TÉCNICA AL SERVICIO DE LA PATRIA"

DR. JUAN CARLOS MARTINEZ ESPINOSA
Profesor TC Instituto Politécnico Nacional - UPIIG

UNIVERSIDAD DE GUANAJUATO

Asunto: Aprobación de Tesis de Posgrado
León, GTO., 26 de Febrero de 2020

Dr. DAVID Y. G. DELEPINE
Director de la División de Ciencias e Ingenierías
Campus León

Estimado Dr. Delepine

Por este conducto me permito informarle que he leído con atención y presentado las correcciones pertinentes a la tesis titulada "***Procesamiento de Imágenes digitales: 2D a 3D aplicado a tumores cancerigenos***", que para obtener el título de Maestró en Física presenta el C. José Zacarías Huamani Luna. La tesis describe una metodología para análisis de imágenes médicas obtenidas por tomografía utilizando la tecnicas de Marching Cube para de este modo generar una imagen en tres dimensiones a partir de una imagen bidimensional, aplicado a tumores. Considero que el trabajo satisface ampliamente los requisitos propios de este nivel académico, por lo que recomiendo que el C. José Zacarías realice la defensa pública de su tesis de maestría.

Sin más por el momento, reitero a usted mi disposición para cualquiera aclaración y aprovecho la ocasión para enviarle un cordial saludos

ATENTAMENTE
"LA VERDAD OS HARÁ LIBRES"

Dr. José Alfredo Soto Álvarez
Profesor - Investigador
Departamento de Ingeniería Física

División de Ciencias e Ingenierías, Campus León
Loma del Bosque 103, Fracc. Lomas del Campestre, 37150 León, GTO, Mx
Tel. 477 788-5100 Ext. 8454, jasa@fisica.ugto.mx http://www.dci.ugto.mx