BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Escáner 3D mediante Triangulación y Luz Estructurada para Reconstrucción de Piezas Arqueológicas

Tesis que presenta:

Francisco Calyecac Marreros

Para obtener el Título de:

INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

Asesor de Tesis:

Dr. Manuel Isidro Martín Ortiz

Junio 2009

RESUMEN

Actualmente existen dispositivos de escaneo (reconstrucción) tridimensional que permiten obtener modelos 3D de objetos del mundo real de manera digital utilizando distintas técnicas de captura, sin embargo su costo es elevado. Razón por la que en este trabajo de tesis se implementó un sistema de escaneo 3D de bajo costo sin mucha pérdida de detalle sobre el modelo 3D reconstruido. Se utilizaron las técnicas de triangulación y luz estructurada, que en su conjunto consisten en la proyección de un patrón de luz, en este caso es un haz de luz láser en forma de línea, sobre el objeto a escanear y con ayuda de una cámara se obtiene un video a diferentes ángulos (sobre 360°). Basándose en la información bidimensional del video con ayuda de la luz estructurada puede obtenerse la ubicación tridimensional y construirse un modelo virtual 3D del objeto. Particularmente, los objetos a reconstruir corresponden a piezas arqueológicas pequeñas, teniendo como objetivo la difusión y preservación de la riqueza cultural de manera digital.

AGRADECIMINETOS

A mis papás, Rafael Calyecac y Paola Marreros, por su gran apoyo, compañía, consejos, amor, regaños, por sus principios de trabajo y lucha constante. Este trabajo se los dedico como una forma de gratitud por ayudarme a llegar a esta primer meta, mi carrera profesional.

A mi hermana, Adriana Calyecac, por su amor y por estar conmigo en todo momento a pesar de todo.

Al Dr. Manuel Martín por su asesoría y tiempo brindado durante el presente trabajo y por ser un ejemplo a seguir.

A mi knovia, Minerva Díaz, por ser una fuente de inspiración y no simplemente profesional.

ÍNDICE

Contenido *Pág.*

RESUMEN II

AGRADECIMINETOS III

ÍNDICE III

LISTA DE FIGURAS V

INTRODUCCIÓN VII

1 GRÁFICOS 3D 1

1.1 FUNDAMENTOS 2
1.2 MODELADO 3D 3
1.2.1 Modelado con Superficies 5
1.2.1 Modelado con Polígonos 8
1.3 LA IMPORTANCIA DE LA ILUMINACIÓN 11
1.4 CARAS OCULTAS 13

2 DIGITALIZACIÓN 17

2.1 DIGITALIZACIÓN EN EL MUNDO 3D 19
2.2 ESCÁNER 3D 20
2.2.1 Activos 22
2.2.2 Pasivos 25
2.3 APLICACIONES DE LA DIGITALIZACIÓN 3D 26

3 RECONSTRUCCIÓN BASADA EN LUZ 28

3.1 LUZ LÁSER 28
3.2 CÓDIGOS BINARIOS 33

4 PLANTEAMIENTO Y ANÁLISIS DEL PROBLEMA 37

4.1 INTRODUCCIÓN 37
4.2 OBJETIVOS 37
4.3 ESPECIFICACIONES DEL SISTEMA 38
4.3.1 Justificación de la Técnica a Implementar 38
4.3.2 Descripción General 38
4.3.3 Alcance y Restricciones 38
4.3.4 Requerimientos Funcionales 39
4.3.5 Requerimientos no Funcionales 40
4.3.6 Suposiciones 40

5 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA 41

5.1 DISEÑO 41
5.1.1 Hardware 41

5.1.2 Software *43*
5.2 IMPLEMENTACIÓN 48
5.2.1 Hardware *48*
5.2.1 Software *50*
5.3 LÓGICA DEL SISTEMA 54
6 PRUEBAS Y VERIFICACIÓN 55
6.1 RESULTADOS 55
CONCLUSIONES 62
Limitaciones *62*
Perspectivas, trabajo futuro *63*
BIBLIOGRAFÍA 64
APÉNDICE A 66
MANUAL DE USUARIO 66

LISTA DE FIGURAS

Número *Pág.*

Figura 1.1 Ubicación de un píxel en el monitor. *2*

Figura 1.2 Sistema coordenado tridimensional. *3*

Figura 1.3 (Izquierda) Parche de una superficie. (Derecha) Parche de un polígono. *4*

Figura 1.4 a) Vaca modelada con una maya de triángulos. b) Wire-Frame. *4*

Figura 1.5 Curva de Bezier con 7 puntos de control. *6*

Figura 1.6 Superficie de Bezier con 16 puntos de control. *7*

Figura 1.7 B-Spline Uniforme No Racional. *8*

Figura 1.8 Que si es un polígono. *9*

Figura 1.9 Redundancia de datos. *9*

Figura 1.10 Almacenamiento de triángulos tipo strip. *10*

Figura 1.11 Almacenamiento de triángulos tipo fan. *10*

Figura 1.12 Estructura de dato tipo Winged-Edge. *11*

Figura 1.13 Objetos sin y con iluminación y efectos de sombras. *11*

Figura 1.14 a) Ley de Reflexión. B) Reflexión Especular y Reflexión Difusa. ... *12*

Figura 1.15 (Izq.) Sin eliminar caras ocultas. (Der.) Algoritmo aplicado. *14*

Figura 1.16 Técnica del Pintor. *15*

Figura 2.1 Dispositivo de Cargas Acopladas (CCD) *18*

Figura 2.2 Escáner 3D de contacto (ZEISS) para uso industrial *20*

Figura 2.3 Range Images sin alineación. *21*

Figura 2.4 Oclusión de partes importantes para la digitalización. *21*

Figura 2.5 Algunos ejemplos de luz estructurada. *22*

Figura 2.6 Escáner láser comercial tipo time-of-flight. *23*

Figura 2.7 Esquema de un escáner láser por triangulación. *24*

Figura 2.8 Esquema de un radar o sonar. *24*

Figura 2.9 Patrón de interferencia moiré. *25*

Figura 2.10 Esquema de un sistema de Estereovisión a partir de dos puntos de vista. *26*

Figura 3.1 Algunos tipos de proyecciones mediante láser. *28*

Figura 3.2 Geometría básica de un escáner láser 3D por triangulación. Los ejes y, Y son perpendiculares a la hoja. *29*

Figura 3.3 Escáner láser 3D por triangulación con objeto giratorio. *30*

Figura 3.4 Escáner láser 3D por triangulación con objeto giratorio. 31
Figura 3.5 Vista superior del frame i 32
Figura 3.6 Códigos binarios en su representación matricial..... 33
Figura 3.7 Proyección de un patrón de códigos binarios. 33
Figura 3.8 [23] Deformación de los stripes sobre el objeto. 34
Figura 3.9 Principio de triangulación con luz estructurada (códigos binarios) y escena estática. 35
Figura 5.1 Arquitectura abstracta del sistema. 41
Figura 5.2 Lente óptico cilíndrico que realiza la refracción de luz y genera una haz tipo línea. 42
Figura 5.3 Diagrama de estados. 43
Figura 5.4 Diagrama de clases. 45
Figura 5.5 Detección de un stripe. 46
Figura 5.6 Construcción de parches triangulares en sentido de las manecillas del reloj. 48
Figura 5.7 Implementación de la plataforma giratoria. 49
Figura 5.8 Implementación de la fuente de luz láser con lente refractor. 49
Figura 5.9 Interfaz de EstractorDePuntos. 51
Figura 5.10 Interfaz del Visualizador 3D. 52
Figura 5.11 Diagrama de estados del Sistema de Digitalización 3D. 54
Figura 6.1 Alineación del sistema de digitalización 3D. 55
Tabla 1: Características de 4 objetos escaneados. 56
Figura 6.2 Objetos a escanear. 57
Tabla 2: Configuraciones previas al escaneo. 58
Figura 6.3 Modelo 3D del escuincle (1). 58
Figura 6.4 Modelo 3D de la cabeza olmeca (2), graficado wireframe. 59
Figura 6.5 Modelo 3D de la vasija (3). 59
Figura 6.6 Modelo 3D del búho (4). 60
Tabla 3: Número de parches de los modelos escaneados sin y con suavizado, y su respectivo tiempo de graficado. 60
Figura 6.7 (Izq.) Vasija sin suavizar. (Der.) Vasija suavizada. 61
Figura A.1 Interfaz de la aplicación Escáner 3D (seleccionando CCD). 66
Figura A.2 Acotando región de búsqueda. 67
Figura A.3 Configuraciones previas al escaneo. 68
Figura A.4 Interfaz del visualizador 3D. 69
Figura A.5 Visualizando un bebé en wireframe. 69
Figura A.6 Bebé en wireframe con colores aleatorios. 70

INTRODUCCIÓN

No muchos países cuentan con la riqueza histórica cultural como la que posee México. El origen de este proyecto fue la difusión y conservación de la cultura prehispánica mexicana y la falta de disponibilidad monetaria para adquirir un costoso digitalizador 3D comercial.

Mediante la reconstrucción tridimensional de piezas arqueológicas de manera digital, es más fácil y accesible poder difundir la riqueza histórica mexicana no solo a nivel nacional sino a todo el mundo, y desde luego, se logra la preservación de estos modelos a lo largo del tiempo.

El objetivo fundamental de esta tesis es la implementación de un Escáner 3D (sistema) tanto la parte del hardware como del software en tiempo real, buscando en todo momento economizar la construcción del escáner sin pérdida de la relación costo-beneficio.

Los primeros dos capítulos de los seis que contiene este trabajo son fundamentos teóricos, el tercero también, con un poco de análisis. El primero trata sobre diferentes conceptos de los gráficos 3D tales como, la iluminación de las escenas, el tipo de modelado 3D así como sus ventajas y desventajas, entre otros. También indica las subáreas que se han ido desprendiendo de los gráficos por computadora. El segundo capítulo aborda el concepto de digitalización desde un punto de vista general para ir acercándose al qué de un escáner 3D. Por otro lado, el tercer capítulo se enfoca a dos técnicas de reconstrucción tridimensional de objetos del mundo real basados en luz.

En el capítulo cuatro se realiza el planteamiento del problema, se fijan las metas y objetivos de esta tesis quedando en claro qué es lo

que se desea realizar. Una vez delimitado el problema, en el capítulo cinco se trabaja con el diseño e implementación del sistema de escaneo 3D. Por otro lado, en capítulo seis se muestran los resultados obtenidos con el sistema desarrollado, también se comentan los problemas que surgieron durante su realización.

Finalmente, se encuentran las conclusiones de este trabajo y se citan los eventos donde ha sido presentado. Del mismo modo se comentan las limitaciones y los trabajos futuros que pudieran desarrollarse.

Capítulo 1

1 GRÁFICOS 3D

En la actualidad al referir tercera dimensión se aborda el concepto de *gráficos por computadora* (un vertiginoso subcampo de las Ciencias de la Computación), que hace referencia al uso de una computadora para la creación y manipulación de imágenes [1].

Las siguientes subáreas de este campo son las más citadas en la literatura existente:

- **Modelado:** trata las especificaciones matemáticas de las propiedades de la forma y apariencia de los objetos y como éstas deben ser almacenadas en la computadora.

- **Renderizar** (adaptación al Castellano del vocablo inglés *Rendering*): es el proceso final que lleva a cabo una computadora al ejecutar algoritmos matemáticos complejos para poder generar una imagen lo más realista posible a partir de una escena tridimensional.

- **Animación:** técnica para crear una ilusión de movimiento a partir de una secuencia de imágenes.

- **Realidad Virtual:** son intentos de inmersión al usuario a un mundo virtual tridimensional donde el principal objetivo es lograr la sensación de sentir. Para ello se requieren gráficos avanzados y sofisticados aparatos tecnológicos.

- **Visualización:** son intentos para dar al usuario una visión de cómo podrían ser las cosas.

- **Procesamiento de Imágenes:** es la manipulación de imágenes bidimensionales.

- **3D Scanning:** se vale de aparatos para crear modelos tridimensionales.

1.1 Fundamentos

El ***pixel*** es el elemento más pequeño de una imagen digital, es su unidad de medición; el monitor de una computadora es su medio de visualización y este también cuenta con sus propios pixeles, que no son sino puntos de luz que pueden tomar diferentes colores. Así pues, las imágenes se forman como una matriz rectangular de pixeles. El modelo de color tradicional que adopta el píxel en los gráficos por computadora es el llamado RGB, obtiene un color a partir de la mezcla de 3 colores primarios, rojo-verde-azul, cada color puede tomar 256 tonalidades, siendo representados físicamente a nivel hardware por 8 bits cada uno [2, 3]. La ubicación de cada píxel tanto en la imagen como en el monitor es semejante al sistema coordenado bidimensional con la única diferencia que los valores del eje Y están invertidos, tal como lo muestra la figura 1.1.

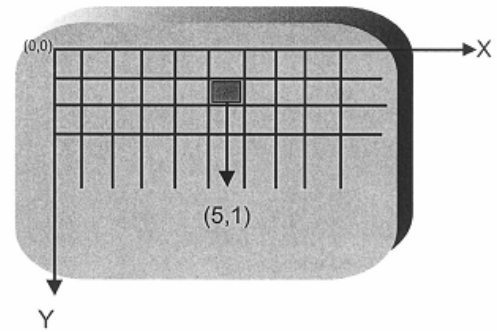


Figura 1.1 Ubicación de un píxel en el monitor.

Inicialmente los gráficos por computadora (GC) trabajan sobre el sistema coordenado bidimensional, es decir, con figuras planas, así como con sus operaciones, que son transformaciones que se le aplican a la imagen tales como escalamiento, rotación y traslación a través de las *matrices homogéneas*.

Lo más interesante de los GC está en el espacio tridimensional, es decir, en el plano (X,Y,Z), ver figura 1.2. Las matemáticas en el mundo 3D son una elegante extensión del mundo 2D [2]. Los conceptos en 3D son un desarrollo de los 2D a través de una simple adición de eje, a saber Z, que representa la profundidad de los objetos. Ahora las transformaciones deberán ser sobre los tres ejes. El objetivo primordial de los GC en el mundo 3D es lograr el *realismo visual* valiéndose desde las relativamente sencillas operaciones matemáticas extendidas antes citadas hasta las más complejas técnicas de procesamiento y almacenamiento de información de los objetos. Las producciones 3D están basadas en tres importantes bloques: *el modelado*, que se encarga de crear el objeto 3D; *el renderizado*, intenta transformar una escena 3D en una imagen realista; y *la animación*, que es la creación de la sensación de movimiento sobre imágenes renderizadas.

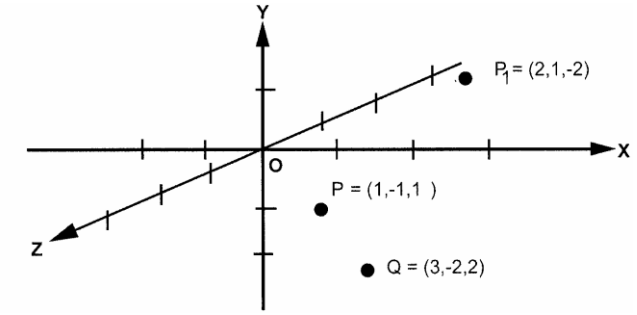


Figura 1.2 Sistema coordenado tridimensional.

1.2 Modelado 3D

El pilar base de los gráficos 3D es el ***modelado*****,** que es el *proceso de crear un modelo 3D* por computadora [2].

Existen dos corrientes importantes para modelar, el modelado a partir de *superficies* y el más socorrido, el modelado a partir de *sólidos/polígonos*. En estas dos corrientes existe un elemento en común llamado ***parche***, que dependiendo de su uso y forma recibe un nombre diferente. Computacionalmente es más complicado trabajar con superficies debido a los complejos métodos matemáticos implícitos, que hacerlo con polígonos.

Un parche se define como una superficie no necesariamente plana que se forma mínimamente con tres puntos, que por obvias razones es un *parche triangular* y en el modelado poligonal un polígono es un parche, la figura 1.3 muestra un ejemplo de ello.

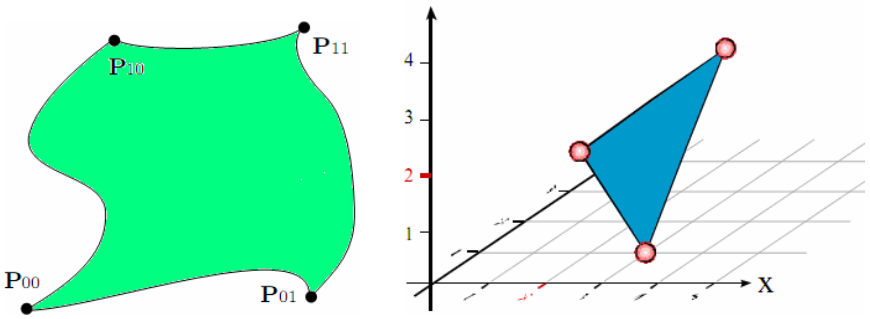
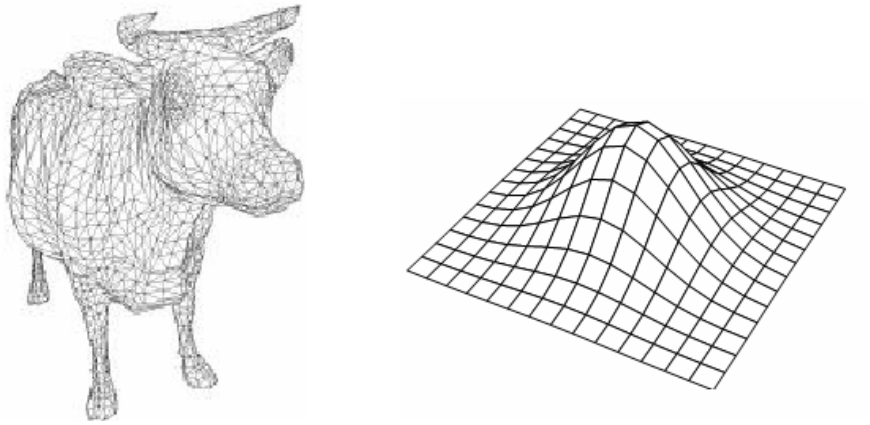


Figura 1.3 (Izquierda) Parche de una superficie. (Derecha) Parche de un polígono.

Un parche por si mismo ya es una superficie o un polígono. La *conexión de parches* individuales permite la construcción de modelos 3D tan complejos como se desee [3, 4].

En ambas imágenes de la figura 1.2 se tiene una malla, mejor conocida por el vocablo inglés como ***mesh,*** en el primer caso es una malla de triángulos y en el segundo se le conoce como ***wire-frame.***



a) b)

Figura 1.4 a) Vaca modelada con una maya de triángulos. b) Wire-Frame.

Tanto en el modelado de superficies como de polígonos existen *primitivas* tales como el cubo, la esfera, el cono, el cilindro, la famosa tetera entre otros, que permiten un fácil desarrollo para objetos más complejos.

1.2.1 Modelado con Superficies

Existen diversos métodos matemáticos de interpolación y aproximación para la modelación de superficies [1, 4, 5]. La más utilizada e implementada son los NURBS. A continuación se enlistan algunos de ellos:

- **Superficies Splines**: En esta categoría se encuentran los muy utilizados *NURBS* (del inglés Non-Uniform Rational Basis Spline).

- **Superficies de Bezier**: (se detalla más adelante).

- **Subdivisión de Poliedros**: Implementa una aproximación distinta. Su fundamento está en alterar los vértices de un poliedro base, agregando otros durante el proceso y subdividiendo los lados. Sus inconvenientes son, el difícil manejo para el usuario y la complicada implementación.

- **Método de Hermite**: No se basa en aproximaciones, sino interpolaciones, donde utiliza como funciones de peso a los polinomios de la familia de Hermite, y lo interesante, no solo actúa sobre los puntos de control, lo que permite mayor manejo de la superficie.

- **Superficie de Coons**: Fue uno de los primeros métodos propuestos en los albores de la modelación de superficies. Se basa en la determinación de cuatro curvas que definan las fronteras de la superficie

Para hablar de las superficies de Bezier primeramente se abordan las características de las curvas de este mismo tipo ya que las superficies simplemente son una extensión en $\mathbb{R}^3$.

Este tipo de curvas (Bezier) se calculan por aproximación a una serie de puntos llamados *puntos de control* también llamados vértices, de esta manera se consi-

guen curvas más suaves que si se realiza con interpolación. El número de puntos de control $n+1$ determina el grado del polinomio, a saber n.

Bezier calcula las coordenadas de un punto de la curva a partir del peso de los puntos de control que influyen en él. Ese peso está dado por la distancia al punto de control y una distribución normal ordinaria [4]. Así pues, el punto se obtiene sumando esos pesos. En la figura 1.5 se observan los puntos de control.

Por tanto, dados $n+1$ puntos de control, $\mathrm{P}_0, \mathrm{P}_1, ..., \mathrm{P}_n$ por sus coordenadas cartesianas (x_i, y_i), el polinomio de Bezier es:

$$\mathrm{B}(t) = \sum_{i=0}^{n} b_{i,n}(t)\,\mathrm{P}_{i,} \qquad t \in [0,1] \qquad (1)$$

Donde

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-1}, \qquad i = 0,...n \qquad (2)$$

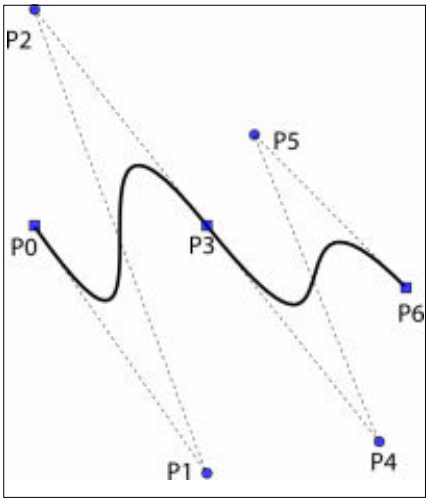


Figura 1.5 Curva de Bezier con 7 puntos de control.

Un inconveniente de este método es su naturaleza global, es decir, todos los puntos de la curva son afectados por los puntos de control, esto implica que toda la curva se altere con tan solo mover un punto de control.

Ahora, para modelar superficies de Bezier se usan dos conjuntos de curvas de este tipo, como lo muestra la figura 1.6. Se deben introducir $(m+1)\cdot(n+1)$ puntos de control, y la superficie se obtiene como el producto cartesiano de las curvas. La fórmula se muestra a continuación, $k_{i,j}$ denota a los puntos de control:

$$p(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}\mathrm{B}_i^n(u)\mathrm{B}_j^m(v)k_{i,j} \qquad (3)$$

Donde $\mathrm{B}_i^n(u)$ y $\mathrm{B}_j^m(v)$ se evalúan independientemente:

$$\mathrm{B}_i^n(u)=\binom{n}{i}u^i(1-u)^{n-i} \qquad (4)$$

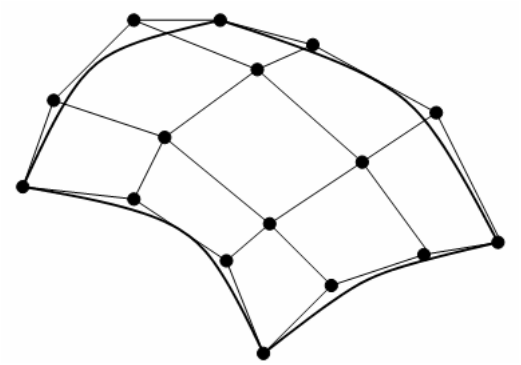


Figura 1.6 Superficie de Bezier con 16 puntos de control.

Ante el inconveniente de Bezier por su naturaleza global antes mencionada, surgió una mejora, los ***NURBS*** (Non-Uniform Rational Basis Spline), que en la actualidad son los implementados en librerías de gráficos para lenguajes de programación y en software de graficación asistida. Las principales mejoras fueron la implementación de un *vector nodal (knot vector)* y la *racionalización* a los puntos de control, permitiendo que los B-Splines sean más generales brindándole al usuario la posibilidad de manipular un mayor número de parámetros que favorecen en la obtención deseada de la forma de la curva o superficie. La racionalización se refiere a la *homogenización de las coordenadas* de los knot mediante la adición de un cuarto elemento $W(t)$ en la ecuación paramétrica

$Q(t) = \left(X(t), Y(t), Z(t), W(t)\right)$ [2]. Una consecuencia positiva de esto es la invariación ante la rotación, escalamiento y traslación. Esto significa que las transformaciones deben ser únicamente para los puntos de control y entonces reevaluar la curva o superficie.

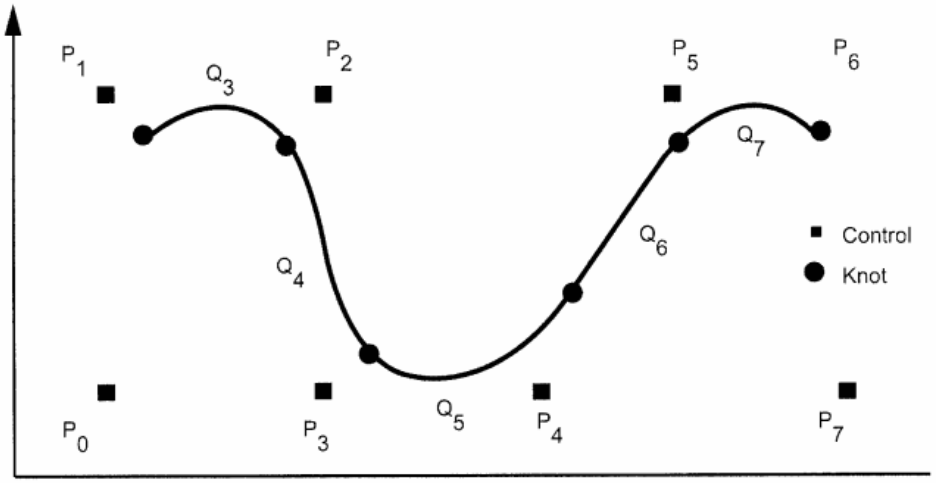


Figura 1.7 B-Spline Uniforme No Racional.

En la figura 1.7 se aprecian los puntos de control representados por los cuadros y los nodos por los círculos. Cada segmento de curva del spline $Q_i, 3 \le i \le n$ está definido por cuatro puntos de control $\mathrm{P}_{i-3}, \mathrm{P}_{i-2}, \mathrm{P}_{i-1}, \mathrm{P}_i$ [2]. Así por ejemplo, de la figura 1.7, el segmento de curva Q_3 está definido por los puntos de control $\mathrm{P}_0, \mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3$. En otras palabras, si P_4 se mueve solo afectará a los segmentos Q_3, Q_4, Q_5 y Q_6.

1.2.1 Modelado con Polígonos

Un polígono se define al menos con tres vértices y cada línea del polígono que une dos vértices se llama *borde* (o arista, pero mejor conocido en inglés como *edge*). Formalmente se puede decir que un polígono es un conjunto de líneas que no se cruzan y están unidas por puntos coplanares que encierran un área simple convexa [1], la figura 1.8 lo muestra.

Enfatizando, ya se ha dicho que un polígono es un parche, por lo que su unión con otros polígonos pueden llegar a formar diferentes tipos de objetos 3D.

La mayoría de los modelos del mundo real están compuestos de complejas formaciones de triángulos que comparten vértices entre sí [1]. Reforzando la definición de parche, a esto se le conoce como ***mallas triangulares*** o como ***TINs*** (del inglés **T**riangular **I**rregular **N**etwork**s**). Ahora, el siguiente paso es *¿dónde almacenar esa información?*, para que el objeto creado pueda ser manipulado con posterioridad y que esa información ocupe una cantidad razonable de espacio y pueda hacer uso del manejo de texturas con eficiencia.

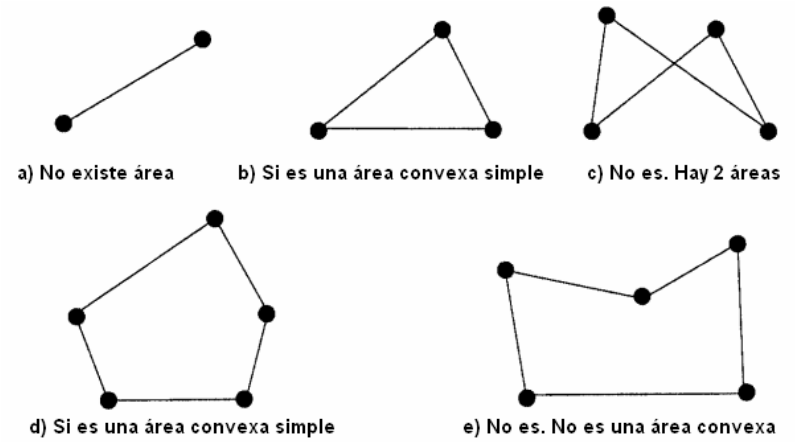


Figura 1.8 Que si es un polígono.

Un primer acercamiento fue almacenar los vértices de cada uno de los triángulos que conforman la malla, sin embargo existe redundancia de datos ya que varios triángulos comparten varios vértices, tal como se ve en la figura 1.9, y la adyacencia no está bien definida.

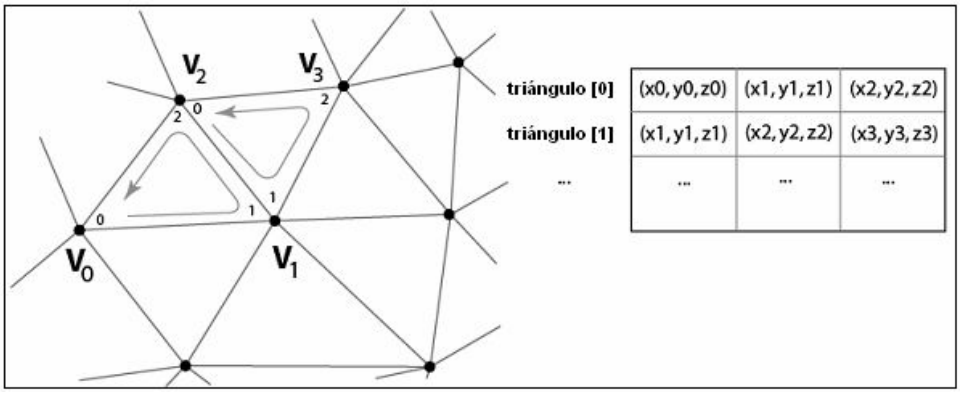


Figura 1.9 Redundancia de datos.

Ante este inconveniente se optó por la *indexación de triángulos*, que consiste en el manejo de dos listas, en una se almacenan las posiciones de todos los vértices y en la otra lista los triángulos donde solo se indican los tres vértices que lo forman. Sin embargo, otras dos alternativas surgieron, que fueron los triángulos tipo ***strip*** y tipo ***fan*****,** donde la idea central es el *reuso de bordes* (edges), esto se puede apreciar mejor en las figuras 1.10 y 1.11.

Sin embargo, cuando se desea conocer más sobre las relaciones que existen entre los vértices, bordes y caras, simplemente no se puede [1]. Cabe señalar que una *cara* (en inglés *face*) es casi un polígono, o bien, un parche. De hecho, un polígono tiene dos caras, la anterior y la posterior.

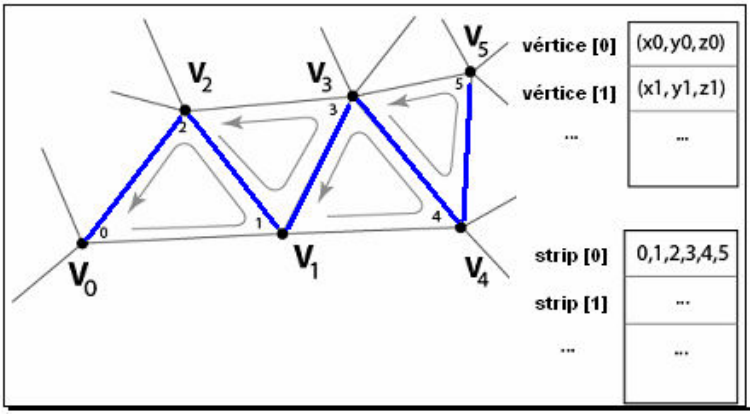


Figura 1.10 Almacenamiento de triángulos tipo *strip*.

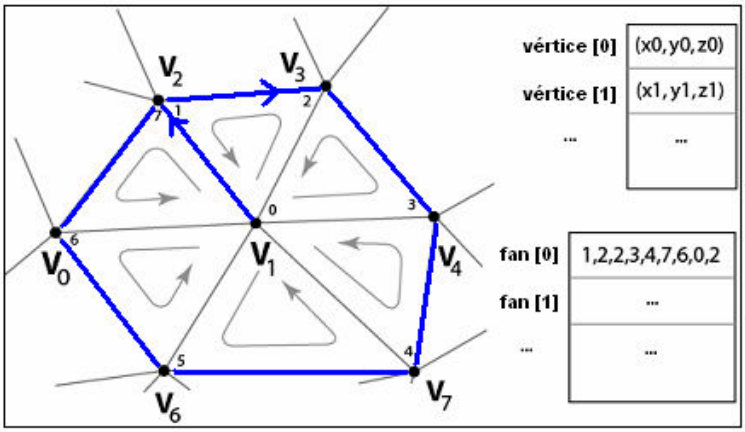
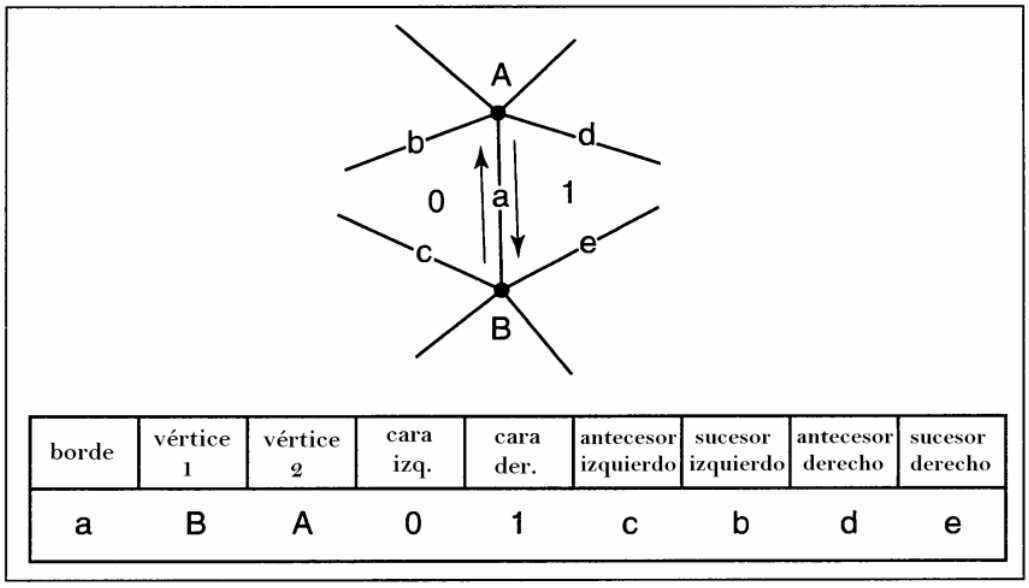


Figura 1.11 Almacenamiento de triángulos tipo *fan*.

Obsérvese como la estructura de datos de *Winged-Edge*, figura 1.12, permite conocer diversas relaciones de utilidad que existen entre los bordes, vértices y

caras que conforman una malla. Por ejemplo, se puede saber que cara es la colindante a determinado borde, o cuales son los vértices que conforman ese borde.



borde	vértice 1	vértice 2	cara izq.	cara der.	antecesor izquierdo	sucesor izquierdo	antecesor derecho	sucesor derecho
a	B	A	0	1	c	b	d	e

Figura 1.12 Estructura de dato tipo *Winged-Edge*.

1.3 La Importancia de la Iluminación

La iluminación y el degradado de luz (este último conocido como *shading*) son importantes herramientas para crear imágenes gráficas que muestren realismo y sean entendibles. Estas herramientas aportan vitales aspectos visuales sobre la curvatura y orientación de las superficies [6], esto se aprecia en la figura 1.13.

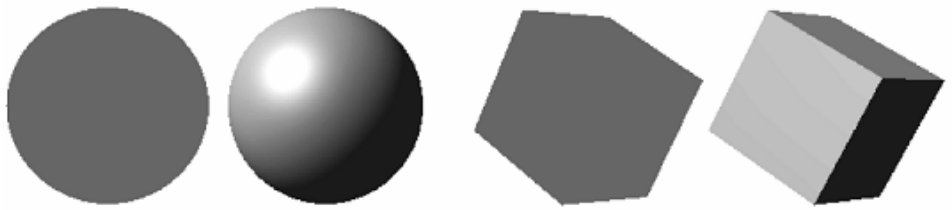


Figura 1.13 Objetos sin y con iluminación y efectos de sombras.

Los modelos de iluminación en los GC están basados en una aproximación modular donde el usuario (generalmente programador) especifica las posiciones y propiedades de las fuentes de luz, e independientemente, indica las propiedades de las caras en interacción con los materiales. Las propiedades de las luces y los materiales se correlacionan para crear la iluminación, color y degradado a partir de un punto de visión.

La luz es modelada como rayos de luz (matemáticamente como *vectores*) y se puede decir que viajan sobre un haz. Lo mejor de esto, es la predicción de su comportamiento [2, 7]. Un rayo de luz al chocar con una superficie plana adopta la *ley de reflexión*, figura 1.14 a); es por eso la vitalidad de la existencia del *vector normal* de la superficie. Un vector normal, es un vector unitario, es decir, tiene norma 1; y es perpendicular a la superficie. Así pues, la luz incidente está representada por el vector ***I*** y la luz de rebote por el vector ***R,*** el ángulo ***θ*** es el mismo con respecto al vector normal ***N***. Es importante aclarar que la reflexión de luz puede ser de dos tipos: ***especular***, cuando la superficie es plana, entonces se comporta como espejo; y ***difusa***, cuando la superficie es irregular, entonces los rayos reflejados salen en todas direcciones y solo se conserva la energía no la imagen, véase en la figura 1.14 b).

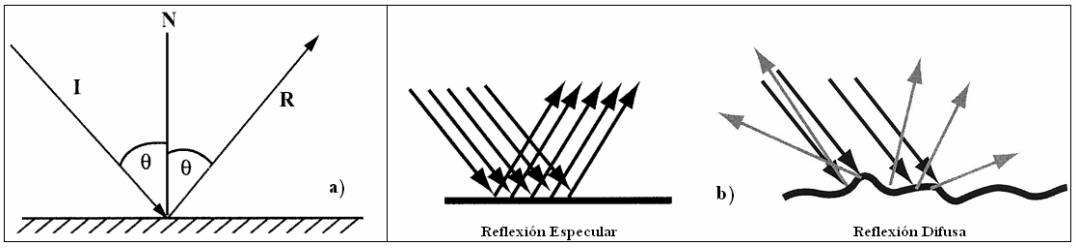


Figura 1.14 a) Ley de Reflexión. B) Reflexión Especular y Reflexión Difusa.

Las técnicas más populares para el degradado-iluminación (*shading*) son dos, la *Interpolación de Gouraud* (1971) y la *Interpolación de Pong* (1975), hacen posible que un modelo con caras planas luzca suavizado. Particularmente el modelo Pong, permite una fácil implementación en hardware y software, razón por la que casi universalmente es utilizada en sistemas gráficos de tiempo real [6].

El modelo Pong provee de propiedades a los materiales que utiliza cada superficie [6, 7]. Las propiedades de los materiales permiten controlar el cuanto y como

será la iluminación de la superficie. A excepción del *exponente especular*, estas propiedades pueden establecerse independientemente para cada componente del color. No olvidar que el modelo de color utilizado es el RGB.

- **Propiedades de la Reflexión Especular:** proporciona un *coeficiente de reflexividad especular* p_s, que controla la cantidad de reflexión especular. Ofrece también un *exponente especular* f, que determina el brillo de la superficie a través de la manipulación de la anchura de la propagación de la luz reflejada.

- **Propiedades de la Reflexión Difusa:** tiene un *coeficiente de reflexividad especular* p_d, que regula la intensidad relativa de la luz difusa reflejada.

- **Propiedades de la Reflexión Ambiental:** también cuenta con un *coeficiente de reflexividad ambiental* p_a.

- **Propiedades de Emisión:** el nivel de emisión de una superficie controla la cantidad de luz que *emite* la superficie en ausencia de cualquier tipo de luz. La luz emitida no afecta a otras superficies como lo hacen las fuentes de luz.

1.4 Caras Ocultas

En toda escena tridimensional se encuentra más de un objeto, cuando se crea una imagen a partir de ella para poderla mostrar al usuario (a este proceso se le conoce como *rendering*), es muy importante determinar el orden en el que se van a mostrar los objetos visibles en el campo de visión. Con esto se logra el ahorro de numerosos y complejos cálculos matemáticos al momento de hacer un rendering. A este proceso se le conoce como *eliminación de caras ocultas* (conocido en inglés como *Back-Face Culling*).

Los algoritmos de eliminación de caras ocultas se dividen en dos familias [8]:

- **Por precisión de objeto**: determinan que superficies de la escena tridimensional vería el observador en la región de visión.

- **Por precisión de imagen**: decide que superficie de las que se proyectan sobre un pixel dado verá el observador y asigna al pixel el color de la superficie.

Se enfatiza que, estos métodos actúan sobre una unidad mínima que es la ***cara*** de un objeto, no el objeto en sí; por ejemplo un cubo tiene 6 caras, entonces el algoritmo se aplica de manera individual a cada una de ellas. Para aclarar este concepto se muestra la figura 1.15, donde a la imagen de la izquierda no se le ha aplicado ningún algoritmo, mientras que en la imagen de la derecha sí.

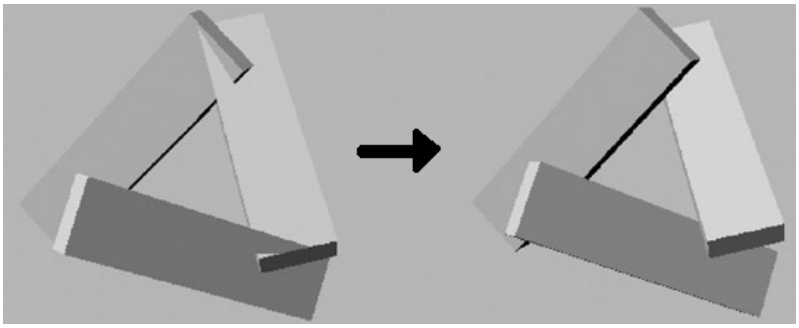


Figura 1.15 (Izq.) Sin eliminar caras ocultas. (Der.) Algoritmo aplicado.

Un algoritmo clásico para este problema dentro de la segunda familia es el de ***Wright***, que esta diseñado para representar en pantalla funciones de dos variables de la forma $z = f(x, y)$. La superficie se obtiene aproximada por polígonos formando una malla que sigue su contorno.

Un método de la primera familia relativamente sencillo de implementar es también llamado ***eliminación de caras ocultas o posteriores***, según la traducción de *hidden surface removal*. El inconveniente de esta técnica es la falta de determinación de que objeto tapa a otro objeto. Sin embargo reduce drásticamente el número de caras a procesar, es decir, indica o descarta las caras de los objetos que no son visibles. La idea central de su funcionamiento recae en el ángulo formado entre el vector normal de la cara el vector de visión. La normal debe tener sentido saliente de la cara, es decir, el producto escalar entre la normal y el vec-

tor de visión debe ser positivo, el ángulo formado será agudo; en caso contrario, será negativo y la cara se descarta.

A continuación se describe un algoritmo de la segunda familia que resulta de la mezcla de la técnica ***ordenación en profundidad*** mejor conocido como la ***técnica del pintor*** (figura 1.16), con la de ***árboles binarios de espacio particionado (árboles BSP)***. La idea central en ambas técnicas es la ordenación relativa de los objetos, para después pintarlos de atrás hacia delante de acuerdo al punto de visión dado. El inconveniente del algoritmo del pintor es que lo hace para un punto de visión definido, contrario a los árboles BSP, que realizan antes que nada un pre-procesamiento de ordenación binaria de los objetos, hecho esto se aplica la técnica del pintor sin importar el punto de visión, ya que los objetos están ordenados, solo cambiaría la forma de recorrer el árbol binario para saber que objeto graficar primero.

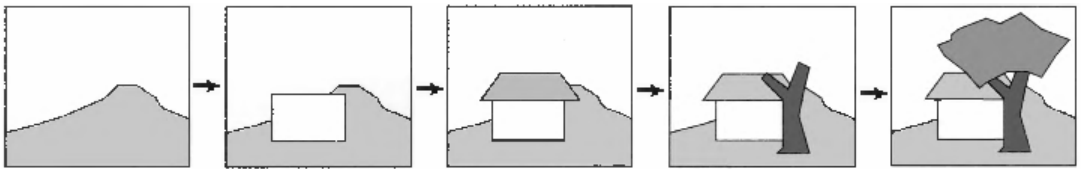


Figura 1.16 Técnica del Pintor.

El ***Buffer-Z*** es otro algoritmo de la segunda familia el cual es de fácil implementación a nivel hardware y software. El algoritmo consiste en dibujar en cada pixel (del plano de visión) la superficie de la escena 3D con menor z_p, es decir, la que tenga menor distancia al plano de visión. Así pues dado un objeto arbitrario de una escena se proyecta cada uno de los vértices (en el plano de visión) y se dibuja el polígono resultante, la manera de rellenar ese polígono es trazando líneas horizontales consecutivas y es en este preciso momento cuando se lleva a cabo la decisión de visibilidad punto a punto (según z_p). Este modelo necesita un buffer de tamaño igual a la resolución del monitor.

Finalmente, el algoritmo *Traza de Rayos* mejor conocido como ***Ray Tracing***, pertenece a la segunda familia. Dada la escena 3D, la posición del observador y la posición de la pantalla, se trazan rayos desde el ojo del observador hasta la escena pasando por la pantalla, donde se encuentran los pixeles. Si el rayo no intersecta con ningún objeto, se asigna al pixel el color de fondo; en caso contrario, se

le asigna el color del objeto. Cabe recalcar que un sentido más amplio el Ray Tracing se extiende a otros conceptos que dan origen al realismo en la presentación de gráficos realizados por computadora.

Capítulo 2

2 DIGITALIZACIÓN

Frecuentemente se utilizan modelos matemáticos para describir imágenes o algún tipo de señal. La señal no es más que un tipo de función matemática que depende de ciertas variables. Se dice que una *función* puede ser *continua* o *discreta* dependiendo de sus conjuntos de dominio y rango. Particularmente, si el *dominio* y *rango* son discretos entonces se dice que es una ***función digital*** [9].

Hoy en día la tecnología avanza vertiginosamente y uno de sus principales logros ha sido la creación y mejoramiento de la computadora. Como ya se sabe, el funcionamiento de la computadora se basa en el sistema binario, es decir, el conjunto se restringe únicamente a dos valores ***0*** (cero) y ***1*** (uno), entonces el conjunto ya está *discretizado*, de ahí que se hable de la *era digital*. Así pues, la ***digitalización*** es el *proceso de conversión* de información (análoga) tal como imágenes, texto, audio, video, señales multidimensionales y más, en ***formato digital***, de esta manera la computadora puede procesar y manipular esta información. La digitalización también ofrece la *preservación*, el *fácil acceso* y la *compartición de* dicha *información*.

La computadora por si misma no puede realizar la digitalización, necesita de dispositivos físicos (*digitalizadores*) adicionales para esta conversión. Debido a que la naturaleza de los objetos a digitalizar no es la misma, consecuentemente, la de los dispositivos tampoco. Por ejemplo, para digitalizar el audio o video es necesaria una tarjeta (de *audio* o *video*) especial que procese la información de manera eficiente. Por otro lado, para obtener digitalmente la representación de la superficie marina se hace uso de radares sonares, su funcionamiento se basa en emitir sonidos a la superficie y sensar el tiempo de rebote y posteriormente se procesa la información obtenida par poderla modelar en 3D.

Otro objeto ineludible en la digitalización y en este trabajo es la imagen. La adquisición de imágenes tuvo un gran auge con la invención del *dispositivo de cargas eléctricas acopladas* (***CCD*** del inglés ***c**harge-**c**oupled **d**evice*), este dispositivo es más barato, más pequeño y más rápido que sus antecesores [10], ver figura 2.1. Un CCD es un arreglo de capacitores muy cercanos entre si, estos convierten la luz (fotones) incidente en voltaje análogo y posteriormente se ***cuantifica*** para poder ser discretizados, es decir, se le asigna un valor proporcional a la luz entrante. Actualmente un CCD sofisticado puede llegar a tener más de 500 mil capacitores en un área de 1 cm^2 [11]; cabe señalar que cada uno de estos capacitores representa un *pixel*.

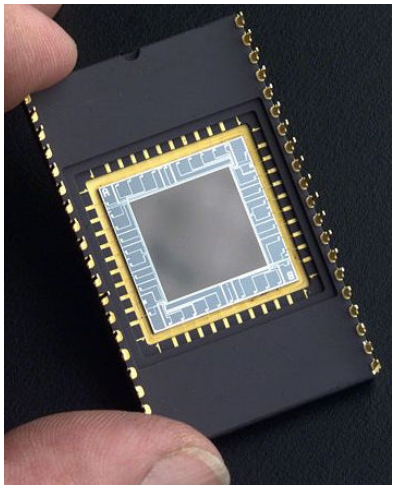


Figura 2.1 Dispositivo de Cargas Acopladas (CCD)

Así pues, una imagen digital $a[m,n]$ pertenece a un espacio discreto 2D (bidimensional) derivada de una imagen análoga $a(x,y)$ en un espacio continuo.

Los principales dispositivos que hacen uso de los CCD son las cámaras de video y fotográficas y los escaners de plancha. En la actualidad los fabricantes de estos aparatos compiten por la mejora de la calidad de las imágenes y las operaciones funcionales que se les puede hacer al momento de capturarlas.

2.1 Digitalización en el mundo 3D

Con los avances tecnológicos y científicos en los campos de procesamiento de imágenes, telecomunicaciones y gráficos por computadora durante la última década se ha dado paso a una nueva *multimedia*, principalmente a los ***datos digitales tridimensionales*** [12]. Esta multimedia 3D está directamente relacionada con la aparición de las *técnicas de adquisición 3D (**escáner 3D**)* y que aunado a los avances de rendering en gráficos 3D se ha impulsado a la creación de *archivos totalmente digitales* con modelos 3D en los diferentes campos de la ciencia, tales como la medicina, la herencia cultural, la arqueología, el reconocimiento de rostros, los videojuegos, en el diseño asistido por computadora (*CAD*), entre otros.

Se debe tener presente que los objetos 3D son más complicados de manipular a comparación de las imágenes, video o señales de audio por ejemplo. Más aún, los objetos 3D pueden tener una diferente representación según el medio de adquisición, mientras que las imágenes (2D) por ejemplo, que sin importar el tipo de dispositivos de adquisición su representación siempre es la misma, un arreglo bidimensional $(m \times n)$.

Un objeto 3D presenta las siguientes dificultades:

- Las diferentes fuentes de datos 3D (p/e una tomografía o escáner láser), no generan la misma representación.

- Las diferentes aplicaciones (p/e CAD, médicas) no trabajan sobre las mismas representaciones.

- La conversión de representaciones es un poco compleja y trae consigo problemas que aún no se resuelven del todo.

El objetivo fundamental de crear un objeto 3D es la reproducción de un objeto del mundo real. A groso modo se puede decir que hay dos tipos de tecnologías: las que *analizan y reconstruyen* el objeto, como un ***escáner 3D***; y las que *reconstruyen su volumen*, como una tomografía [12,13].

2.2 Escáner 3D

Un escáner 3D es un dispositivo físico que tras la implementación de hardware, software y técnicas específicas, se obtiene generalmente una nube de puntos que describen tridimensionalmente la forma de un objeto del mundo real de manera digital.

Existen varios parámetros para clasificar un escáner 3D. Sin embargo, desde el nivel más superior se puede decir que hay escáner 3D de ***contacto*** (*ó máquinas de medición por coordenadas*) y ***no contacto***. En el primer caso, el escáner debe forzosamente tener contacto físico con el objeto de interés, trayendo como consecuencias la lentitud en la adquisición de datos y el muy posible daño al objeto dependiendo de la composición de su material y la fuerza que aplique el escáner, generalmente se trata de un brazo electro-mecánico. Sin embargo, por la naturaleza de algunos objetos con el interés de ser digitalizados, este tipo de escaners son la mejor opción; generalmente utilizados en la industria, p/e ZEISS [14] fabrica este tipo de escáner, ver figura 2.2.

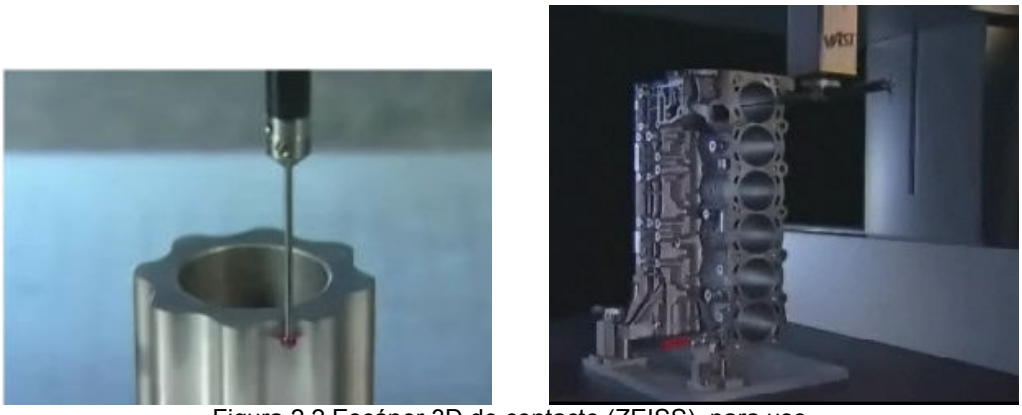


Figura 2.2 Escáner 3D de contacto (ZEISS) para uso industrial.

Las investigaciones que se llevan a cabo y los escaners 3D implementados hasta la fecha, en su mayoría son de no contacto.

El corazón de una cámara tradicional es el CCD, sin embargo, existen artefactos parecidos (costosos) que proporcionan una imagen especial porque incluyen un sensor extra (***range sensor*** [13]), que proporciona la *profundidad* del pixel, es decir, aparte de almacenar los tres componentes del color (RGB) también alma-

cena la distancia a la cual se encuentra un punto de la escena. A este tipo de imagen se le conoce como *range image*. Usualmente este tipo de adquisición de datos se emplea para objetos 3D complejos, por lo que se requiere de múltiples escaneos (conocido en inglés como ***range images***) desde diferentes ángulos, donde el trabajo principal se centra en el ***registro*** y ***alineación*** de los datos obtenidos, es decir, hacer las operaciones necesarias (p/e traslación y rotación) para que los datos tengan el mismo sistema coordenado 3D. En la figura 2.3 [12] se aprecia un range images *sin* alineación. Es claro que esta herramienta se utiliza en los escaners de *no* contacto.

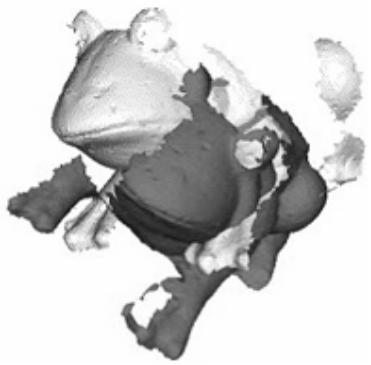


Figura 2.3 Range Images sin alineación.

Una desventaja inmediata que presentan los escaners de no contacto es la dificultad para analizar los hoyos que pudiesen tener los objetos, como en el caso de la figura 2.2, o bien, las oclusiones (figura 2.4) de ciertas partes a otras debido a la posición de la cámara.

Figura 2.4 Oclusión de partes importantes para la digitalización.

2.2.1 Activos

Ahora bien, los digitalizadores 3D de *no contacto* se dividen en ***activos*** y ***pasivos*** [13]. Los primeros emiten algún tipo de energía (p/e un patrón de luz ó pulsos sonares) sobre la escena y detectan su posición para llevar a cabo la medición, o bien, aprovechan las variaciones de los principios de la física con ayuda de sensores especiales (p/e el enfoque de un lente óptico (conocido en inglés como *focus*) o un radar). Por otro lado, los segundos solo se basan en la intensidad de luz de las imágenes para poder obtener la profundidad y así lograr la reconstrucción (p/e la estereovisión).

Para la implementación de un escáner activo se hace uso de diferentes técnicas, primordialmente depende del tipo de tecnología que se utilice y es así como recibe su nombre, se tiene a los escaners de *luz estructurada*, *láser* (que bien podría estar contenido en el anterior), *por triangulación*, *radar o sonar*, *interferometría Moiré* y *focusing/defocusing*. A continuación se describen con más detalle.

Los escaners 3D que emplean ***luz estructurada*** se basan en la proyección de un patrón de luz sobre la escena u objeto de interés y una cámara (CCD) que registre los sucesos. A lo largo de las investigaciones al respecto se han utilizado un sin fin de patrones de luz, argumentando las ventajas que cada uno provee, entre los más implementados está la cuadrícula binaria o de colores así como líneas horizontales o verticales (figura 2.5 [15,16]), particularmente en la imagen derecha se utilizan código de grises. El cálculo de la profundidad consiste en resolver las intersecciones del plano-recta de la proyección. Otra característica importante es que la proyección utilizada cubre al objeto completamente.

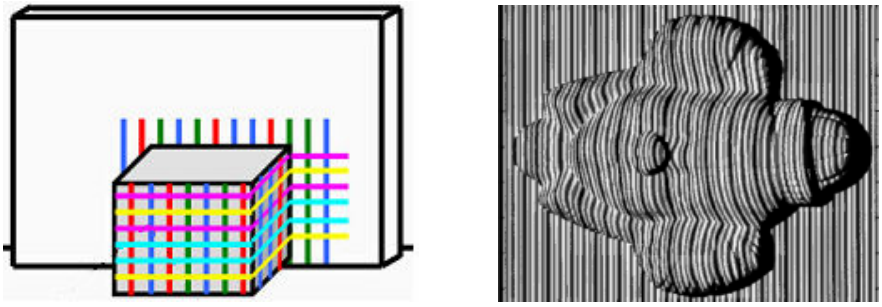


Figura 2.5 Algunos ejemplos de luz estructurada.

Los ***escáner 3D láser*** son un subconjunto de los de luz estructurada ya que también emite un patrón de luz, que bien puede ser una línea, uno o varios puntos, entre otros, sin embargo, por su gran importancia, utilidad y variantes que se han creado, se considera como una técnica independiente. Esencialmente existen dos tipos de tecnología, los llamados ***time-of-flight*** (o *laser range finder* [12]), que se trata del mismo concepto de *range images* (citado párrafos arriba), en este caso se emite un haz de luz láser para ser reflejado en el objeto y del cual se deduce la distancia a partir de la velocidad-tiempo que le toma al haz en rebotar. Este tipo de escaners ofrecen buenos resultados para escenas complejas, de tamaño considerable y largas distancias (p/e más de 50 metros). Como ejemplo tenemos al escáner comercial de la figura 2.6 [17] fabricado por la empresa *Laser Digital Inc*.

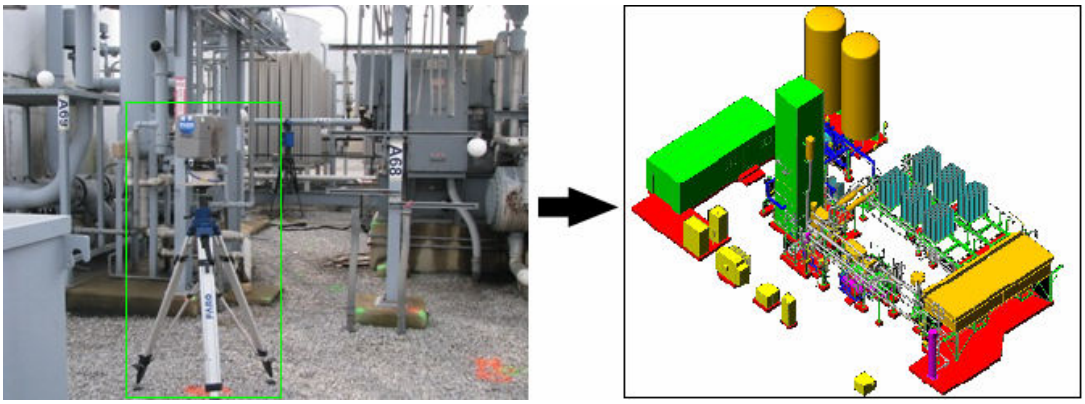


Figura 2.6 Escáner láser comercial tipo time-of-flight.

El otro tipo de escáner *láser es* por ***triangulación***, en este, también se lanza un haz de luz láser hacia el objeto y se usa una cámara (CCD) para analizar la ubicación de ese haz láser sobre el objeto. Se da el nombre de escáner por triangulación porque la fuente de luz, el CCD y el objeto representan los tres vértices de un triángulo, esto se aprecia en la figura 2.7. A diferencia de la técnica anterior, esta no puede trabajar a largas distancias [21]. Nótese que en los escaners mediante luz estructurada también se puede manejar el concepto de triangulación, pero se ha popularizado más en la implementación láser. Es frecuente que ante el uso de la triangulación se utilice una superficie giratoria [18] (en inglés *turntable*), donde se coloca el objeto a escanear y el dispositivo CCD toma imágenes durante los 360° y procesa la información a partir de los perfiles proyectados para finalmente obtener la geometría completa del objeto.

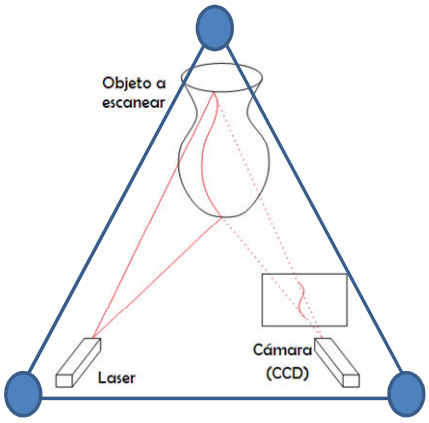


Figura 2.7 Esquema de un escáner láser por triangulación.

Los ***radares*** y ***sonares*** son sensores que *emiten* una *onda* corta *electromagnética* ó *acústica* respectivamente, o bien un *pulso*, y son capaces de detectar la onda (eco) que rebota en las superficies (figura 2.8). Así, la distancia puede obtenerse como una función del tiempo. Por el esquema que se sigue, también es un escáner tipo time-of-flight y produce range images, inclusive también se puede lanzar un pulso de luz [19].

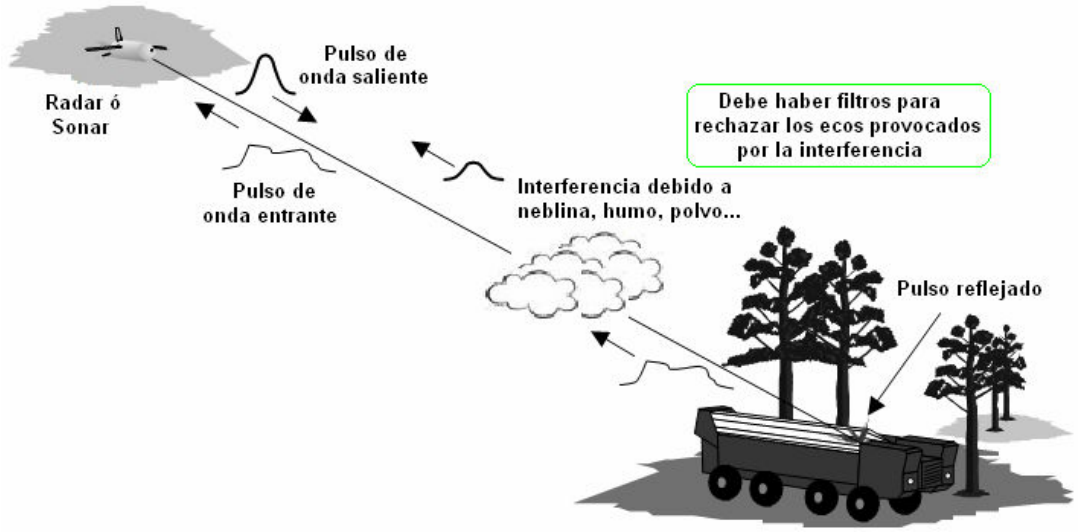


Figura 2.8 Esquema de un radar o sonar.

Existe otro método que permite obtener la profundidad de las imágenes y es mediante la proyección de un ***patrón*** de ***interferencia*** tipo ***Moiré*** (figura 2.9). Este consiste en intercalar dos o más rejillas de líneas (o del mismo tipo) con ángu-

los diferentes. Ahora bien, el range sensor proyecta sobre la superficie este tipo patrón y en función del cálculo de *diferencia de fases* obtiene la profundidad de la imagen [13].

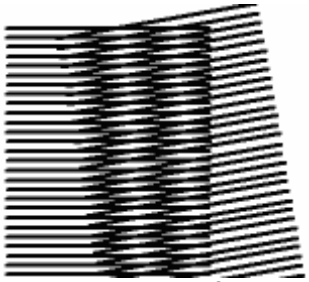


Figura 2.9 Patrón de interferencia *moiré*.

Solo por hacer mención, el ***enfoque y desenfoque*** de un lente óptico (conocido en inglés como *focusing/defocusing*) también permite obtener la profundidad de una escena. Básicamente consiste en la variación constante del foco mediante la implementación de motores a los lentes ópticos y un sensor para medir los valores de foco y así lograr que este procedimiento se realice de forma automática. Se necesitan de dos o más imágenes de la misma escena y cada una de ellas con un valor de foco distinto y a partir de la imagen mejor enfocada se realizan los cálculos para obtener la distancia.

2.2.2 Pasivos

Estereovisión es la técnica predominante en los ***escaners 3D pasivos***, pero también se puede hablar de los escaners a partir de ***contornos***.

La visión estéreo está dada a partir del sistema de visión humano, que nos permite obtener información de profundidad mediante la fusión de dos escenas monoculares, es decir, la escena que cada uno de nuestros ojos capta. Así pues, la estereovisión es la capacidad para inferir una estructura 3D y la distancia de una escena a partir de dos o más imágenes tomadas desde diferentes puntos de vista [11, 13, 20], ver figura 2.10. Computacionalmente hablando, el sistema estéreo se enfrenta a dos problemas, la *correspondencia de puntos* y la *reconstrucción 3D*. En el primer caso, se trata de localizar en las imágenes un mismo punto y también decidir cuando un punto solo es visible en solo una o unas imágenes.

Con la visión estéreo se da lugar a la *geometría epipolar* [11, 13, 15, 18] y se hace uso del concepto de triangulación.

Cabe mencionar que la implementación de esta técnica no suele ser fácil, por lo que en varios proyectos de investigación se ha mezclado la luz estructurada con la estereovisión para facilitar la correspondencia de puntos.

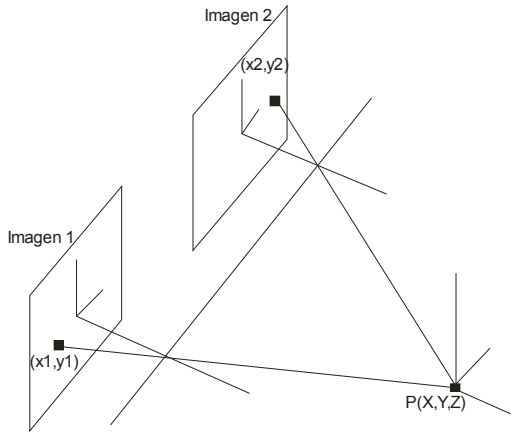


Figura 2.10 Esquema de un sistema de Estereovisión
a partir de dos puntos de vista.

Como referencia basta decir que también se puede lograr la reconstrucción 3D, aunque con ciertos inconvenientes, a partir de los ***contornos*** del objeto, aquí es importante contar con un fondo contrastante al objeto.

2.3 Aplicaciones de la digitalización 3D

La reconstrucción 3D junto con los gráficos 3D han impulsado la creación de grandes archivos con modelos 3D aplicables en diversos contextos, algunos de ellos son los siguientes:

- **Juegos y entretenimiento**: los modelos 3D han mejorado el realismo de las aplicaciones. Y también se ha hecho reutilización y adaptación de modelos similares para la reducción de costos.

- **CAD**: socorrido en por los Técnicos e Ingenieros de compañías manufactureras.

- **Patrimonio cultural**: el principal objetivo es la preservación digital de piezas y sitios históricos.

- **Medicina**: para la detección de deformaciones de órganos y poder ser usadas como herramientas de diagnóstico.

- **Reconocimiento facial 3D**: implementado principalmente en asuntos policíacos.

- **Bioinformática**: una de sus aplicaciones es la comparación estructural de diferentes proteínas que juegan un rol importante en la vida de los humanos.

- **Información geográfica**: su primordial objetivo es el modelado de las diferentes superficies, p/e para las aplicaciones en cartografía o para una planeación espacial.

Capítulo 3

3 RECONSTRUCCIÓN BASADA EN LUZ

En este capítulo se detallan dos técnicas de reconstrucción específicas, a partir de luz láser y códigos binarios. En ambos métodos se emplea el concepto de triangulación y de forma explicita en la primera técnica ya que es importante para su implementación, mientras que en la segunda, lo que más atención requiere es el tipo de código a proyectar y la calibración de la cámara.

La ventaja principal de utilizar luz estructurada son sus características visuales porque son fáciles de distinguir para una cámara (CCD). La *adquisición de la forma* del objeto es el principal interés.

3.1 Luz Láser

El haz láser puede ser representado por diferentes formas, como lo muestra la figura 3.1. Las formas más utilizadas son el punto, la línea y la cruz, no solamente se emplean en la reconstrucción, sino también en cuestiones de seguridad, de visión industrial, como marcado de línea de corte, entre otros. Existen empresas que fabrican exclusivamente este tipo de aparatos, p/e StockerYale [25]. La digitalización 3D láser no se restringe a un solo tipo de haz, se puede utilizar el que mejor se considere o de resultados satisfactorios.

Figura 3.1 Algunos tipos de proyecciones mediante láser.

La geometría básica para un sistema en triangulación con luz láser la describe la figura 3.2. La fuente de luz se coloca a una distancia b (conocida como línea

base) desde el centro de la proyección de la cámara. Es vital que el eje Z y el eje óptico de la cámara coincidan. Se debe tener presente que en la figura 3.2 los ejes y, Y son perpendiculares a la hoja. Los ejes x, X y y, Y son respectivamente paralelos, nótese que existen dos sistemas coordenados, a saber, XYZ y xyz. La longitud focal de la cámara está dada por f. El haz que emite el proyector (plano de luz) es perpendicular al plano XZ y forma un ángulo controlado, *θ*, con el plano XY. Así pues, enfatizando, el plano de luz y el eje Y son paralelos y ambos son perpendiculares a la hoja. La intersección del plano de luz con la superficie del objeto forma una curva plana llamada *stripe*, que bien se podría traducir como *perfil* o *raya*, mismo que es observado por la cámara. De esta manera, con las características dadas, la coordenada de un punto del stripe $\mathrm{P} = \left[X, Y, Z\right]^{\mathrm{T}}$ está dado por:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \frac{b}{f \cot\theta - x} \begin{pmatrix} x \\ y \\ f \end{pmatrix} \qquad (1)$$

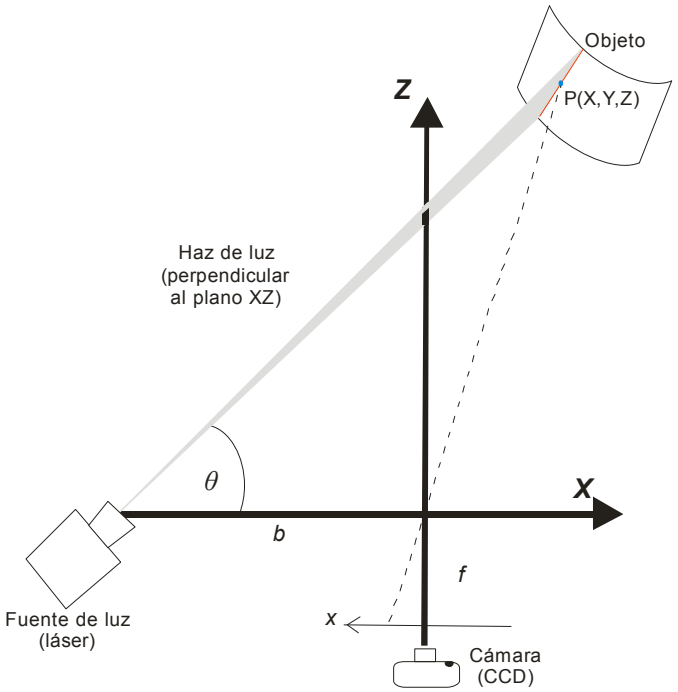


Figura 3.2 Geometría básica de un escáner láser 3D por triangulación. Los ejes *y*, *Y* son perpendiculares a la hoja.

Si se adopta este esquema particularmente, el inconveniente que se presenta es que no se podrá escanear al objeto en su totalidad ya que permanece estático. Nótese, a pesar de variar el ángulo θ el haz de luz no alcanzará a cubrir la parte posterior del objeto. Una solución sería mover la cámara y la fuente atrás del objeto, para así posiblemente escanearlo totalmente; o bien, algo más sencillo, tomar al objeto y girarlo un cierto número de veces a determinado ángulo; sin embargo surge otro problema en ambos casos, habrá que realizar las operaciones adecuadas (p/e rotación) digamos a partir de la segunda ronda de adquisición de puntos para que la forma del objeto se preserve y se mantenga en un mismo sistema coordenado, esto equivaldría a la alineación y registro de los range images citados en el capítulo anterior.

Ante el inconveniente citado, aparece el concepto de *superficie giratoria* (referido en la literatura como *turntable*), que consiste en girar el objeto 360° mientras es grabado (analizado) con el haz de luz láser proyectado, el principio de triangulación se conserva (figura 3.3), sin embargo, la manera de obtener las coordenadas tridimensionales cambia, así como las consideraciones previas al escaneo.

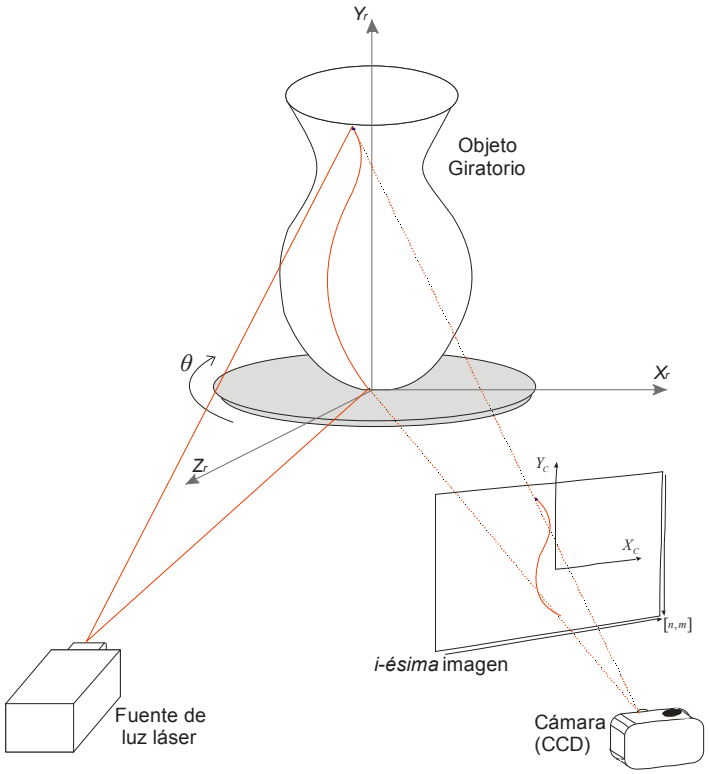


Figura 3.3 Escáner láser 3D por triangulación con objeto giratorio.

Primeramente, se debe seguir un proceso de ***alineación*** que consiste en centrar el haz de luz láser con el eje de rotación Y_r de tal manera que coincidan, el objeto también se debe colocar en el centro de la plataforma y de igual manera la cámara se debe ajustar tal que en el centro de la imagen coincida con el eje Y_r, es decir que Y_C se paralela y coincida con Y_r. Si esto no se lleva a cabo, el modelo resultante será una especie de volumen elíptica deforme.

En los range images p/e se necesita de múltiples puntos de vista para la reconstrucción, sin embargo en este caso lo que se requiere es, varias imágenes tomadas por el CCD mientras el objeto gira (360°) con el haz proyectado. Generalmente se graba un video y posteriormente se descompone en cuadros (*frames*) para su análisis, la figura 3.4 lo muestra.

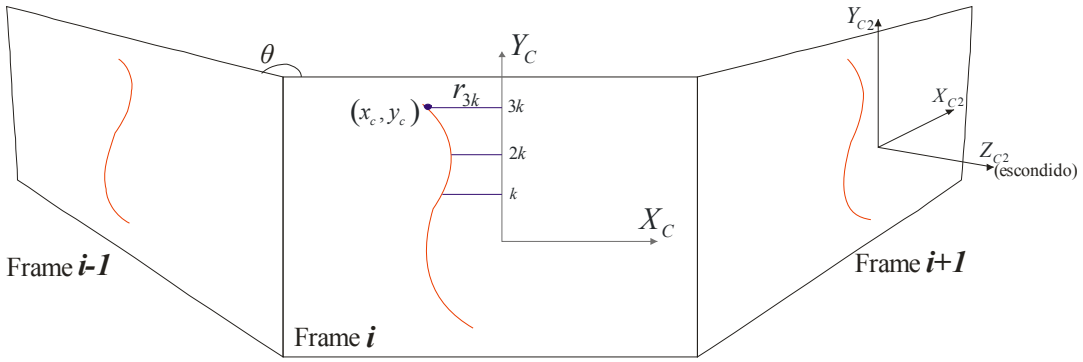


Figura 3.4 Escáner láser 3D por triangulación con objeto giratorio.

Así pues, para cada frame se busca el borde o stripe (en caso de utilizar un haz tipo línea) que genera la fuente de luz, nótese que en la figura 3.4 a lo largo del eje Y_C con determinado paso k, existe una distancia $r = |x_c|$ desde dicho eje hasta un punto del stripe (x_c, y_c). Ahora bien, el sistema coordenado de la imagen será el sistema de referencia para obtener los puntos 3D. Con los parámetros dados hasta el momento es posible obtener la profundidad del punto. Primero, no olvidar que la coordenada (x_c, y_c) es conocida y que el eje $Y_C = Y_{C2}$ (figura 3.4 frame $i+1$), por lo tanto y_c es la misma en el mundo tridimensional (y_{c2}), solo basta encontrar x_{c2} y z_{c2}. Imagínese que se rota el sistema C 90° sobre el eje X_C, es decir, ahora la hoja está representada por el plano $X_{C2}Z_{C2}$ y el eje Y_C es

perpendicular a la hoja, como lo muestra la figura 3.5. Finalmente con senos y cosenos se obtienen las coordenadas faltantes, a saber:

$$\theta = \frac{2\pi}{n^{\circ}\, frames}, \qquad \theta_i = \theta_{i-1} + \theta \qquad (2)$$

$$\begin{bmatrix} x_{c2} \\ y_{c2} \\ z_{c2} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) * r \\ y_c \\ sen(\theta_i) * r \end{bmatrix} \qquad (3)$$

En la fórmula anterior (3) no olvidar que $r = r_{3k}$, para concordancia de las ilustraciones.

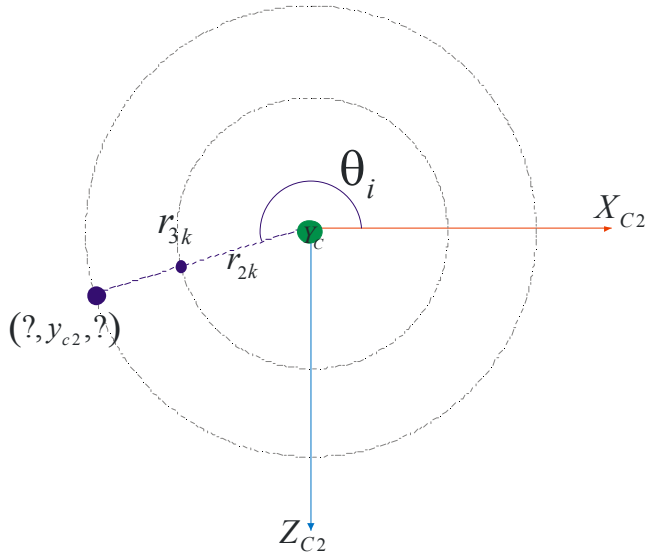


Figura 3.5 Vista superior del frame i.

El proceso anterior se realiza para todos o algunos puntos (dependiendo del paso k) del stripe, y para cada frame. Hecho esto, el resultado es una nube de puntos que representan la figura del objeto.

3.2 Códigos Binarios

Hacia 1982, Posdamer y Altschuler propusieron una codificación binaria que consistía en un patrón de rayas (*striped pattern*). En esta técnica solo existen dos niveles de iluminación que los puntos $\left(x_{p1}, y_{p1}\right)$ del patrón proyectado pueden tomar, codificados en 0 y 1, es decir, ausencia o presencia de luz sobre el objeto [15, 22]. Así pues, un *código* es la secuencia de 0's y 1's. Como la fuente de luz generalmente es un proyector y la calidad del mismo está determinada por su resolución (en pixeles), análogamente un patrón de luz de códigos binarios se puede ver como una matriz de puntos con haz de luz binaria, como lo muestra la figura 3.6; y se tiene control sobre el patrón de cada columna

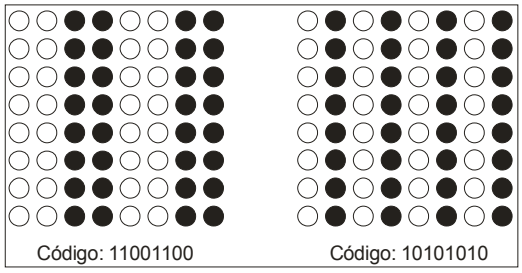


Figura 3.6 Códigos binarios en su representación matricial.

Generalmente el patrón de rayas (*stripes*) a proyectar tiene una secuencia lógica, a saber, el número de rayas es factor de dos, es decir, debe haber 2, 4, 8, 16… stripes alternados, algunos autores lo catalogan por el bit más significativo o menos significativo, ver figura 3.7.

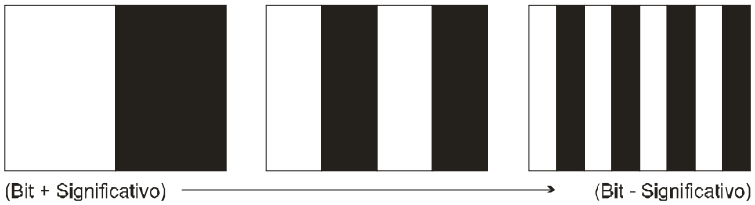


Figura 3.7 Proyección de un patrón de códigos binarios.

Una vez elegido el código binario en base a ciertos parámetros, el siguiente paso es, procesar las imágenes obtenidas (CCD) de la escena con el patrón de luz proyectado. En otras palabras, se debe analizar la imagen en busca de los

bordes de los stripes. Es importante tener en cuenta que las rayas sobre la escena *no* serán verticales, a menos que la superficie del objeto sea totalmente lisa, plana y paralela al plano de proyección, es decir, los stripes sobre el objeto estarán deformados y presentarán ruido debido a los problemas de reflexión según la superficie, como lo muestra la figura 3.8 [23]. Ante el problema de ruido de reflexión de luz, se deben implementar filtros capaces de minimizar este inconveniente.

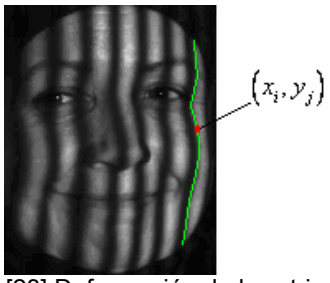


Figura 3.8 [23] Deformación de los stripes sobre el objeto.

Una vez localizados los bordes, para cada $(x_i, y_j), 1 \le i \le n, 1 \le j \le m$ de la imagen $a[n,m]$, pertenecientes a un punto de un borde (de algún stripe), a través del concepto de triangulación y *previa* ***calibración de la cámara***, se puede obtener la profundidad de ese punto y por ende reconstruir tridimensionalmente el objeto. Cabe señalar que el objeto debe permanecer estático y la reconstrucción será sobre el campo visible de la imagen.

La *calibración de una cámara* consiste en el cálculo de parámetros intrínsecos y extrínsecos, logrando así la relación de medición *pixel-unidad _ métrica*, donde la unidad métrica pueden ser milímetros, centímetros o alguna otra unidad.

La calibración no es objeto de este trabajo, sin embargo, a continuación se describe vanamente un procedimiento matemático para la obtención de los puntos tridimensionales del objeto. Para mayor referencia al respecto véase [13,22,24].

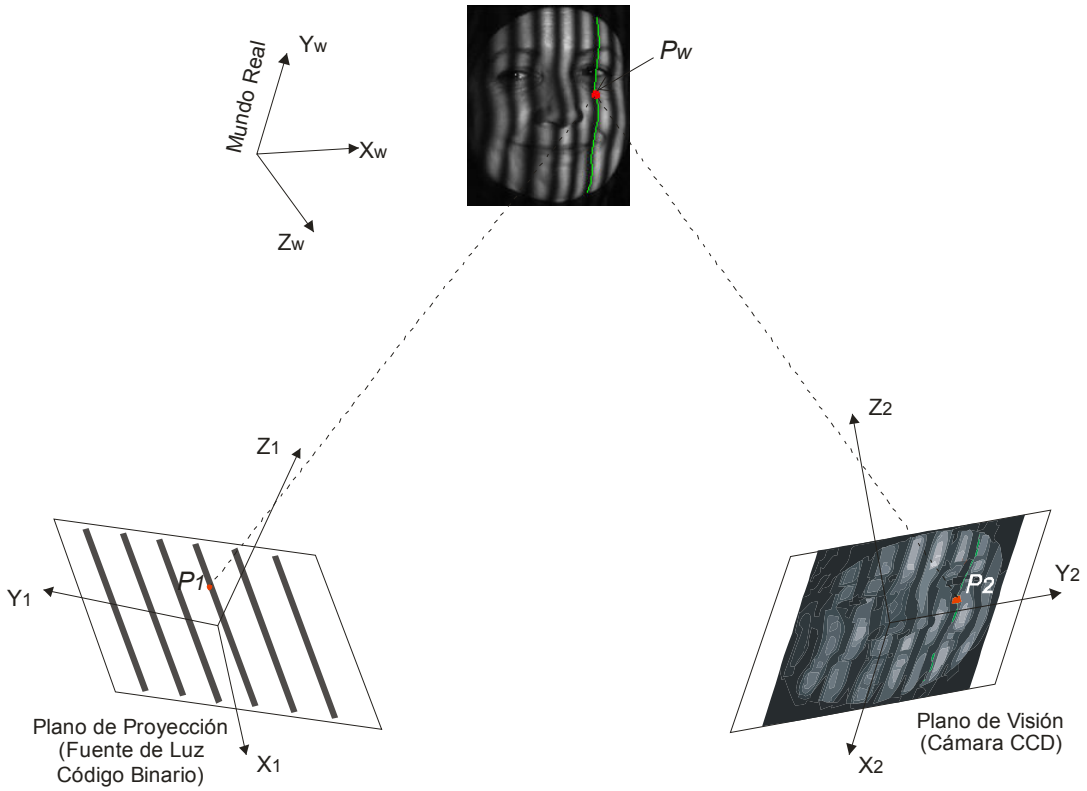


Figura 3.9 Principio de triangulación con luz estructurada (códigos binarios) y escena estática.

En base al esquema de la figura 3.9, dadas las *matrices de transformación* a partir de la calibración de la cámara que relaciona un punto P_w del objeto con su proyección (P_2) en la imagen capturada y en la imagen del proyector (P_1), las coordenadas de un punto del objeto (del mundo real) pueden ser determinadas por:

$$N = \left(\mathrm{P}^t\mathrm{P}\right)^{-1}\mathrm{P}^t F \qquad (4)$$

Donde:

$$N = \begin{bmatrix} x_{pw} \\ y_{px} \\ z_{pw} \end{bmatrix}, \qquad F = \begin{bmatrix} A_{134}x_{p1} - A_{114} \\ A_{134}y_{p1} - A_{124} \\ A_{234}x_{p2} - A_{214} \\ A_{234}y_{p2} - A_{224} \end{bmatrix} \qquad (5)$$

$$\mathrm{P} = \begin{bmatrix} A_{111} - A_{131}x_{p1} & A_{112} - A_{132}x_{p1} & A_{113} - A_{133}x_{p1} \\ A_{121} - A_{131}y_{p1} & A_{122} - A_{132}y_{p1} & A_{123} - A_{133}y_{p1} \\ A_{211} - A_{231}x_{p2} & A_{212} - A_{232}x_{p2} & A_{213} - A_{233}x_{p2} \\ A_{221} - A_{231}y_{p2} & A_{222} - A_{232}y_{p2} & A_{223} - A_{233}y_{p2} \end{bmatrix} \quad (6)$$

Se recalca que estas tres matrices son resultado de la simplificación y asociación de términos comunes de las matrices de calibración, referencia [22].

Capítulo 4

4 PLANTEAMIENTO Y ANÁLISIS DEL PROBLEMA

Este capítulo tiene por objetivo describir cual es la problemática que inspira el trabajo de tesis y delimitar mediante el análisis la solución a dicho problema.

4.1 Introducción

No muchos países cuentan con la riqueza histórica cultural como la que posee México. El origen de este proyecto fue la difusión y conservación de la cultura prehispánica mexicana y la falta de disponibilidad monetaria para adquirir un costoso digitalizador 3D comercial.

Mediante la reconstrucción tridimensional de piezas arqueológicas de manera digital, es más fácil y accesible poder difundir la riqueza histórica mexicana no solo a nivel nacional sino a todo el mundo, y desde luego, se logra la preservación de estos modelos a lo largo del tiempo.

4.2 Objetivos

El objetivo fundamental de esta tesis es la implementación de un Escáner 3D (sistema) tanto la parte del hardware como del software en tiempo real, buscando en todo momento economizar la construcción del escáner sin pérdida de la relación costo-beneficio.

Este sistema permitirá la reconstrucción tridimensional de piezas arqueológicas relativamente pequeñas y los modelos obtenidos tendrán un aspecto suavizado.

4.3 Especificaciones del Sistema

Con el marco teórico expuesto en los capítulos anteriores, se ha optado por implementar un escáner 3D mediante triangulación y luz estructurada, específicamente el patrón de luz a utilizar es el láser con un haz tipo línea, también se manejará el concepto de superficie giratoria. Con esto, se obtendrá una nube de puntos tridimensionales que representarán la forma del objeto, mismos que serán utilizados para modelar al objeto con parches poligonales y a través de la interfaz se podrá navegar en el espacio para visualizar el modelo digital.

4.3.1 Justificación de la Técnica a Implementar

Utilizar un dispositivo con un range sensor que nos proporcione la profundidad de los pixeles definitivamente está fuera del objetivo por su alto costo. La versatilidad que ofrece la luz estructurada orilló a su utilización, sobre todo a luz láser (como se verá más adelante), ya que no necesita de un costoso proyector que proporcione los patrones de luz. Y finalmente, la superficie giratoria es muy buen elemento para obtener casi la totalidad de la forma del objeto.

4.3.2 Descripción General

El sistema final (hardware y software) será capaz de escanear un objeto del mundo real que cumpla con ciertos parámetros para poder ser almacenado y tratado digitalmente, específicamente se obtendrá un modelo 3D en tiempo real. El sistema ofrecerá al usuario la posibilidad de realizar ciertas configuraciones necesarias mediante una interfaz gráfica, que afectarán directamente la adquisición de datos, una de esas configuraciones es la elección del dispositivo de video (CCD) así como sus propiedades de iluminación. El sistema también dispondrá de una interfaz que permita visualizar el modelo tridimensional.

4.3.3 Alcance y Restricciones

Dadas las características anteriores del sistema a implementar se pueden acotar las particularidades de los objetos a digitalizar y señalar las limitantes del sistema, a continuación se describen:

- **Tamaño y peso objeto**: la pieza a escáner no debe tener una altura mayor a 30 cm y su peso no debe sobrepasar los 10 kg.

- **Forma del objeto**: para obtener mejores resultados en el modelo 3D, el objeto debe tener una forma un tanto cilíndricas-volumétricas, no plana como p/e un celular, que debido a su forma existiría un alto número de oclusiones de partes. Los objetos tampoco deben presentar hoyos, o al menos no tan profundos, p/e si se digitalizará el rostro (figura 2.4, cap.2) con la boca abierta (que representaría el hoyo) sería imposible reconstruirla mediante el método que se implementará.

- **Límites del digitalizador 3D**: debido a la naturaleza del escáner no será posible reconstruir la parte superior e inferior del objeto.

- **Material del objeto**: el tipo de material de la pieza también es importante porque si la superficie es muy brillante o muy opaca entonces existirá índices altos de reflexión que provoque ruido o el material podrá absorber gran cantidad del haz de luz proyectado.

4.3.4 Requerimientos Funcionales

Los requerimientos funcionales más importantes del sistema serán:

- La facilidad al usuario de poder elegir el dispositivo de entrada de adquisición de imágenes (CCD), así como modificación de sus propiedades de iluminación.

- El usuario tendrá la posibilidad de acotar la región de búsqueda del stripe para alcanzar el análisis en tiempo real.

- El usuario podrá modificar el paso de la imagen raster para la búsqueda de los puntos de los stripes, así como también el umbral de detección del haz de luz.

- Se tendrá la opción de guardar el fichero con los puntos tridimensionales.

- La interfaz permitirá la visualización y navegación 3D de cualquier modelo previamente escaneado y almacenado.

4.3.5 Requerimientos no Funcionales

Los principales requerimientos no funcionales corresponden a la alineación del escáner (a nivel hardware):

- Es vital importancia centrar el haz de luz láser con el eje de rotación, representada por el centro de la plataforma giratoria.

- Se debe hacer coincidir el centro de la imagen de la cámara con el eje de rotación.

- Finalmente en cuanto alineación se refiere, el objeto debe colocarse en el centro de la plataforma.

- La iluminación ambiental juega un papel importante en la detección del haz de luz láser.

- La intensidad de luz del haz debe ser constante durante la digitalización.

4.3.6 Suposiciones

Se espera que el haz de luz láser de clase II [25], que cuenta con una longitud de onda de 650-20 nm, sea suficiente para ser detectado por la cámara. Del mismo modo, se espera hallar alguna manera de obtener un haz de luz láser en forma de línea, ya que los lasers fabricados con esta característica son costos. Un haz en forma de línea facilitaría la digitalización comparado con un haz en forma de punto.

Capítulo 5

5 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

En este apartado se modelará e implementará el sistema de digitalización 3D, es importante recordar que se realizará a nivel hardware y software.

5.1 Diseño

El primer punto que se debe tener en claro es la identificación de los componentes del sistema correspondientes al hardware y software, en la figura 5.1 se describen estos componentes que conciernen a su arquitectura desde un punto de vista abstracto.

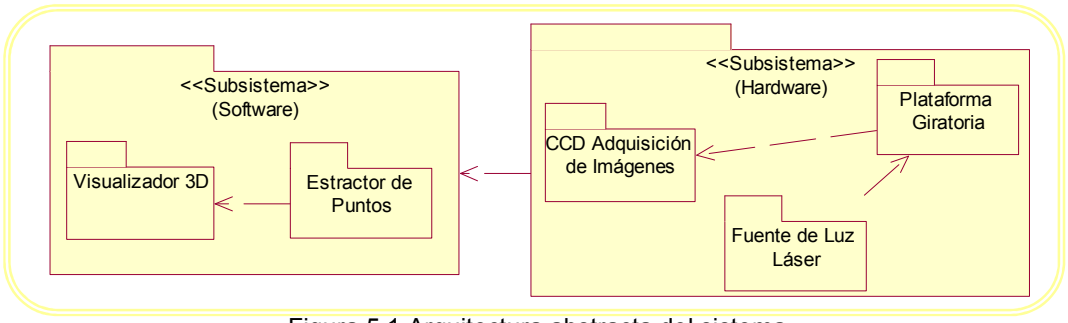


Figura 5.1 Arquitectura abstracta del sistema.

A continuación se detalla cada módulo del sistema para dar paso a los diagramas de clase e interacción (subsistema *software*).

5.1.1 Hardware

La parte de los componentes físicos es muy importante, ya que de ellos se verá reflejado el costo del escáner. Así pues, la *fuente de luz láser* será proporcionada por un apuntador láser común, el suministro de energía deben ser baterías recargables o bien una fuente de voltaje constante para evitar el consumo desmedido

de baterías. Con este tipo de apuntadores láser siempre se produce un haz de luz en forma de punto, pero se necesita obtener una línea. Este inconveniente se puede resolver con ayuda de la óptica a través del concepto de *refracción* de ondas que consisten en la dispersión-curvado de ondas ante un obstáculo [26,27,28]; no se debe olvidar que el haz de luz láser (visible) tiene una longitud de onda comprendida entre los 400-780 nanómetros. De esta manera, un lente cilíndrico es una buena solución para obtener un haz tipo línea (ver figura 5.2). Se debe elaborar un soporte donde se coloque el apuntador y el lente. Es importante recalcar que la línea resultante debe alinearse con el eje de rotación.

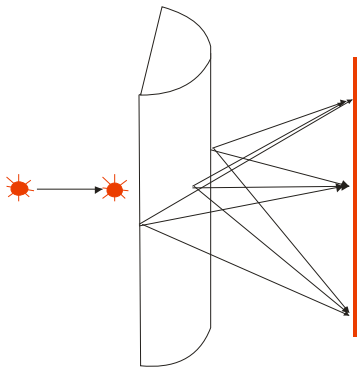


Figura 5.2 Lente óptico cilíndrico que realiza la refracción de luz y genera una haz tipo línea.

En lo que respecta a la *plataforma giratoria*, se hará uso de un motor de microondas con alimentación de 127 volts para evitar incluir algún convertidor de corriente y se colocará una superficie circular sobre su eje (es importante que el eje del motor se coloque en el centro de esta superficie), sitio donde se colocarán las piezas a escanear. Este motor debe ser montado sobre una base de tal manera que el eje no aplique todo el torque al momento de girar con el objeto encima, porque de lo contrario podría quebrarse, esta parte debe verificarse con las especificaciones técnicas del motor. Nótese que el eje del motor es el eje de rotación del sistema.

Finalmente, la cámara (sensor CCD) debe conectarse a la PC y fungir como un dispositivo de entrada para que a través de la aplicación que se desarrollará se puedan obtener las coordenadas de los stripes en tiempo real. Es importante que este dispositivo proporcione un video con un tamaño de al menos $640x480$ pixe-

les con el fin de obtener buenos resultados. No se debe olvidar que el centro del video se tiene que alinear con el eje de rotación.

5.1.2 Software

En este apartado se deben contemplar más detalles y conviene ser cuidadoso, no es tan simple el montaje como se observa en la figura 5.1. A continuación se detalla el subsistema que corresponde al software, en la figura 5.3 se encuentra su diagrama de estados. Como punto inicial se supone el flujo de datos provenientes de la cámara, al final del capítulo se mostrará un diagrama con ambos subsistemas.

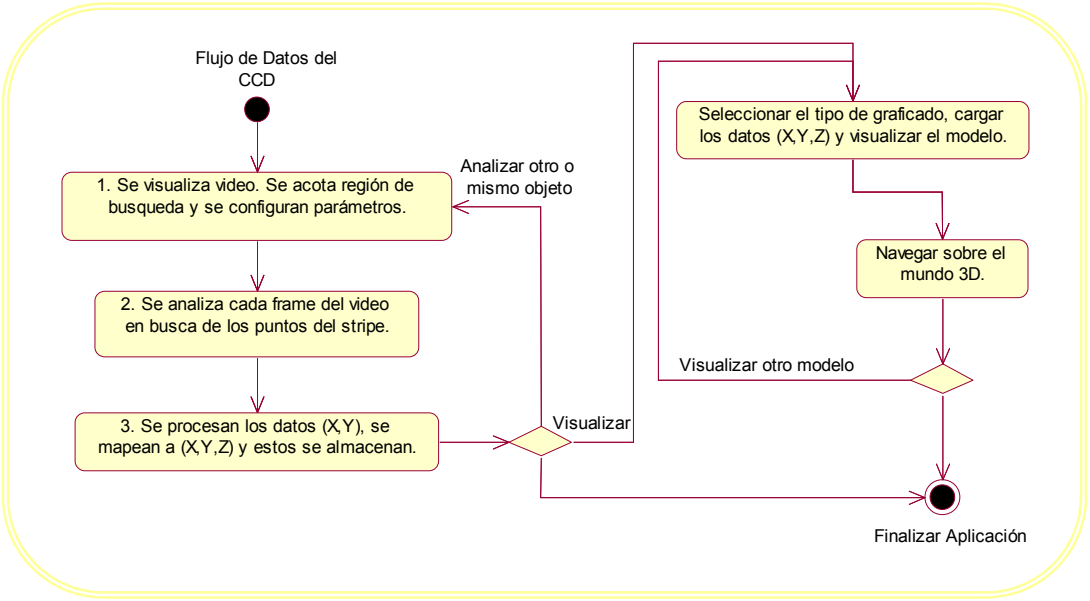


Figura 5.3 Diagrama de estados.

Antes de iniciar con el diagrama de clases, es de vital importancia dar solución a inconvenientes que se puedan presentar y de no tratarlos los resultados se verán reflejados en el modelo 3D. De hecho, hay uno posible, pero muy determinante en el nivel de detalle del objeto: *las oclusiones*. En la figura 2.4 del capítulo 2 se aprecia este fenómeno. Esto se da en los estados 2 y 3 de acuerdo al diagrama de la figura 5.3.

Así pues, una estructura de datos en donde se pueden almacenar las coordenadas (x, y) es suficiente; es decir, todos los puntos de cada uno de los stripes

(cada stripe pertenece a un frame) es un arreglo bidimensional de puntos 2D, digamos $a[m,n]$ donde m es el número de frames/stripes y n es el número de puntos de cada stripe, n es el mismo para cada stripe. Debido a las oclusiones durante el escaneo se puede llegar a tener una matriz como la siguiente:

$$a = \begin{bmatrix} - & \mathrm{P}_{2,1} & - & \mathrm{P}_{i-1,k-3} & \mathrm{P}_{i,k-3} & \mathrm{P}_{i+1,k-3} & \mathrm{P}_{7,1} & - & \mathrm{P}_{9,1} \\ - & \mathrm{P}_{2,2} & \mathrm{P}_{3,2} & - & - & - & \mathrm{P}_{7,2} & \mathrm{P}_{8,2} & \mathrm{P}_{9,2} \\ \mathrm{P}_{1,3} & \mathrm{P}_{2,3} & \mathrm{P}_{3,3} & - & - & - & \mathrm{P}_{7,3} & \mathrm{P}_{8,3} & - \\ - & - & \mathrm{P}_{3,4} & - & \mathrm{P}_{i,k} & - & \mathrm{P}_{7,4} & \mathrm{P}_{8,4} & - \\ \mathrm{P}_{1,5} & \mathrm{P}_{2,5} & \mathrm{P}_{3,5} & \mathrm{P}_{i-1,k+1} & - & \mathrm{P}_{i+1,k+1} & - & - & \mathrm{P}_{9,5} \\ \mathrm{P}_{1,6} & - & \mathrm{P}_{3,6} & - & \mathrm{P}_{i,k+2} & - & \mathrm{P}_{7,6} & - & \mathrm{P}_{9,6} \end{bmatrix} \quad (1)$$

Donde $\mathrm{P}_{i,k} = \left(x_{i,k}, y_{i,k}\right)$ es un punto del $i-ésimo$ stripe y $k-ésima$ posición. Como se dijo, los huecos corresponden a pérdida de información y detalle en la forma del objeto en consecuencia. Después de un análisis, se ha optado que es posible llenar los huecos con el promedio de sus vecinos, un criterio puede ser con los vecinos que no son huecos y se encuentran arriba, abajo, a la izquierda y a la derecha, o bien en diagonal. De esta forma la matriz (1) quedará llena.

Independientemente del lenguaje de programación que se utilice, será necesario valerse de bibliotecas gráficas que implementen por ejemplo OpenGL para el manejo eficiente de los gráficos, y una biblioteca más, que permita la adquisición y manipulación de video a través de un dispositivo de entrada (cámara). Es por esta razón que en el siguiente diagrama de clases solo se muestran las clases que se implementarán en este trabajo de tesis, el resto de procesos serán manejados por las bibliotecas.

En la figura 5.4 se aprecia el diagrama de clases. Éstas son suficientes para la implementación del sistema de digitalización 3D. En ambas clases se da por hecho la existencia de procesos que ayudarán a la implementación, pertenecientes a bibliotecas, en su momento, definido el lenguaje se ahondará en ellas, por ejemplo objetos básicos de gráficos 3D se encuentran en la clase *Visualizador3D* los objetos *Canvas3D, Lights3D* y *Mesh3D* son necesarios para llevar a cabo la implementación de ésta; la biblioteca no manejará en general los mismos nombres y tal vez dependan de más objetos, pero esto se verá en la implementación.

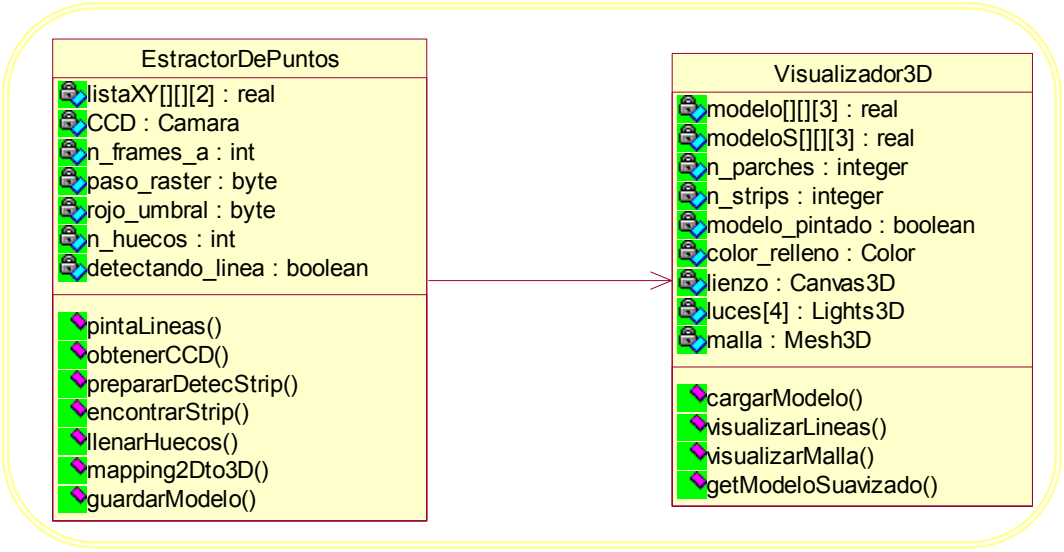


Figura 5.4 Diagrama de clases.

Ahora se detallará cada clase, así como sus propiedades y métodos. En *EstractorDePuntos* el principal objetivo es: obtener los puntos (x, y) de cada unos de los stripes pertenecientes a los frames, estos serán proporcionados por el video que ofrece la cámara (objeto *Camara* llamado *CCD*). Estos puntos se almacenarán en *lista[][][2]*, esta matriz será dinámica porque el número de stripes y de sus puntos es dependiente de la figura que se escanee, el tercer índice representa al valor de x o al valor de y. Es importante tener la cuenta del número de frames analizados (n_frames_a) ya que se utilizará en la fórmula (2) del cap.3, para obtener el ángulo θ de cada frame. Con *n_huecos* se podrá saber si existen oclusiones, entonces se aplicará el algoritmo para eliminarlos.

En lo que concierne a los métodos de la clase *EstractorDePuntos*, está *obtenerCCD()* que es quien proporcionará la comunicación de la aplicación con el dispositivo de entrada de video. Los demás métodos se describen a continuación:

- *pintaLineas()*: este método tendrá que ver con un proceso interactivo con el usuario, para que él pueda acotar la región de búsqueda del stripe, de esta manera se acelerará el proceso de detección, logrando que se obtengan los datos en tiempo real.

- *prepararDetecStrip()*: este método se encargará de preparar las variables que influyan en la detección del stripe, algunas configuradas por el usuario, otras, necesarias por la implementación misma; como son: *paso_raster* y *rojo_umbral*. En la primera el usuario debe de proporcionar el tamaño del paso, es decir, como la imagen se irá escaneando fila por fila para determinado y, en ocasiones será tedioso, innecesario y lento porque tendrá un costo computacional el detectar el stripe con un paso unitario, (ver figura 5.5). Es importante recordar como se dijo al inicio del capítulo 2 que un pixel de la imagen tiene tres componentes, rojo, verde y azul (*RGB*), cada uno de ellos representados por un *byte* (*8 bits*), de esta manera cada componente puede tomar un valor entre *0-255*. Así pues, con el haz de luz láser proyectado se alcanza un rojo saturado, pero no siempre a *255*, por lo que entra en juego la variable *rojo_umbral* misma que el usuario puede determinar.

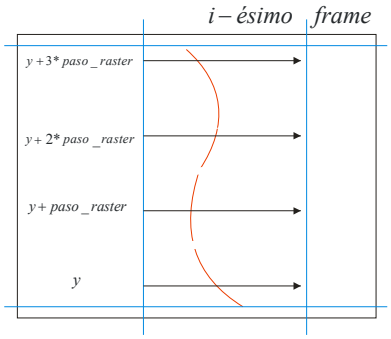


Figura 5.5 Detección de un stripe.

- *encontrarStrip()*: este método se encargará propiamente de la detección del stripe para cada frame del video como lo muestra la figura 5.5, e irá almacenando las coordenadas en *lista[][][2]*. Se enfatiza, que es en este momento cuando se sabe si hay o no oclusión, si la hay, almacena en el arreglo un identificador, mismo que servirá para llenar los huecos mediante el algoritmo descrito párrafos arriba.

- *llenarHuecos()*: con la variable *n_huecos* se decidirá si se aplica o no el algoritmo.

- *mapping2Dto3D:* una vez que se garantice la ausencia de huecos, se aplican las fórmulas (2) y (3) del capítulo 3 para hacer la transformación de espacios: del bidimensional (XY) al tridimensional (XYZ).

- *guardarModelo():* finalmente se guardan los puntos (x, y, z) en un archivo para que puede ser cargado por el visualizador.

En la clase *Visualizador3D* la matriz *modelo[][][3]* almacenará (mediante *cargarModelo()*) los puntos (x, y, z) previamente generados por la clase anterior. El valor de *n_stripes* será conocido al momento de cargar en memoria los puntos 3D y será el tamaño del primer índice de la matriz *modelo.* La variable *n_parches* será meramente informativa. El método *visualizarLineas()* se encargará de graficar cada uno de los stripes como líneas. Contrario a *visualizarMalla()*, donde con los puntos tridimensionales de la matriz *modelo* se formarán parches triangulares y con ello se obtendrá un modelo poligonal (mesh). Mismo que podrá ser sometido a suavizado mediante el método *getModeloSuavizado()*, que implementará curvas de Bezier (capítulo 1) o splines, o bien, si existe, utilizar alguna función de la biblioteca gráfica. Sin entrar a detalles del algoritmo para el suavizado, lo que se obtendrá será un número mayor de puntos para cada stripe, y es aquí donde se utilizará la matriz *modeloS[][][3]*, que en comparación a la otra matriz, crecerá sustancialmente en el segundo índice, razón por la que será liberada de memoria una vez que se visualice también con parches triangulares.

Aunque parezca sencilla la implementación de la clase *Visualizador3D*, no lo es. Sobre todo se pueden encontrar dificultades en el manejo de los objetos gráficos 3D, y serán oportunamente mencionadas en la implementación. Un aspecto que se debe tener muy presente es la *iluminación* en la escena 3D, de ello dependerá el realismo del modelo. Quienes también juegan un papel importante en este aspecto, son los vectores normales de los parches, que influye en el como se recibe y se refleja la luz, ahondando en éste inconveniente; tiene mucho que ver como se forman los parches, es decir, deben seguir una sola codificación. En este caso, será en sentido de las manecillas del reloj, como lo muestra la figura 5.6.

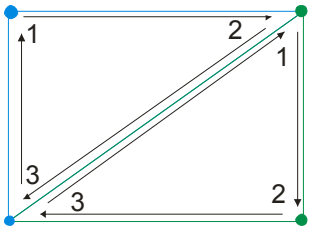


Figura 5.6 Construcción de parches triangulares en sentido de las manecillas del reloj.

Otro aspecto importante a considerar, es la liberación de memoria oportuna de los objetos 3D, de lo contrario la aplicación podría consumir muchos recursos hasta llegar a *colgar el sistema*, como coloquialmente se dice.

5.2 Implementación

En este apartado se explicará el *cómo* de las implementaciones de los subsistemas. Primero se abordará el hardware y al finalizar la parte del software se mostrará con un diagrama de bloques la lógica de todo el sistema.

5.2.1 Hardware

Efectivamente, se utilizó un motor de microondas para implementar la *plataforma giratoria*, en la figura 5.7 se muestra su diseño final. Consiste en un cilindro de madera con 10 cm de diámetro formado por tres tablas, en su interior se hizo un agujero para colocar y atornillar el motor de tal forma que su eje quedara e el centro del cilindro. El siguiente paso consistió en desfondar un bote de aluminio para que encajara y se atornillara al cilindro. La tapa superior del bote tiene una especie de riel en su circunferencia donde se colocaron cuatro balines, mismos que son los puntos de apoyo para la plataforma de madera, en ellos recae el peso del objeto ayudando a disminuir el torque que el eje del motor tiene que ejercer. Este eje se asoma por un agujero que se le hizo a la tapa del bote, en él se acopla un conector que está pegado a la superficie circular. La alimentación eléctrica del motor es de 127 volts y su velocidad de giro es 9 segundos por vuelta (360°). En la plataforma circular se trazaron dos líneas perpendiculares que se intersectan en el centro de la plataforma, marcadas cada centímetro para ayuda durante la alineación del objeto.

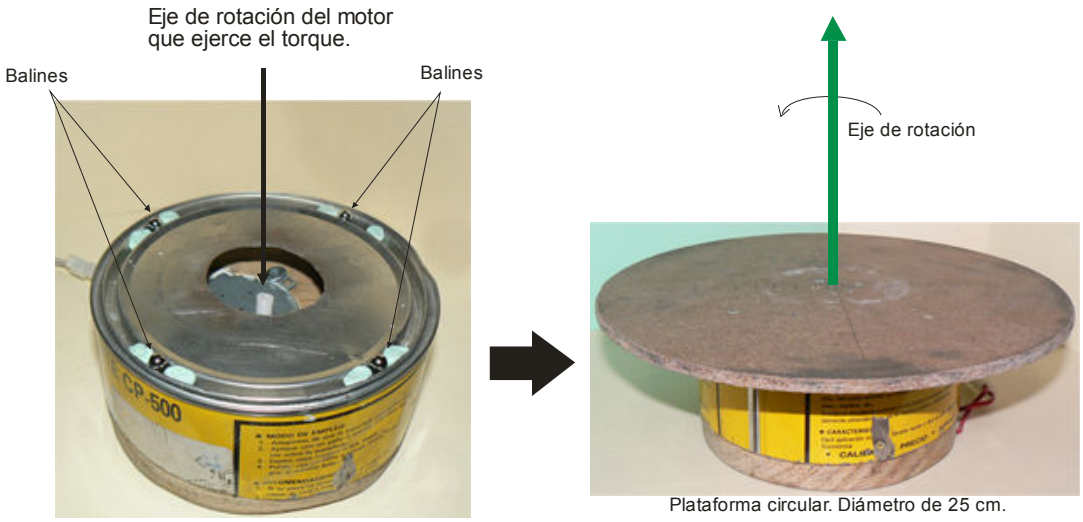


Figura 5.7 Implementación de la plataforma giratoria.

En lo que respecta a la fuente de luz láser, se hizo un soporte donde se colocó el apuntador láser y el lente cilíndrico refractor de luz como se observa en la figura 5.8.

Figura 5.8 Implementación de la fuente de luz láser
con lente refractor.

Se hizo uso de pilas AAA recargables de 300 mA (miliampers) para hacer funcionar el apuntador. El láser tiene un consumo de energía de 1mW (miliwatt), pertenece a la clase II considerado como de *ojo-seguro* [25], sin embargo no se averiguo esta última aseveración. Su longitud de onda está en el rango de 650-20 nanómetros dependiendo de la carga de las baterías. El lente cilíndrico que se

ocupó fue un fragmento de agitador de laboratorio (varilla de vidrio) con un diámetro de ¼", proporcionado por el Dr. Gustavo Zurita responsable del laboratorio de óptica de la Facultad de Ciencias Físico Matemáticas de la BUAP.

Finalmente para esta sección, se utilizó una cámara web con foco ajustable de manera manual.

5.2.1 Software

La aplicación se desarrolló bajo el ambiente Windows, específicamente, XP. La PC cuenta con un procesador Centrino a 2.0 MHz, un acelerador de gráficos Intel de 256 MB, 3 GB de memoria RAM.

El lenguaje de programación que se utilizó para el desarrollo de la aplicación fue Delphi 10 Lite, una versión minimizada con las principales características de Delphi 2006. Es un lenguaje de propósito general orientado a objetos y eventos, basado en pascal. Su entorno de desarrollo y sus componentes visuales permiten un rápido y fácil diseño de interfaces, de esta manera uno se enfoca más en la funcionalidad y resolución del problema. A pesar de que no cuenta con la abundante documentación como lo hacen otros lenguajes, en la red se pueden hallar un sin fin de ejemplos, VCL's (librerías de componentes visuales) gratuitos y en su mayoría con código fuente. Otro aspecto importante es que su código no es obsoleto a través de las diferentes versiones, a menos que se utilicen funcionalidades novedosas de cada versión sobre una anterior. Su compilador optimiza el código reduciendo la creación de código basura brindando un mejor desempeño.

Las bibliotecas (gratuitas) que se utilizaron fueron GLScene [29] y VideoLab [30]. La primera es una biblioteca que implementa el OpenGL para la creación de gráficos 3D, pero también permite el uso OpenGL puro. Por otra parte, la segunda biblioteca proporciona objetos que manipulan y adquieren el video.

En la figura 5.9 se muestra la interfaz de la clase *EstractorDePuntos*. Aquí fue donde se utilizó la biblioteca de VideoLab. Ayudó a establecer la comunicación de la cámara con la aplicación a través del objeto *TVLDSCapture*, así como la visualización del video en una especie de lienzo llamado *TVLDSImageDisplay*. En ○ se tiene a los botones para seleccionar una cámara disponible en el sistema, la búsqueda de los dispositivos de entrada la lleva a cabo de manera automática el

primer objeto, también proporciona algunas propiedades de la cámara como son: brillo, nitidez, contraste, entre otros. Con el segundo botón de ○ se muestra una ventana desarrollada en este trabajo que permite la manipulación de las propiedades descritas anteriormente. El usuario también tiene la opción de almacenar en disco duro el video del análisis, esto se hace con ayuda de otro objeto (*TVLDSVideoLogger*) de la biblioteca de VideoLab.

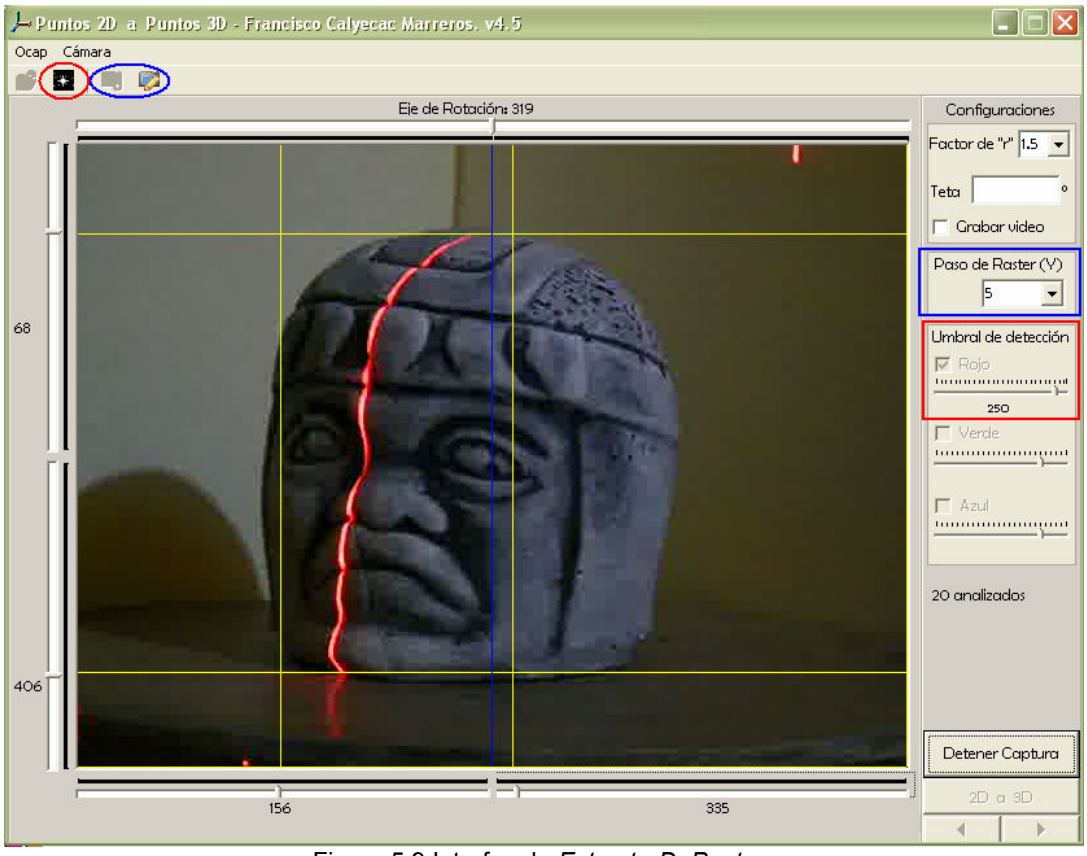


Figura 5.9 Interfaz de *EstractorDePuntos*.

Nótese (figura 5.9) que la región de búsqueda puede ser acotada por el usuario según convenga, así como la traslación del eje de rotación. Otros parámetros configurables, dichos en el análisis, se implementaron, como son: □ el indicar el paso del raster sobre la imagen, □ el señalar el umbral de detección del componente rojo en la imagen y el *factor de* r, que es un escalar que multiplica el valor de r (descrito en la figura 3.4 del capítulo 3) para que el modelo 3D resultante no quede angosto. Con el botón en ○ se muestra la ventana del visualizador 3D. Una vez acotada la región y preparadas las configuraciones se procede al análisis de

los frames. Esto se hace con el botón “Detener Captura”, inicia y detiene el procesamiento del video. Es aquí donde los métodos, *prepararDetecStrip(), encontrarStrip(), llenarHuecos()* y *mapping2Dto3D()*, se ejecutan; el último método lo hace previo a la petición del nombre y ruta del archivo donde se almacenarán los puntos 3D.

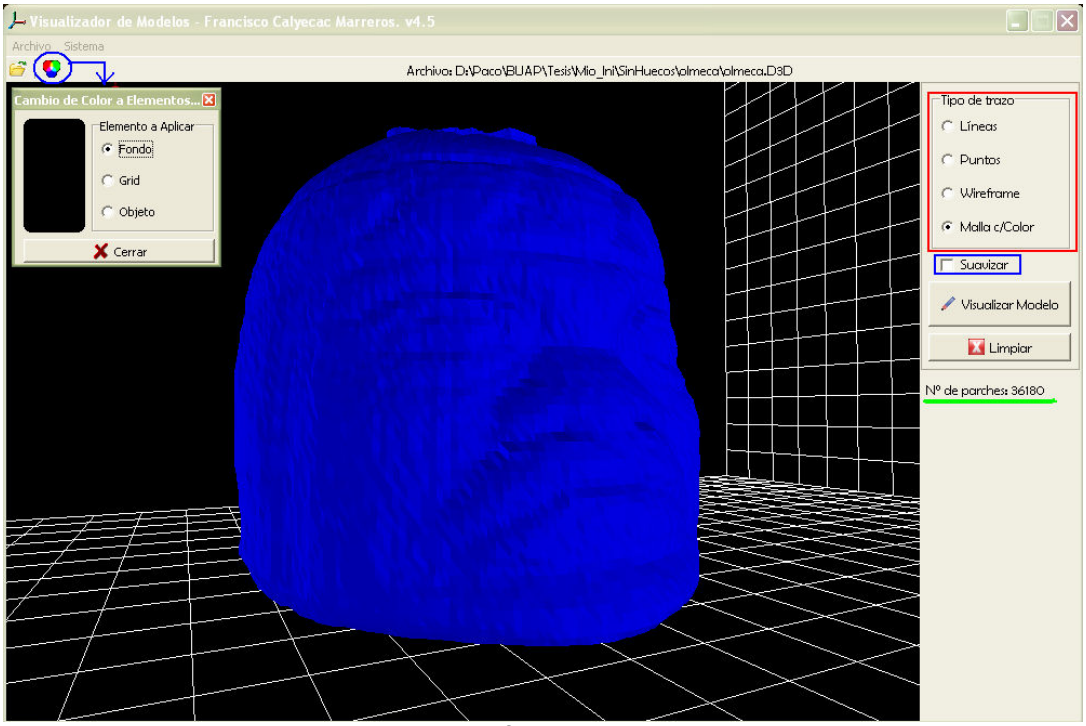


Figura 5.10 Interfaz del *Visualizador 3D*.

Hecho lo anterior, ahora toca el turno al visualizador 3D, en la figura 5.10 se aprecia la interfaz. Como es de pensar, aquí fue donde se utilizó la librería GLScene. Alguno de los objetos utilizados fueron: el *TGLSceneViewer*, un lienzo en donde se muestran las escenas 3D; el *TGLScene* un administrador de todos los objetos 3D; un objeto de vital importancia es el *TGLCamera*, sin una cámara no se puede ver absolutamente nada de la escena, generalmente siempre va acompañada de un *dummycube* (elemento fundamental y conocido en los gráficos 3D, en este caso *TGLDummyCube*) ya que sirve como objetivo a seguir por parte de la cámara; el *TGLXYZGrid* se utilizó como superficie del mundo 3D; las lámparas (*TGLLightSource)* son igual de importantes como una cámara, para una buena visualización del modelo influye en gran medida la configuración de varios de sus parámetros como son: la posición, la dirección, el tipo, la atenuación de la luz,

entre otras; el *TGLMesh* también es un objeto fundamental en esta aplicación porque en él se codifican los triángulos a través de los puntos de la matriz *modelo[][][3]*.

La aplicación permite 5 formas de visualización incluyendo el suavizado. Un extra que tiene la aplicación es el cambio de colores a la escena de fondo, al grid o al objeto. Se utilizaron splines cúbicos para el suavizado del modelo con ayuda de procedimientos de la biblioteca GLScene, por cado nodo (vértice) se generaron dos más.

En el siguiente apartado (5.3), se muestra mediante un diagrama de estados (figura 5.11) la lógica de todo el Sistema de Escaneo 3D desarrollado en este trabajo de tesis.

5.3 Lógica del Sistema

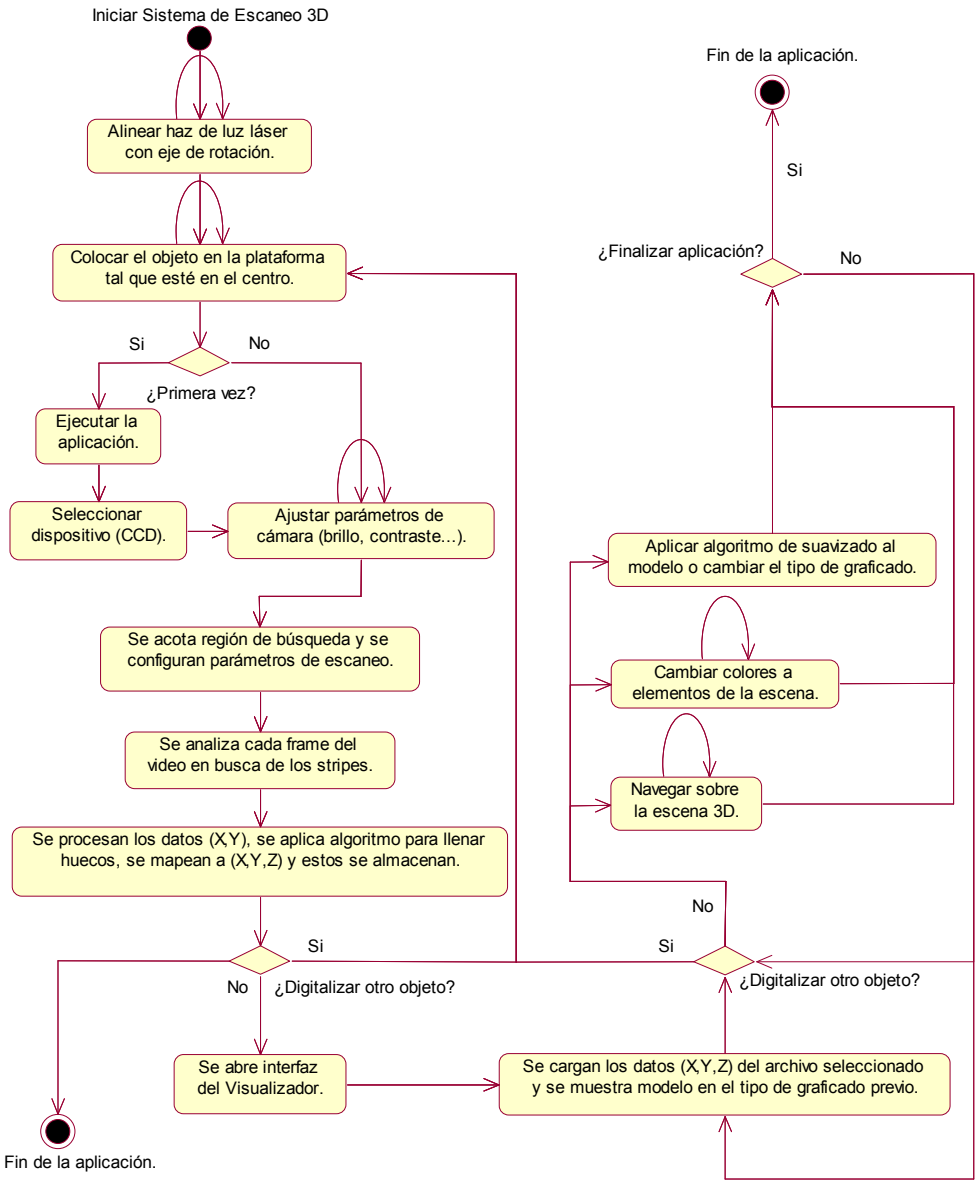


Figura 5.11 Diagrama de estados del Sistema de Digitalización 3D.

Capítulo 6

6 PRUEBAS Y VERIFICACIÓN

En este capítulo se resumen las pruebas del sistema de escaneo 3D. Se mostrarán los resultados de cuatro modelos 3D. También se hará mención sobre las dificultades, observaciones y soluciones encontradas durante este proceso. Las pruebas y verificación se llevarán a cabo en base a la lógica del sistema representado por el diagrama de estados (figura 5.11) del capítulo anterior.

6.1 Resultados

La figura 6.1 muestra el proceso de alineación. Una vez generado el haz de luz láser tipo línea se colocó un objeto pequeño en el centro de la plataforma para ayudarse visualmente y llevar a cabo la correspondencia del eje de rotación y la línea.

Figura 6.1 Alineación del sistema de digitalización 3D.

Es oportuno comentar que existieron inconvenientes con la fuente de luz y el lente refractor. En un inicio se trabajaba con un apuntador láser común de lapicero con baterías tipo botón, pero su consumo era excesivo. Entonces se le adaptó una fuente de voltaje de 5 volts – 300 mA. Por un tiempo, el primer apuntador funcionó adecuadamente, sin embargo se quemó. Existió un segundo, que al momento de soldar los cables de alimentación se excedió de temperatura y también se quemó. En otros dos apuntadores subsecuentes se logró soldar, pero su tiempo de vida disminuyó drásticamente en cuestión de minutos y dejaron de funcionar. Finalmente se acertó al utilizar un láser de mayor calidad que estuvo incluido en un control de cañón y se empleó con pilas AAA recargables como se dijo en la implementación.

Otro inconveniente fue el lente cilíndrico, que debido a su material de baja calidad la luz era refractada con ruido; es decir, el haz de luz láser tipo línea resultante proyectado en el objeto se mostraba ensanchado en algunas regiones, en otras estaba difuminado y era pequeño de longitud. Se probaron varias opciones, inclusive con un lente cilíndrico convexo y cóncavo (diminutos por cierto), el haz generado era ideal. El impedimento fue que pertenecían al laboratorio de óptica y eran muy costosos; no obstante, el encargado sugirió el uso de varillas de vidrio (agitadores). Y así fue como el sistema llegó a emplear este refractor.

Una anécdota más, fue la adquisición errónea del motor de microondas, era de 12 volts, se enchufó a un tomacorriente de 127 volts. Se tuvo que adquirir uno más.

En la siguiente tabla (1) se muestran las características de 4 de los más de 10 objetos escaneados con este sistema.

	Objeto	**Peso Aprox.**	**Dimensiones aprox. (ancho-largo)**	**Altura**	**Material**
1	Escuincle (perro azteca)	200 gr.	5x7 cm	8cm	Barro
2	Cabeza olmeca	1 Kg.	7x8 cm	10 cm	Piedra blanca
3	Vasija	1 Kg.	14x14 cm	17cm	Barro
4	Búho	5 Kg.	13x13 cm	19 cm	Piedra

Tabla 1: Características de 4 objetos escaneados.

La figura 6.2 expone las imágenes de estos cuatro objetos.

Figura 6.2 Objetos a escanear.

Ahora, en la tabla 2, se muestran las configuraciones realizadas al sistema previo al escaneo de cada una de las cuatro piezas.

	Objeto	Umbral de detección. (canal rojo)	Paso del raster.	Factor 'r'
1	Escuincle (perro azteca)	210	5	1.5
2	Cabeza olmeca	225	5	1.5
3	Vasija	205	5	2
4	Búho	200	5	1.5

Tabla 2: Configuraciones previas al escaneo.

A continuación se muestran los modelos 3D obtenidos de cada una de las piezas. La figura 6.3 pertenece al modelo del escuincle (1), la figura 6.4 a la cabeza olmeca (2), la figura 6.5 a la vasija (3) y finalmente la figura 6.6 al búho (4).

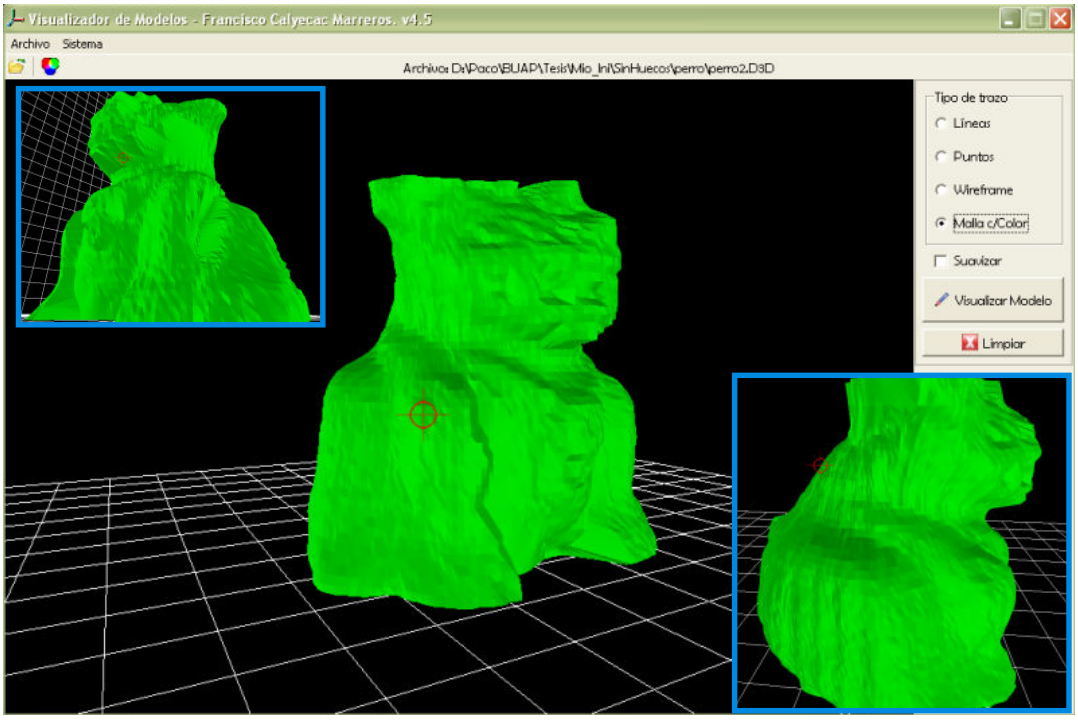


Figura 6.3 Modelo 3D del escuincle (1).

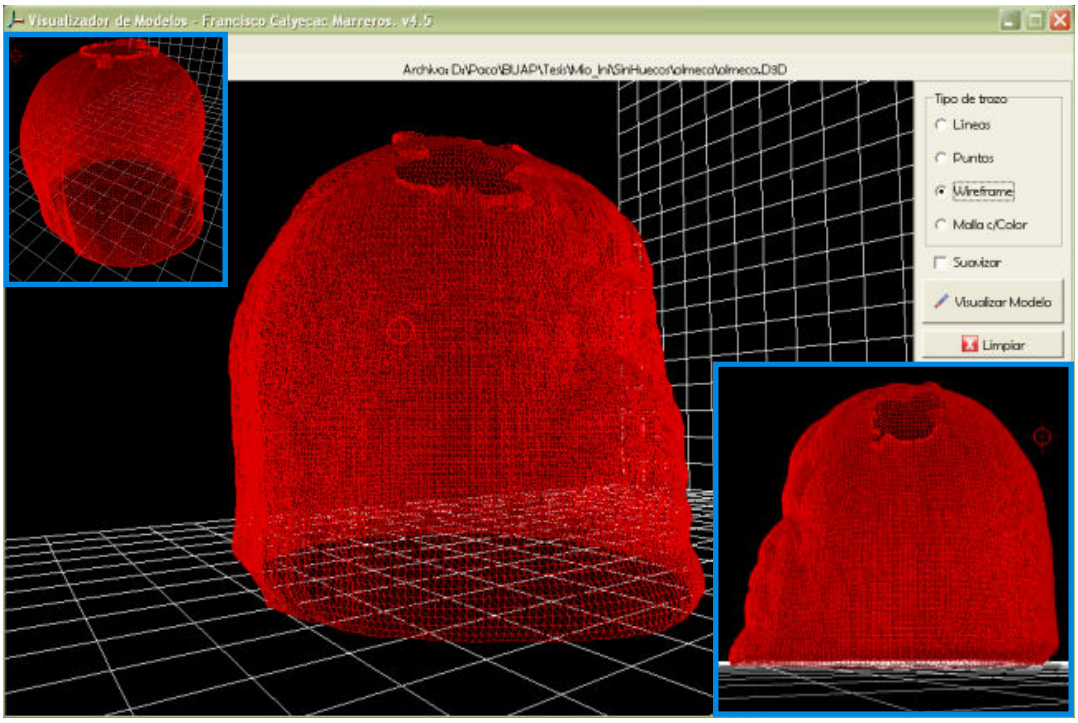


Figura 6.4 Modelo 3D de la cabeza olmeca (2), grafi-
cado wireframe.

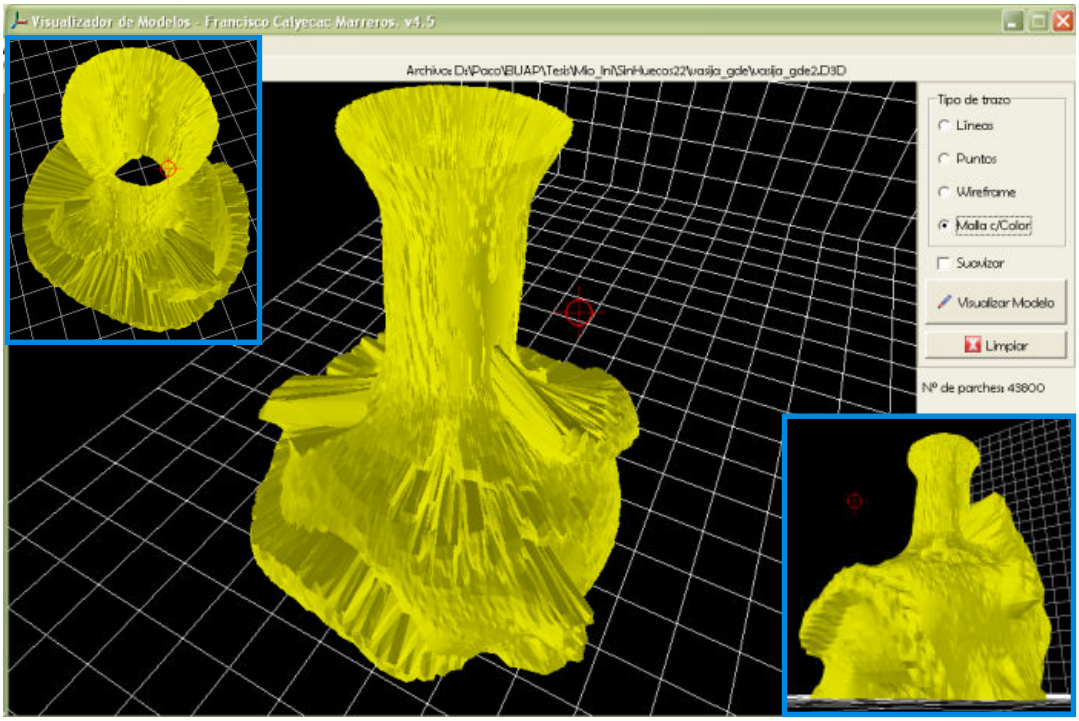


Figura 6.5 Modelo 3D de la vasija (3).

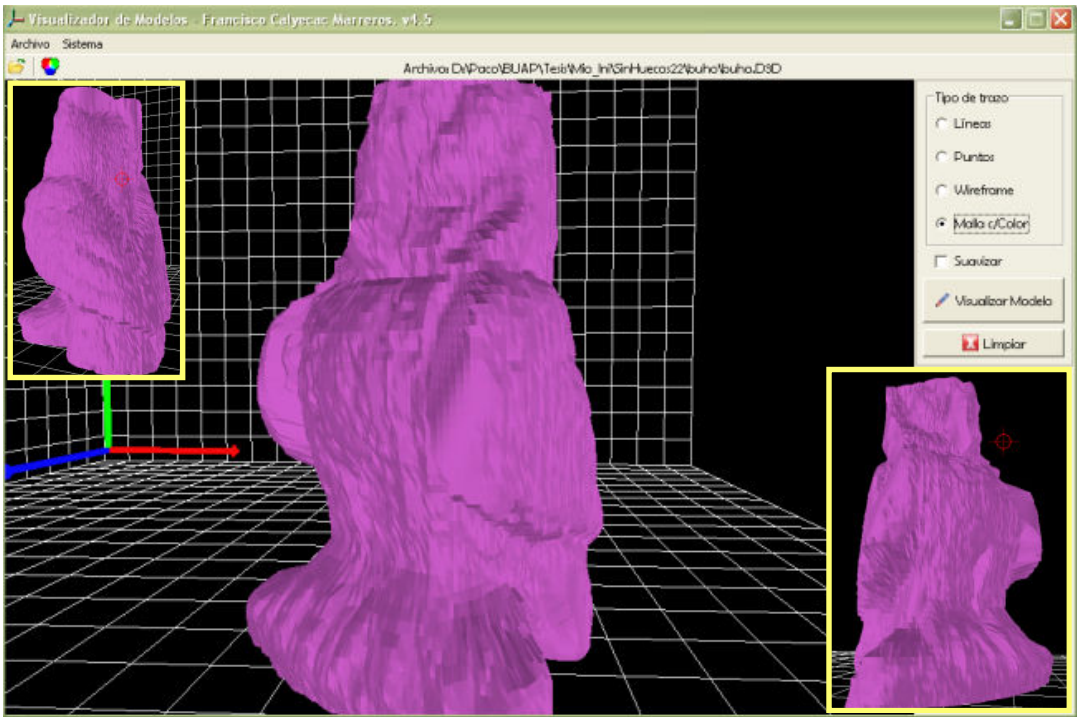


Figura 6.6 Modelo 3D del búho (4).

La tabla 3 muestra el número de parches triangulares formados para cada modelo escaneado, nótese que al aplicar el suavizado el número de parches crece en un factor mayor a diez.

	Objeto	# de parches S / Suavizar	# de parches Suavizado
1	Escuincle (perro azteca)	26,010	264,690
2	Cabeza olmeca	36,180	366,660
3	Vasija	43,800	443,400
4	Búho	40,800	413,400

Tabla 3: Número de parches de los modelos escaneados sin y con suavizado, y su respectivo tiempo de graficado.

En la figura 6.7 se tienen dos modelos del escuincle, del lado izquierdo sin suavizar y del lado derecho suavizado. El color de los modelos discrepante es para poder apreciar esta diferencia.

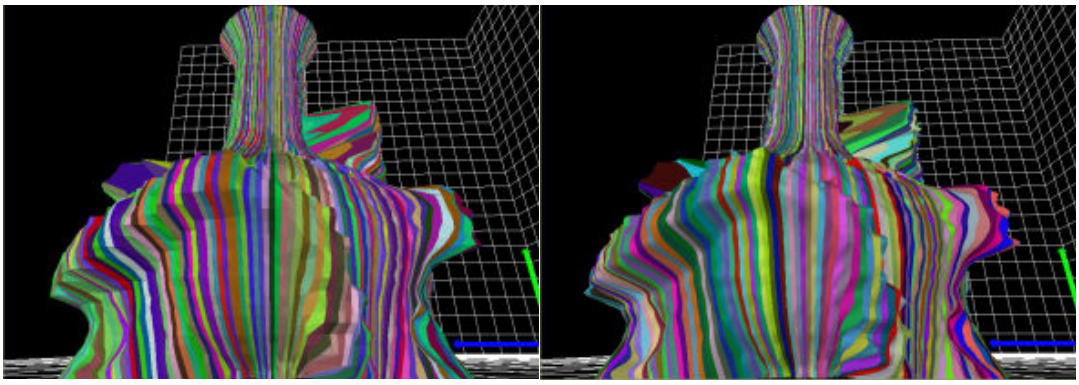


Figura 6.7 (Izq.) Vasija sin suavizar. (Der.) Vasija suavizada.

La primera ocasión que se aplicó el suavizado a dos modelos sin cerrar la aplicación se provocaba un error de *memoria insuficiente*. La aplicación llegaba a consumir más de un GB de memoria RAM. Después de un análisis se encontró que el problema estaba en la *no liberación* de los objetos que creaban los splines cúbicos.

Un último aspecto importante que se observó fue que se obtiene una mejor detección de puntos (de cada frame) si la calidad-resolución del video que ofrece el CCD es más alta. Si esto sucede, entonces los huecos de la matriz (discutidos en 5.1.2) se reducen; es decir, se producirán menos datos artificiales.

CONCLUSIONES

Este trabajo de tesis alcanzó sus objetivos principales. Se logró la implementación de un sistema de digitalización 3D a bajo costo sin sacrificar mucho detalle en la adquisición de la forma del objeto.

Los conceptos básicos de gráficos 3D fueron necesarios para la implementación del sistema, porque de ellos depende una buena visualización de las escenas 3D.

Con ayuda de la biblioteca de VideoLab se logró escanear los objetos en tiempo real. La matriz con los puntos 2D se llena en ese instante y una vez finalizado el giro del objeto se procesan inmediatamente los datos para el mapeo de espacios (XY a XYZ).

También se logró el suavizado de los modelos mediante splines cúbicos.

Durante los albores de este trabajo se participó en el programa de Jóvenes Investigadores por parte de la Vicerrectoría de Investigación y Estudios de Posgrado de la BUAP.

También se publicó en la revista de proyectos de la Facultad durante el Congreso CONCIC 2008. ISBN: 978-607-7541-89-9, “Escaneo 3D de Piezas Arqueológicas”.

Con este trabajo se obtuvo el 2do lugar en el área de computación en la 4ta Convención de Investigación Aplicada y Desarrollo Tecnológico 2008 por parte del CONCYTEP del estado de Puebla.

Limitaciones

Por la naturaleza del escáner se presenta un inconveniente, no se puede escanear la parte inferior y superior del objeto, porque el haz de luz no puede cubrir esas zonas y mucho menos la cámara tiene el campo de visión.

Otra desventaja que presentan los escaners que utilizan luz estructurada, es la falta de robustez ante los hoyos que pudiera tener el objeto, sería una tipo de

oclusión que no bastaría con el promedio de los vecinos, porque no habría manera de saber que tan profundo y que forma tiene el hoyo.

Perspectivas, trabajo futuro

Obtener la información de la parte superior e inferior del objeto podría solucionarse si se mezcla otra técnica de escaneo donde se tenga que calibrar la cámara y tomar más muestras del objeto desde otros puntos de vista, para después llevar a cabo el proceso de alineación y registro de puntos.

El problema de los hoyos puede solucionarse parcialmente con *range sensors*.

Un trabajo futuro para esta tesis es la implementación del mapeo de la textura del objeto al modelo 3D para darle mayor realismo.

BIBLIOGRAFÍA

[1] Shirley Meter. *Fundamentals of Computer Graphics.* AKPeters LTD, 2005.

[2] Shalini Govil-Pai. *Principles of Computer Graphics: theory and practice using OpenGL and Maya.* Springer, 2004.

[3] Durand Frédo. *A Short Introduction to Computer Graphics.* http://people.csail.mit.edu/fredo/ArtAndScienceOfDepiction/1_Introduction/reviewGraphics.pdf

[4] Salomon David. *Curves and Surfaces for Computer Graphics.* Springer, 2006.

[5] Chandra Satish. *Engeneering Graphics.* Alpha Science, 2003.

[6] R. Buss Samuel. *3D Computer Graphics: A Mathematical Introduction with OpenGL.* Cambridge, 2003.

[7] Klawonn Frank. *Introduction to Computer Graphics: Using Java 2D and 3D.* Springer, 2008.

[8] Escribano Manuel. *Programación de Gráficos en 3D.* Addison-Wesley, 1995.

[9] Sonka Milan, Boyle Roger. *Image Processing Analysis and Machine Vision.* Thomson, 2008.

[10] Crane Randy. *A Simplified Approach to Image Processing.* Prentice Hall, 1997.

[11] Russ John. *The Image Processing Handbook.* Springer, 1999.

[12] Dugelay Jean-Luc, Baskurt Atilla. *3D Object Processing: Compression, Indexing and Watermarking.* Wiley, 2008.

[13] Trucco, Verri Alessandro. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[14] ZEISS Corporation. *http://www.zeiss.com/ct-services-3d*.

[15] Pages Jordi. *Assisted Visual Servoing By Means of Structured Light*. PhD Thesis, Universitat de Girona, 2005.

[16] Robayo D., Valderrama Z. *Reconstrucción 3D de Objetos Discontinuos Usando el Método de Código de Grises*. Revista Colombiana de Física, Vol. 35, Nº 2 2003.

[17] Laser Design Inc. *http://www.laserdesign.com/services/long_range_scanning*.

[18] Bernardini F. y Rushmeier H. *The 3D Model Acquisition Pipeline*. Computer Graphics forum, Vol. 21, N° 2.

[19] Bhanu B. y Pavlidis I. *Computer Beyond the Visible Spectrum*. Springer, 2005.

[20] Modesti M., Pedrazzini M. *Calibración de un Sistema de Visión Estéreo para Navegación de un AGV*. Córdoba, Argentina.

[21] Levoy M., Pulli K., Fulk D. *The Digital Michelangelo Project: 3D Scanning of Large Statues*. Siggraph 2000.

[22] Salvi M. Joaquim. *An Approach to Coded Structured Light to Obtain Three Dimensional Information*. PhD Thesis, Universitat de Girona, 1998.

[23] Zhang Song, Huang Peisen. *High Resolution, real-time three-dimensional shape measurement*. Optical Engineering 45 (12). December 2006.

[24] Ricolfe C., Simarro R. *Técnicas de Calibrado de Cámaras*. Dpto. de Ing. de Sistemas y Automática, Universidad Politécnica de Valencia.

[25] StockerYale Inc. *http://www.stockeryale.com*.

[26] Dispersión y difracción. *http://www.xtal.iqfr.csic.es/Cristalografia/par-te_05.html*

[27] Dyson J. *Circular and spiral diffraction gratings.* Royal Soc. of London, 1958.

[28] Óptica: Láseres, Lentes, Fibras ópticas. *http://www.directindustry.es/cat /optica-laseres-lentes-fibras-opticas/lentes-opticas-AC-551.html*

[29] Sitio de GLScene. *http://glscene.sourceforge.net/wikka/HomePage*

[30] Sitio de VideoLab Mitov. *http://www.mitov.com/*

APÉNDICE A

Manual de Usuario

Este pequeño manual de usuario de la aplicación está dividido en dos partes: la primera depende del hardware del sistema, razón por la que se supondrá que se cuenta con él para poder explicar el funcionamiento de la aplicación. La segunda trata sobre el manejo del visualizador 3D.

En el supuesto que se cuenta con el hardware, se debe realizar el proceso de alineación como se muestra en la figura 6.1 del capítulo 6. Al ejecutar la aplicación *"Escáner3D.exe"* la primera interfaz que aparece es la de la figura A.1.

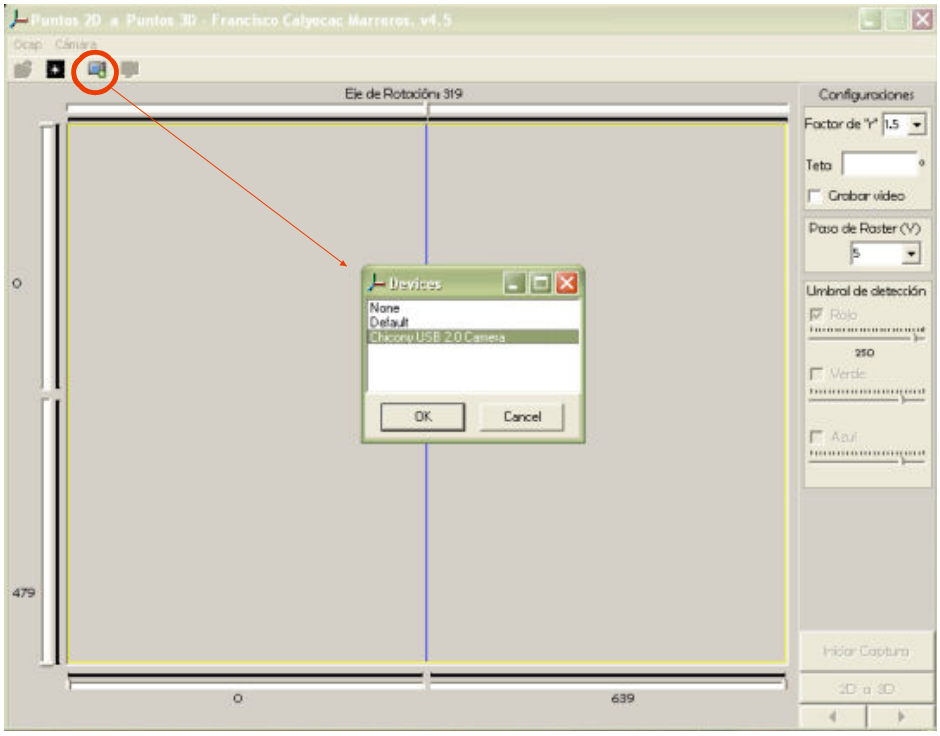


Figura A.1 Interfaz de la aplicación Escáner 3D (seleccionando CCD).

Lo primero que se debe hacer es seleccionar una cámara, elige en el menú *Cámara→ Elegir Dispositivo* ó en la barra de herramientas da click en el único

botón disponible con el icono de una cámara. Es importante tener una cámara instalada en la PC, *none* y *default* no son una opción válida.

El siguiente paso es (figura A.2) acotar la región de búsqueda del stripe para acelerar el proceso de análisis de puntos, también es posible mover el eje de rotación (línea azul) si es necesario. El menú *Cámara → Configuraciones* ó el botón encerrado en el círculo azul que muestra la imagen, facilitan las opciones de configuración de la cámara tales como: brillo, contraste, nitidez, entre otros.

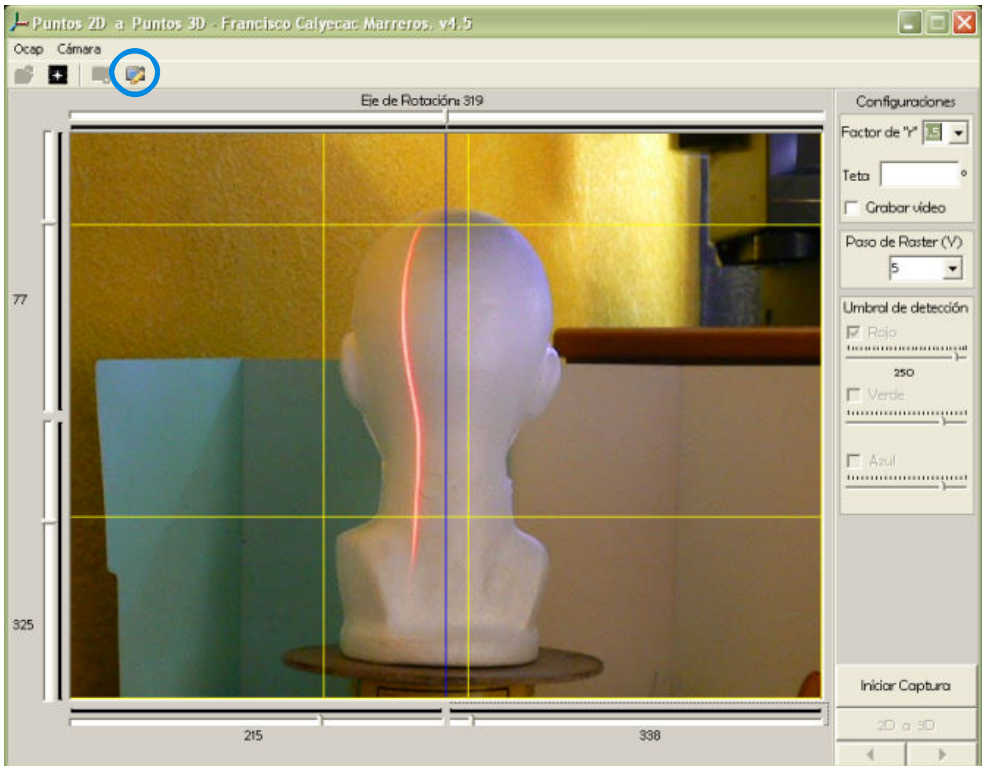
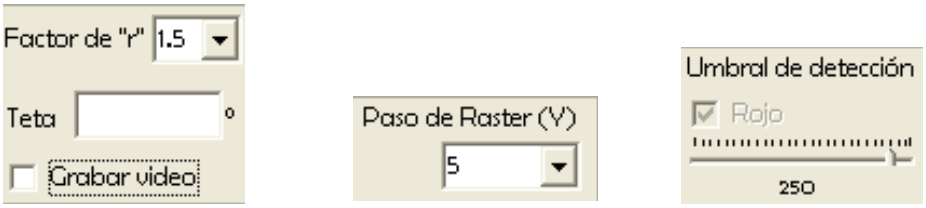


Figura A.2 Acotando región de búsqueda.

El último paso previo a iniciar el escaneo es la configuración de parámetros del panel lateral derecho. En la figura A.3 a) se debe seleccionar un valor para el *factor r* (descrito en la figura 3.4 del capítulo 3), por defecto es 1.5, en las pruebas realizadas se observó que 1.5 y 2 son buenos valores. También se puede elegir grabar el video, se recomienda hacerlo solo cuando las características de la PC sean iguales o superiores a las descritas en 5.2.1, de lo contrario el escaneo podría no ejecutarse en tiempo real. En el inciso b) de la misma figura el valor por defecto del *paso del raster* (ver 5.1.2 → *prepararDetecStrip()*) es 5, el valor puede ser

diferente. Finalmente en c) es necesario disminuir el umbral de detección del canal rojo porque debido a las condiciones de iluminación ambiente, a la calidad del video y al material del objeto el valor 250 no resulta conveniente, se recomienda colocarlo en 220.



a) b) c)

Figura A.3 Configuraciones previas al escaneo.

Hecho lo anterior, fíjate en la posición del objeto y presiona el botón *Iniciar_Captura*, cuando se haya completado una vuelta detén la captura con el mismo botón.

Internamente, después de detener la captura se verifica si hay huecos en la matriz (ver 5.1.2), si los hay, se ejecuta el algoritmo para llenarlos hasta por cuatro veces e indica el número de huecos restantes. Inmediatamente después, solicita el nombre del archivo (con extensión *D3D*) donde se guardarán los datos 3D.

Todo el proceso anterior se puede llevar a cabo nuevamente con otro objeto o con el mismo. De lo contrario se puede abrir el visualizador 3D con *Ocap→ Graficar modelo 3D* o con el botón en la barra de herramientas que tiene el icono de una estrella blanca y fondo negro.

La interfaz del visualizador se muestra en la figura A.4. Existen cinco tipos de trazo o graficado incluyendo el suavizado; líneas, puntos, wireframe y mesh con color de relleno. Por defecto, el tipo de graficado es con mesh y color azul. Con *Ctrl+O* se abre un archivo (**.D3D*) y se grafica automáticamente. Cuando se tiene presionado el botón izquierdo del ratón se puede navegar alrededor de la escena y se puede subir o bajar con respecto al eje Y cuando se sostiene el botón derecho. Una vez cargado el modelo se puede suavizar.

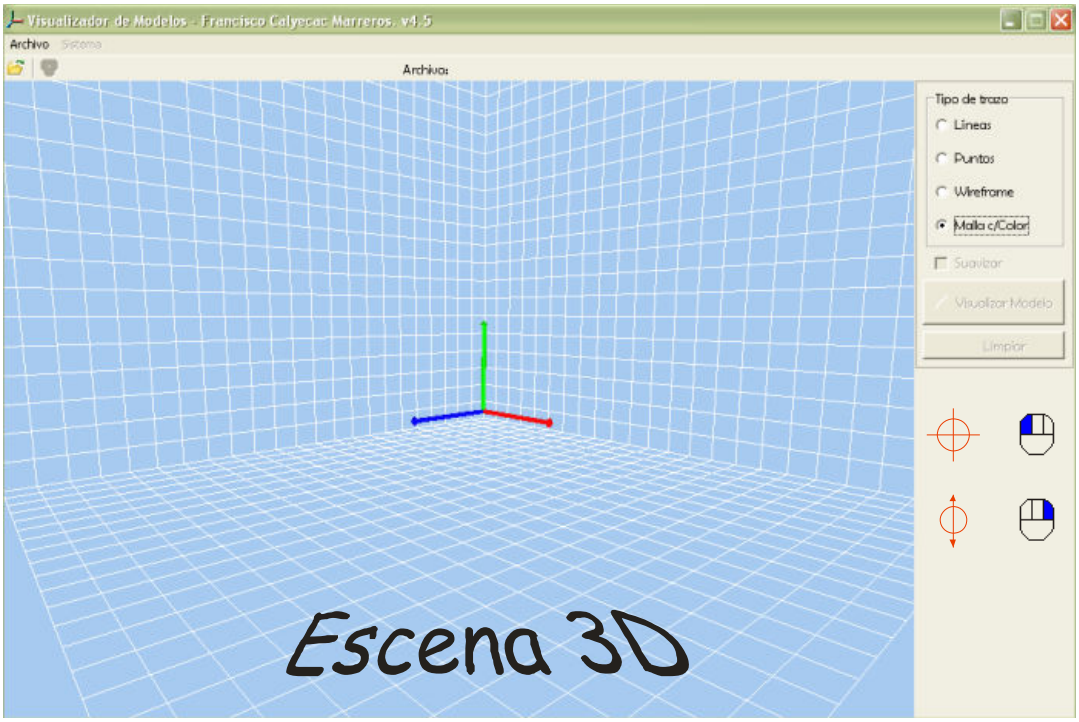


Figura A.4 Interfaz del visualizador 3D.

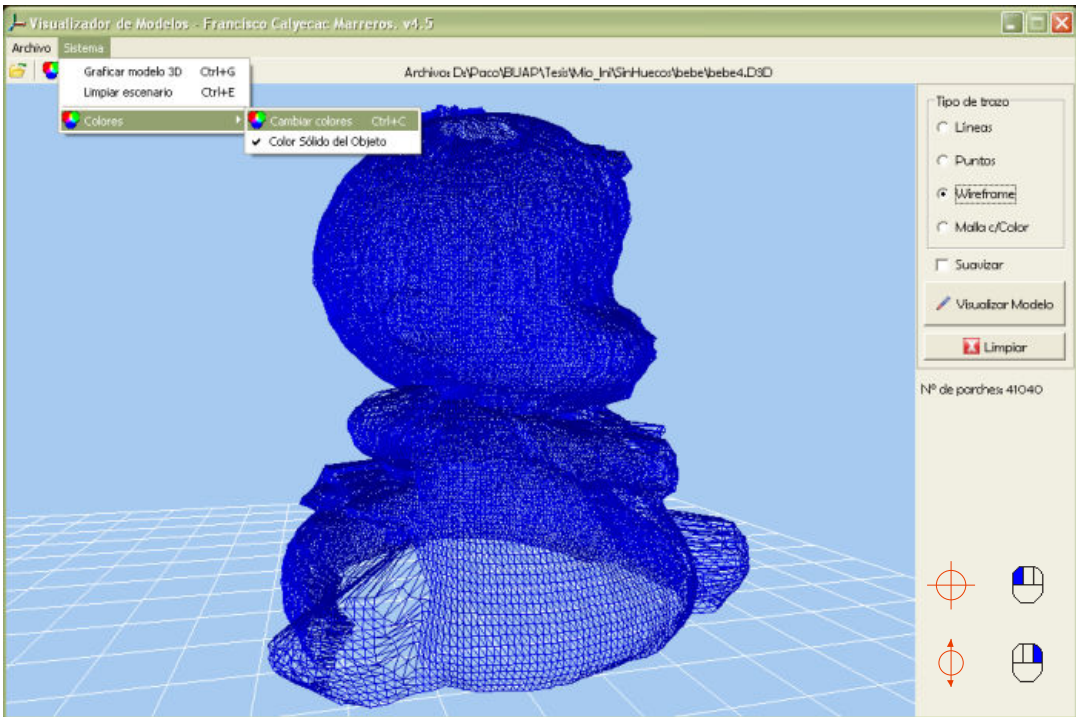


Figura A.5 Visualizando un bebé en wireframe.

En la figura A.5 se muestra el modelo de un bebé de porcelana en wireframe. También se tiene la opción de cambiar el color a los componentes de la escena. En *Sistema→ Colores → Cambiar colores* se abre una ventana donde seleccionas el color del fondo de la escena, del grid o del objeto. Esto sucede siempre y cuando *Sistema→ Colores → Color sólido del objeto* esté habilitado, de lo contrario el modelo 3D se pintará con colores aleatorios como se observa en la figura A.6.

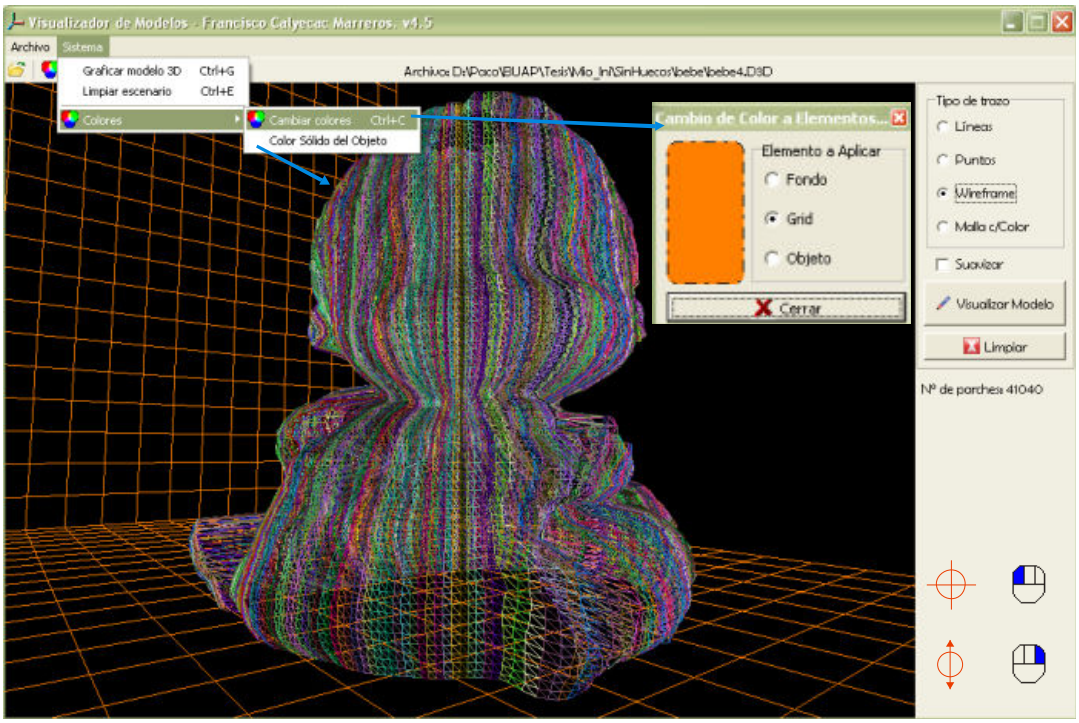


Figura A.6 Bebé en wireframe con colores aleatorios.