UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE PARA RECONSTRUCCIÓN TRIDIMENSIONAL DE OBJETOS A PARTIR DE IMÁGENES, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL"

REALIZADO POR:

López L., Lenin Armando

Sandoval R., José Miguel

Trabajo de grado presentado como requisito parcial para optar al Título de
INGENIERO EN COMPUTACIÓN

Barcelona, Agosto de 2010

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE PARA RECONSTRUCCIÓN TRIDIMENSIONAL DE OBJETOS A PARTIR DE IMÁGENES, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL"

ASESOR:

_____

Ing. Stefano Larese.
Asesor Académico

Barcelona, Agosto de 2010

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE PARA RECONSTRUCCIÓN TRIDIMENSIONAL DE OBJETOS A PARTIR DE IMÁGENES, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL"

JURADO CALIFICADOR:

_____

Ing. Stefano Larese
Asesor Académico

_____

Profesor Claudio Cortínez
Jurado Principal

_____

Profesor José Luis Bastardo
Jurado Principal

Barcelona, Agosto de 2010

RESOLUCIÓN

De acuerdo con el artículo 41 del Reglamento de Trabajos de Grado:

“Los Trabajos de Grado son de la exclusiva propiedad de la Universidad de Oriente, y sólo podrán ser utilizados para otros fines, con el consentimiento del Consejo de Núcleo respectivo, quien deberá participarlo al Consejo Universitario, para su autorización”

DEDICATORIA

Principalmente quiero dedicar este trabajo de grado a *Dios* por ser mi guía espiritual y darme la fortaleza necesaria para poder culminar esta etapa de mi vida.

A mis padres, *Manuel López y Eunice Loroima*, por su ayuda, apoyo incondicional, amor en todo momento y por ser ellos el incentivo para hacer realidad mí más anhelado sueño.

A mi única hermana *Hindira*, por haberme brindado su apoyo en los momentos más importantes de mi vida. Que este trabajo de grado, que encierra años de constancia y dedicación, sirva de ejemplo para la culminación de sus estudios. Te quiero mucho Hermana.

A toda mí *familia y amigos* que de alguna u otra forma estuvieron conmigo en los momentos más difíciles a lo largo de mí carrera.

Lenin Armando López Loroima

DEDICATORIA

Este trabajo este dedicado a Dios todo poderoso por que hasta ahora me ha dado la salud necesaria para continuar adelante.

A mis padres, José Inés Sandoval Mata, Milena Josefina Rojas Montilla, por todo ese amor y apoyo incondicional que me han brindado en todo momento

A mis hermanos, Neuris Marina, Nereida José, Héctor Enrique, milena Cristina, que siempre han estado a mi lado brindándome todo el apoyo que he necesitado

A toda mi familia que siempre ha estado a mi lado…

José Miguel Sandoval Rojas

AGRADECIMIENTOS

Agradezco primero que nada a *Dios* por haberme permitido alcanzar mí meta.

A mis *Padres*, por haberme apoyado y por estar siempre conmigo en los momentos de alegría y de tristeza a lo largo de mi carrera.

A mí compañero de tesis, *José Miguel Sandoval* por haber compartido conmigo esta etapa de mí vida.

A mi asesor, *Stefano Larese* por guiarnos durante todo este tiempo, parte del éxito es de él.

A la profesora, *Zulirais Garcia* por su ayuda y consejos, gracias por prestarme parte de su tiempo y atención.

Al profesor, *Tirso Garcia* por su ayuda y consejos, y por los momentos difíciles que me hizo pasar en matemática discreta, gracias.

A todos los profesores del Departamento de Computación y Sistemas, por contribuir a mi formación como profesional.

A mis grandes amigos, *Pablo Smeja*, *Luis cordero (chino), Carlos Mónaco, Richard Fernández, Javier Pérez, Enoes Medina, Luis Martínez, Roneld Zapata, Roger Garban, Dimas, José Ángel Hernández, Liliana Varrone, María Luisa y Hendri Guzman,* por su amistad, por acompañarme cuando más los necesite, por su apoyo incondicional durante mi carrera y en mi vida personal.

A todos aquellos que de una u otra forma pusieron su granito de arena, para que se hiciera posible este sueño de graduarme de Ingeniero en Computación.

Lenin Armando López Loroima

AGRADECIMIENTOS

Agradezco a Dios por darme el aliento y la sabiduría, por mostrarme el camino para poder llegar al final de esta meta.

A mis Padres y mis Hermanos, por haberme apoyado y por estar siempre conmigo en los momentos de alegría y de tristeza a lo largo de mi carrera.

A mí compañero de tesis y amigo Lenin Armando López Loroima, fue un placer haber realizado este trabajo de grado contigo.

A mi asesor, Stefano Larese por la confianza que mostro en nosotros para sacar adelante este tema de tesis, y a su esposa la profesora Zulirais Garcia por la ayuda prestada durante la redacción de este trabajo de gado.

A mis compañeros de carrera y amigos, Pablo Smeja, Luis cordero (chino), Carlos Mónaco, Richard Fernández, Javier Pérez, Enoes Medina, Luis Martínez, Roger Garban, Dimas, José Ánge (enano), Hendri Guzman, porque sin ustedes no hubiese podido seguir adelante, fueron una pieza fundamental para la culminación de mi carrera.

A mis amigos profesores de departamento. Tirso García, Víctor Mujica, Héctor Moises, Reinaldo Pastrana, Gabriela Veracierta, Aida Caraballo, Liliana Varrone, José Luis Bastardo, Luis Solórzano, Lenin Benítez, gracias a todos por la amistad brindada y la formación académica recibida, también agradezco aquellos profesores que aquí no nombro pero que formaron parte de mi formación como profesional.

A unas persona muy especiales que de alguna manera u otra también formaron parte de mi formación academica, María Luisa, Dayana, Sr. Maria, gracias por ser tres mujeres tan hermosas, que Dios las bendiga y las quiero un monton.

A la familia Lopez Loroima que me recibieron en su casa todo el tiempo que duro este capítulo de nuestras vidas, gracias por la amistad brindada.

A la familia Ramos Estrada, gracias viejos por esa bonita amistad.

A mis amigos y concañeros, Julio Ramos, Marien Alejandra, David gallardo, José Manuel (Papote), Ramon, Sergio, Oscar, Dairible, Carlibel, Jesus (Pine), Isble, Keila, siempre se necesitan momentos de RELAX y ustedes han estado allí para matar el estrés, se quiero mucho siempre hacen falta amigos como ustedes…

A todos aquellos que de una u otra forma pusieron su granito de arena, para que se hiciera posible este sueño de graduarme de Ingeniero en Computación.

José Miguel Sandoval Rojas

RESUMEN

El objetivo del trabajo de investigación titulado **"DESARROLLO DE UN SOFTWARE PARA RECONSTRUCCIÓN TRIDIMENSIONAL DE OBJETOS A PARTIR DE IMÁGENES, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL",** es la adquisición de imágenes y modelado tridimensional de objetos utilizando técnicas de visión artificial mediante la luz estructurada. Para ello se desarrollo una aplicación que consiste básicamente en proyectar en un plano una secuencia de patrones de luz sobre un objeto determinado, estos son captados por una cámara, posteriormente dichas imágenes son enviadas al computador quien las procesara, Utilizando técnicas de triangulación se obtiene la profundidad del objeto a reconstruir esto permitirá la generación del modelo tridimensional. Para la calibración de la cámara se utilizo el método de calibración propuesto por Zhang. Para la implementación de la aplicación, se uso el lenguaje de programación JAVA para la construcción de las interfaces de usuario, y el lenguaje de programación C++ para los motores de reconstrucción 3D y calibración de cámara, junto con las librerías JNI para la comunicación entre JAVA y C++ y OPENCV para el tratamiento de imágenes.

INDICE GENERAL

CAPITULO I
PLANTEAMIENTO DEL PROBLEMA

1.1 INTRODUCCIÓN

El término "inteligencia artificial" (IA) fue acuñado formalmente en 1956 durante la conferencia de Darthmounth, mas para entonces ya se había estado trabajando en ello durante cinco años en los cuales se había propuesto muchas definiciones distintas que en ningún caso habían logrado ser aceptadas totalmente por la comunidad investigadora. La IA es una de las disciplinas más nuevas que junto con la genética moderna es el campo en que la mayoría de los científicos " más les gustaría trabajar".

Puede decirse que la IA es una de las áreas más fascinantes y con más retos de las ciencias de la computación, en su área de ciencias cognoscitivas. Nació como mero estudio filosófico y razonístico de la inteligencia humana, mezclada con la inquietud del hombre de imitar la naturaleza circundante, hasta inclusive querer imitarse a sí mismo. Sencillamente, la IA busca el imitar la inteligencia humana. Obviamente no lo ha logrado todavía, al menos no completamente.

El inicio de la visión artificial, desde el punto de vista práctico,
por Larry Roberts, el cual, en 1961 creó un programa que podía "ver" una estructura de bloques, analizar su contenido y reproducirla desde otra perspectiva, demostrando así a los espectadores que esa información visual que había sido mandada al ordenador por una cámara, había sido procesada adecuadamente por él.

La visión artificial llegará a ser cada vez más importante. La investigación actual se está concentrando en producir sistemas más robustos y más versátiles.

Los objetivos típicos de la visión artificial incluyen:

- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas).
- La evaluación de los resultados (ej.: segmentación, registro).
- Registro de diferentes imágenes de una misma escena u objeto, i.e., hacer concordar un mismo objeto en diversas imágenes.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la escena; tal modelo podría ser usado por un robot para navegar por la escena.
- Estimación de las posturas tridimensionales de humanos.
- Búsqueda de imágenes digitales por su contenido.

Estos objetivos se consiguen por medio de reconocimiento de patrones, aprendizaje estadístico, geometría de proyección, procesado de imágenes, teoría de gráficos y otros campos.

La reconstrucción 3D permite la obtención de las propiedades tridimensionales de una superficie u objeto (coordenadas de vértices, aristas, caras, curvaturas, entre otras.) partiendo de imágenes o vistas planas en 2 dimensiones obtenidas ya sea de fotografías o cámaras de video. La información tridimensional obtenida, es mostrada en un computador por medio de herramientas de visualización y puede ser posteriormente modificada por medio de herramientas CAD. La tecnología de reconstrucción 3D ha tomado 2 métodos distintos para lograr su objetivo, estos son: reconstrucción mediante el uso de vistas planas obtenidas ya sea mediante fotografías o dibujos en 2D hechos en el computador. Reconstrucción en tiempo real mediante el uso de cámaras de video (u otros sensores), las cuales están conectadas al computador quién procesa la información a medida que la va recibiendo desde la cámara.

1.2 EL PROBLEMA

La Universidad de Oriente, Núcleo de Anzoátegui no se ha quedado atrás en cuanto a las investigaciones de este tipo, en ella se encuentra el grupo de investigación SISVA (Sistemas de Visión Artificial), perteneciente al Departamento de Computación y Sistemas. Este grupo está conformado por el Ing. Stefano Larese como coordinador, y los Ing. Claudio Cortines y Zulirais García como miembros. El grupo de investigación SISVA tiene como objetivo primordial, desarrollar prototipos de sistemas basados en visión artificial que resuelvan problemas de cierta complejidad. Entre los objetivos que persiguen se destacan:

- Promover el desarrollo del área de Visión Artificial

- Desarrollar la participación de profesores, estudiantes y profesionales relacionados con el área para la formación de grupos interdisciplinarios.

- Proponer trabajos de grado para incentivar la investigación de los nuevos profesionales.

Considerando este último objetivo se propone la creación de un sistema de reconstrucción 3D (Escáner 3D) basado en visión artificial.

La descripción del software es la siguiente:

a) El software recibirá una secuencia de imágenes de una rejilla proyectada sobre el objeto, en este caso obtenidas por medio de un dispositivo de captura de video. Aplicando algoritmos y técnicas de visión artificial se obtendrá una nube de puntos.

b) El proceso de reconstrucción 3D se llevara cabo mediante la aplicación de técnicas de mallado 3D a la nube de puntos. La idea es que el objeto real sea representado en la memoria del computador manteniendo sus características físicas (dimensiones, volumen, forma).
c) El resultado será una versión digital del objeto original almacenado en un archivo ".OBJ", que posteriormente podrá ser editado y manipulado por cualquier software de edición 3D.

1.3 PROPÓSITO Y METODOLOGÍA

Este proyecto está orientado a fomentar las investigaciones que realiza el grupo de investigación SISVA específicamente en el área de visión artificial, con la finalidad de construir una buena base para impulsar el desarrollo de nuevos trabajos de investigación en esta área.

Para el desarrollo de este proyecto se utilizaron técnicas de visión artificial, tales como el procesamiento de imágenes, algoritmos de calibración de cámaras, algoritmos de reconstrucción 3D, algoritmos de triangulación. Además de aplicar como metodología el proceso unificado y sus herramientas para guiar el desarrollo del software.

1.4 IMPORTANCIA Y ALCANCE.

Este proyecto será de suma importancia para el grupo de investigación SISVA ya que pretende introducir la reconstrucción 3D como elemento adicional para el estudio y resolución de problemas.

Con este proyecto se busca desarrollar un software que permita reconstruir un modelo 3D a través de la captura de imágenes 2D, de modo que es una técnica de

visión artificial innovadora dentro del Grupo de Investigación de Visión Artificial de la Universidad de Oriente.

1.5 OBJETIVOS

1.5.1 Objetivo General

Desarrollar de un software para reconstrucción tridimensional de objetos a partir de imágenes, usando técnicas de visión artificial.

1.5.2 Objetivos Específicos

- Realizar la descripción y especificación de los requerimientos del proyecto.
- Analizar las diferentes técnicas de visión artificial que nos permitan desarrollar el proyecto.
- Diseñar los módulos acordes de los requerimientos del proyecto planteado.
- Codificar los módulos ya diseñados en un leguaje de programación.
- Realizar la integración y documentación de todos módulos que conforman el proyecto.
- Realizar pruebas de integración y de software.

1.6 MARCO TEORICO

1.6.1 ANTECEDENTES

- **Llaraza W. y Bravo J. (2000) "Desarrollo de un software para obtener un ambiente virtual partiendo de las primitivas geométricas extraída de fotos estáticas utilizando técnicas de procesamiento de imágenes".** Este trabajo se trata sobre el desarrollo de un software que se encarga de ver una imagen con la extensión JPG para luego procesarla con una serie de algoritmos e interpretarlas como primitivas geométricas y representar esta primitivas en el lenguaje de modelado virtual VRML [1].

- **Romero L., Meneses J., Rodríguez A., Gonzáles O. (2006) "Sistema de reconstrucción tridimensional para el análisis dinámico de un cuerpo: estudio cuantitativo del vulcanismo de lodo".** En este trabajo se plantea un alternativa que permite reconstruir la forma 3D del cuerpo a partir de un sistema coordenado global. Un dispositivo de tratamiento digital estereoscópico permite determinar la posición absoluta de la cabeza óptica de reconstrucción. A parir de esta información la altura se determina con respecto a un sistema coordenado fijo, calculándose la forma 3D del cuerpo independientemente del desplazamiento masivo de la cabeza óptica. El dispositivo se implementó para medir la variación temporal de la superficie externa en modelos análogos cualitativos de volcanes de lodo causados por el impulso generado por gases al interior del volcán [2].

- **Patiño A., Miranda D., Meneses J. (2003) "Escáner 3D de objetos a 360° de observación".** Se diseñó y construyó un sistema para la reconstrucción topográfica de objetos 3D difusos, utilizando el principio de triangulación con iluminación estructurada. El sistema está formado por los siguientes módulos: iluminación, posicionamiento angular, captura y procesamiento de imágenes. La iluminación consiste en un plano láser, que al proyectarse sobre el objeto forma una línea. La proyección del plano láser sobre el cuerpo lo realiza un espejo sujeto a un brazo mecánico. El barrido de la línea láser sobre el cuerpo, se efectúa rotando el brazo en uno de sus extremos con la ayuda de un motor paso a paso. La reconstrucción es coordinada de manera automática por un software de control, el cual enlaza los demás módulos para obtener finalmente la representación 3D de la vista observada del objeto [3].

- **Blanes F., Jiménez M., Puerto R., Ñeco P., Reinoso O. (2007) "Reconstrucción tridimensional de escenas con un par estereoscópico de cámaras"** En el amplio campo de la visión artificial destaca por sí misma la parte de reconstrucción tridimensional por el simple hecho de recrear el proceso de visión "tridimensional" humano. Como bien es sabido, el proceso de reconstrucción tridimensional a partir de un par estereoscópico de cámaras consta de tres fases bien diferenciadas: calibración, correspondencia y reconstrucción. El presente artículo recoge aspectos fundamentales sobre el estudio, comparación e implementación de algoritmos de calibración, tanto euclídea como proyectiva, así como de reconstrucción, obviando la fase de correspondencia estereoscópica [4].

- **Cerca M., Barrientos B., García J., Hernández C. (2007) "Obtención del relieve digital mediante proyección de luz estructurada en modelos analógicos de extensión"** Se muestran resultados del uso de una técnica de proyección de luz estructurada para obtener un mapa digital del relieve durante la deformación por extensión en modelos analógicos que simulan la parte superior de la corteza terrestre. La técnica para obtener el relieve consiste en la proyección de un patrón de luz estructurada en franjas binarias, luz y sombra, sobre la superficie del modelo. El modelo es deformado progresivamente y se obtiene una fotografía digital de la superficie para cada incremento de deformación. El sistema de deformación es de tipo squeeze-box y consiste de una caja de acrílico dentro de la cual se construyen los modelos usando arena y silicón con diferentes diseños experimentales. Una pared vertical móvil dentro de la caja se desplaza a velocidad constante permitiendo la extensión del modelo. Los resultados obtenidos ilustran la utilidad de las técnicas ópticas para analizar la deformación superficial en los modelos físicos y representar los resultados de manera digital [5].

1.6.1.1 Inteligencia Artificial (IA)

Es definida como la inteligencia exhibida por una entidad artificial. Generalmente se asume que dicha entidad sea un sistema o un computador. La inteligencia artificial forma una rama vital en la ciencia de la computación, la cual lidia con el comportamiento inteligente, el aprendizaje y la adaptación en las máquinas [6].

La IA se ha extendido a muchas áreas que han creado ramas de investigación enormes y diferenciadas entre estas se nombran las siguientes:

- Aprendizaje Automático (Machine Learning).
- Ingeniería del conocimiento (Knowledge Engineering).
- Lógica difusa (Fuzzy Logic).
- Redes neuronales artificiales (Artificial Neural Networks).
- Sistemas reactivos (Reactive Systems).
- Sistemas multi-agente (Multi-Agent Systems).
- Sistemas basados en reglas (Rule-Based Systems).
- Razonamiento basado en casos (Case-Based Reasoning).
- Sistemas expertos (Expert Systems).
- Redes Bayesianas (Bayesian Networks).
- Vida artificial (Artificial Life). La VA no es un campo de la IA, sino que la IA es un campo de la VA.
- Computación evolutiva (Evolutionary Computation).
    - Estrategias evolutivas.
    - Algoritmos genéticos (Genetic Algorithms).
- Técnicas de Representación de Conocimiento.

    - Redes semánticas (Semantic Networks).
    - Frames.
- Visión artificial.
- Audición artificial.
- Lingüística computacional.
- Procesamiento del lenguaje natural (Natural Language Processing).
- Minería de datos (Data Mining).

1.6.1.2 Visión Artificial

También conocida como Visión por Computador (*del inglés Computer Vision*) o Visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen con el fin de extraer la información necesaria para controlar un proceso o una actividad como:

- control de calidad.
- Ordenación por calidades (grading).
- manipulación de materiales.
- prueba y calibración de aparatos.
- monitorización de procesos.
- reconstrucción 3D [7].

1.6.1.3 Reconstrucción 3D

La reconstrucción 3D es el proceso mediante el cual, objetos reales, son representados en la memoria de una computadora, manteniendo sus características

físicas (dimensiones, volumen y forma). Existen dentro de la visión artificial, multitud de técnicas de reconstrucción y métodos de mallado 3D, cuyo objetivo principal es obtener un algoritmo que sea capaz de realizar la conexión del conjunto de puntos representativos del objeto para forma elementos de superficie, tales como: triángulos, cuadrados o cualquier otra forma geométrica [8].

1.6.1.4 Escáner

El escáner es un dispositivo de entrada que permite digitalizar imágenes y documentos (escáner de ordenador o de barras) o encontrar un objeto o señal (escáner de un aeropuerto, o de radio) [9].

1.6.1.5 Escáner 3D

Un escáner 3D es un artefacto que analiza un objeto o el ambiente físico para reunir los datos de su forma y posiblemente color. Los datos completos entonces se pueden usar para construir modelos digitales tridimensionales que se usan en una amplia variedad de aplicaciones. Estos artefactos son usados extensamente por la industria en la producción de películas y videojuegos. Otras aplicaciones importantes incluyen el diseño y prototipos industriales, análisis estructural por computadora y la documentación de artefactos culturales [10].

1.6.1.6 Imagen

Una **imagen** (del latín *imago*) es una representación visual de un objeto mediante técnicas diferentes de diseño, pintura, fotografía, video, entre otras.

1.6.1.7 Píxel

Un píxel (acrónimo del inglés *picture element*, "elemento de imagen") es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea ésta una fotografía, un fotograma de video o un gráfico [11].

1.6.1.8 Imagen digital

La imagen digital es todo un conjunto de puntos (pixeles) de colores o no que integrados forman a veces algo coherente, puede ser de cualquier tamaño o resolución, puede ser gráfica o fotográfica y pude tener diversos tipos de formatos dependiendo de la calidad, de la compresión, etc.

1.6.1.9 Representación de las imágenes en los computadores digitales

Aunque el sistema de visión humano tiene mayor resolución en la fóvea[1] y menos en la periferia, se ha observado que a pesar de que la distribución de los fotorreceptores no es uniforme, la percepción visual si lo es. Los humanos percibimos con una única resolución. Estas circunstancias han conducido a la utilización de sensores con

1.- **Fóvea**: En todos los mamíferos, la fóvea es el área de la retina donde se enfocan los rayos luminosos y se encuentra especialmente capacitada para la visión aguda y detallada.

matrices de resolución uniforme. Por tanto, la organización corresponde a una matriz 2D uniforme.

Las imágenes para ser procesadas en el computador se adquieren a través de un dispositivo de captura de video. Esta señal es de carácter bidimensional y emplea variables discretas. El acceso a esta información elemental se hace indicando la fila y la columna que ocupa. El origen de coordenadas de la imagen se encuentra en la esquina superior izquierda. El eje horizontal corresponde con las columnas y el eje vertical con las filas. Se emplearán índices enteros para posicionar el píxel. Se denotará el valor del píxel a través de una función, del tipo f(x,y), siendo x el índice de la fila e y de la columna.

Si la imagen es acromática, sólo se presenta la luminancia, esto es, los niveles de grises. La función f(x,y) retornará el nivel de gris del píxel mencionado. En caso de que la imagen sea en color, f(x,y) devolverá un vector. Normalmente, suele expresarse como una proyección del color sobre el sistema RGB (Red-Green-Blue). En la figura 1 se muestra como se encuentran organizados los pixel en una imagen bidimensional.

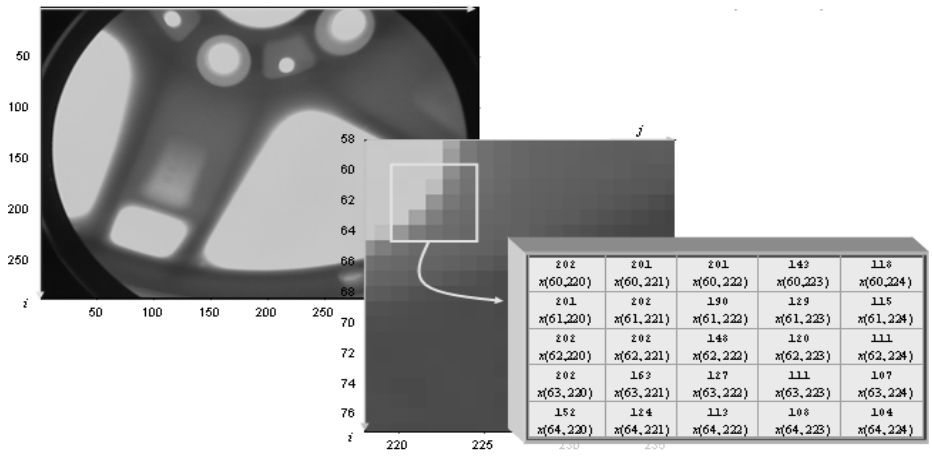


Figura 1.1: Organización matricial uniforme de una imagen digital.

Si la imagen es de tipo 3D, (por ejemplo, en resonancia magnética, luz estructurada, entre otras), ésta se presenta como una pila de imágenes 2D a la cual se les añade otro índice, denominado k o z, que indica el orden en que se encuentra de imagen 2D dentro de la pila.

La secuencia temporal de imágenes estáticas da lugar al video. En el cine se emplea 24 fotogramas por segundo, gracias a la persistencia retiniana en el ojo humano, da sensación de continuidad en la escena [12].

1.6.1.10 La luz como onda electromagnética

Algunos tipos de energía requieren de un medio conductor para propagarse. Como así sucede con la energía eléctrica o mecánica. Pero hay otros tipos de fuentes energéticas que no necesitan de un soporte conductor, éste el caso de la luz. Las radiaciones electromagnéticas se propagan en forma de dos ondas vectoriales mutuamente acopladas y perpendiculares entre sí; una onda para el campo eléctrico y otra para el campo magnético (figura 2a). Según la teoría ondulatoria, la luz se propaga en forma de onda que viaja en el vacío con una velocidad constante $c = 3\times10^8$ m/s. El espectro visible es una porción muy pequeña del conjunto de ondas electromagnéticas que tiene la peculiaridad de ser captada por los ojos y procesada en el cerebro. El ojo humano es capaz de distinguir radiaciones de longitudes de onda comprendidas entre los 380 nm a los 780 nm, cuyas frecuencias oscilan entre los 3.2×10^{14} Hz y los 7.7×10^{14} Hz1 (figura 2b). Al ser recibidas interpreta las diferentes amplitudes y frecuencias, produciendo sensaciones conocidas como brillo y color respectivamente [12].

a. b.

Figura 1.2: a) Campo electromagnético, b) Espectro de la luz.

1.6.1.11 Fuentes de luz

La distribución espectral de la energía radiada, λ, es una función que representa la cantidad de potencia asociada a cada longitud de onda. Si la distribución presenta un pico sobre una determinada longitud de onda y es despreciable el resto de componentes, se dice que es una radiación monocromática. Éste es el caso de la luz láser. La luz blanca se caracteriza por tener una distribución uniforme en su espectro.

Cada onda luminosa monocromática lleva asociada una energía, cuyo valor es igual a:

$$Q = h.f = \frac{h.c}{\lambda}$$

Donde *h* es la constante de Planck, igual a 6.63×10-34 J×s, *f* es la frecuencia, *c* la velocidad de la luz y λ la longitud de la onda. Así, la luz de menor frecuencia tiene menor contenido energético, mientras que la luz de menor longitud de onda posee mayor energía [12].

1.6.1.12 Luz estructurada

Los sistemas de luz estructurada se basan en estudiar la deformación que sufre un patrón de luz al ser interseccionado por cualquier objeto. Este es el problema principal de este tipo de herramientas, ya que se necesita un tipo de luz concentrada en un punto. No valdría como sistema de iluminación, cualquiera de los sistemas normales que se emplean actualmente, como bombillas, fluorescentes, entre otras, ya que, están compuestos por ondas de diferentes frecuencias provocando que el haz se difumine por todo el entorno.

Una de estas técnicas consiste en la proyección de una línea recta de luz sobre el objeto, obteniendo una imagen bidimensional que es adquirida por medio de una cámara digitalizadora colocada en un ángulo adecuado.

La información de profundidad se adquiere a partir del desplazamiento relativo de los diferentes puntos de la línea de luz. Una particularidad del método de línea recta es la necesidad de procesar múltiples imágenes para reconstruir el objeto completo. La totalidad del objeto puede cubrirse ya sea moviendo la fuente de luz, o dejando el sistema de iluminación fijo y darle movimiento al objeto. Una variación de este método de luz estructurada consiste en la proyección de múltiples líneas sobre la escena con una regularidad o un patrón definido. Lo anterior es logrado mediante el uso de una rejilla de difracción diseñada para el caso. La técnica de múltiples líneas tiene la ventaja obvia de requerir en principio una sola imagen para lograr la reconstrucción de la escena completa. Esta ventaja, en términos computacionales, representa una simplificación en el manejo de los “buffers” o memoria RAM. Sin embargo, la complejidad de los algoritmos de reconstrucción aumenta considerablemente pues es necesario resolver ciertas dificultades inherentes al método, entre las cuales destaca la posible confusión entre las diferentes líneas. El principio de extracción de profundidad

mediante el desplazamiento relativo de puntos de luz conduce al diseño de algoritmos demasiado complejos [13].

1.6.1.13 Ingeniería de Software

Según la definición del IEEE, "software es la suma total de los programas de computadora, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo". Según el mismo autor, "un producto de software es un producto diseñado para un usuario". En este contexto, “la Ingeniería de Software es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software", que en palabras más llanas, se considera que "la Ingeniería de Software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas (eficaces en costo o económicas) a los problemas de desarrollo de software", es decir, "permite elaborar consistentemente productos correctos, utilizables y costo-efectivos" .

1.6.1.14 El Proceso Unificado

El Proceso Unificado "es un proceso de desarrollo de software configurable que se adapta a través de los proyectos variados en tamaños y complejidad. Se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos. El Proceso Unificado guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos, para diseñar y probar el sistema hecho de acuerdo a los requerimientos. El proceso describe qué entregables[1] producir,

1.- Entregables: dentro del ámbito de los proyectos de TI (Tecnologías de la Información), es común utilizar la palabra, (por ej., un informe, un prototipo, etc.) que alguien debe entregar para mostrar el progreso que ha logrado en un proyecto.

cómo desarrollarlos y también provee patrones. El proceso unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

El Proceso Unificado ha adoptado un enfoque que se caracteriza por:

- Interacción con el usuario continúa desde un inicio
- Mitigación de riesgos antes de que ocurran
- Liberaciones frecuentes
- Aseguramiento de la calidad
- Involucramiento del equipo en todas las decisiones del proyecto
- Anticiparse al cambio de requerimientos

1.6.1.15 Lenguaje de Modelado Unificado UML

El Lenguaje de Modelado Unificado (UML-Unified Modeling Leguage) es un lenguaje grafico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocios y funciones de sistemas, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de bases de datos y componentes de software reusables.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Usos para modelar procesos “business”.

- Diagramas de Secuencias para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar iteraciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagrama de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

1.6.1.16 Programación orientada a objetos

Es un paradigma de programación en el cual un programa se ve como un conjunto de objetos discretos, los cuales a su vez son colecciones autocontenidas de estructuras de datos y rutinas que interactúan con otros objetos.

De esta forma, un objeto contiene toda la información, (atributos) que permiten definirlo e identificarlo frente a otros objetos pertenecientes a otras clases (e incluso entre objetos de una misma clase). A su vez, dispone de mecanismo de interacción (métodos) que favorecen la comunicación entre objetos, y en consecuencia, el cambio de estado entre los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan los métodos de los

atributos; esto difiere de los lenguajes imperativos tradicionales, en que los datos y los procedimientos están separados y sin relación, ya lo que único que se busca es el procesamiento de unos datos de entrada para obtener otro de salida.

1.6.1.17 Conceptos Básicos de la Programación Orientada a Objetos [14]

1.6.1.17.1 Clases

Una clase es un tipo definido por el usuario que determina las estructuras de datos y las operaciones asociadas con ese tipo. Cada vez que se construye un objeto de una clase, se crea una instancia de esa clase. En general, los términos objetos e instancias de una clase se pueden utilizar indistintamente.

1.6.1.17.2 Objeto

Un objeto es una unidad que contiene datos y las funciones que operan sobre esos datos. Los datos se denominan miembros dato y las funciones métodos o funciones miembro. Los datos y las funciones se encapsulan en una única entidad. Los datos están ocultos y sólo mediante las funciones miembro es posible acceder a ellos.

1.6.1.17.3 Atributo

Contenedor de un tipo de datos asociado a un objeto (o a una clase de objeto), que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser alterado por la ejecución de algún método.

1.6.1.17.4 Métodos

Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la resección de un "mensaje" des del punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, y/o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

1.6.1.18 Características de la Programación Orientada a Objetos [14]

1.6.1.18.1 Abstracción

Cada objeto en el sistema sirve como modelo de un agente abstracto que puede realizar trabajo, informar y cambiar su estado, y comunicarse con otros objetos en el sistema sin revelar como implementan estas características.

1.6.1.18.2 Encapsulamiento

Es el resultado de ocultar los detalles de implantación de un objeto con respecto a su usuario. Cada objeto está aislado del exterior, es un modulo natural, el aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas.

1.6.1.18.3 Polimorfismo

La referencia y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en un objeto producirá el comportamiento correcto para el tipo real del objeto referenciado.

1.6.1.18.4 Herencia

En orientación a objetos la herencia es el mecanismo fundamental para implementar la reutilización y extensibilidad del software. A través de ella los diseñadores pueden construir nuevas clases partiendo de una jerarquía de clases ya existente (comprobadas y verificadas) evitando con ello el rediseño, la remodificación y verificación de la parte ya implementada. La herencia facilita la creación de objetos a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a los ya existentes.

CAPITULO II
LA VISION ARTIFICIAL

2.1 VISION ARTIFICIAL O POR COMPUTADORA

El termino visión por computadora dentro del campo de la inteligencia artificial puede considerarse como el conjunto de todas aquellas técnicas y modelos que permitan el procesamiento, análisis y definición de cualquier tipo de información especial obtenida a través de imágenes digitales.

El propósito de un sistema de visión artificial es obtener de una imagen la información necesaria y útil para la ejecución de una tarea. En el caso mas simple, la información se refiere solamente a la posición y orientación de un objeto aislado; en otro caso se debe reconocer los objetos y determinar sus relaciones espaciales; esto puede observarse como un proceso en el que se desarrolla la descripción apropiada de la escena que se está viendo, a partir de una hilera de imágenes para una aplicación.

La visión artificial se ha construido, en parte, sobre ideas de tras camp relacionados: el procesamiento de imágenes, el reconocimiento de objetos y el análisis de escenas. Tiene que ver con la producción de nuevas imágenes a partir de las ya existentes, lo que interesa es la generación de aplicaciones simbólicas y el uso de estas descripciones para permitir al computador controlar el sistema tomando sus decisiones apropiadas.[15]

2.2 IMAGEN

Se refiera a una función bidimensional representando intensidad de luz *f (x,y)*, donde *x* e *y* son las coordenadas espaciales y el valor de *f* en cualquier punto (*x,y*) es proporcional al brillo (o nivel de gris) de la imagen en ese punto [16].

2.3 IMAGEN DIGITAL

La imagen digital está formada por un conjunto definido de puntos llamados pixeles, por lo tanto la imagen digital es todo conjunto de puntos (pixeles) de colores o no que integrados forman a veces algo coherente, puede ser de cualquier tamaño y resolución, puede ser grafica o fotográfica y puede tener diversos tipos de formatos dependiendo de la calidad de la compresión [16].

2.4 RESOLUCIÓN DE IMAGEN

La resolución de una imagen, x es la cantidad de píxeles que la componen. Suele medirse en píxeles por pulgada (ppi) o píxeles por centímetro (pcm). Cuanto mayor es la resolución de una imagen, más calidad tendrá su presentación, pero, más espacio ocupará en el disco el archivo gráfico que la contiene [16].

2.5 TIPOS DE IMÁGENES [16]

2.5.1 Imágenes vectoriales

Las imágenes vectoriales están compuestas por entidades geométricas simples: segmentos y polígonos (de hecho, una curva se reduce a una sucesión de segmentos).

Cada una de estas entidades está definida matemáticamente por un grupo de parámetros (coordenadas inicial y final, grosor y color del contorno, color del relleno, etc.); cualquier imagen, independiente mente de su complejidad, puede reducirse a una colección de entidades geométricas simples.

2.5.2 Imágenes en mapa de bits

Las imágenes en mapa de bits están construidas mediante una gran cantidad de cuadraditos, llamados pixel que componen a la imagen digital. Cada uno de estos cuadraditos está relleno de un color uniforme, pero la sensación obtenida es el resultado de integrar visualmente, en la retina, las variaciones de color y luminosidad entre píxeles vecinos.

Las imágenes en mapa de bits, también llamadas bitmap, son la alternativa ideal para reproducir objetos sutilmente iluminados y escenas con gran variación tonal.

2.6 COMPRESION DE ARCHIVOS [17]

Debido a que los archivos pueden ocupar mucho espacio especialmente los asociados con imágenes digitales, volviéndose inmanejables, se han desarrollado diferentes técnicas de compresión. Estas tratan de reducir mediante algoritmos matemáticos, el volumen del archivo, para así disminuir los recursos que consuma, y abreviar el tiempo de transferencia. Estos complejos algoritmos matemáticos reducen de múltiples maneras los 0 y 1 que conforman una imagen digital. La clasificación mas común es: compresión sin pérdida y compresión con pérdida. Hay que agregar que algunos formatos pueden utilizar varias de las diferentes técnicas para comprimir.

2.6.1 Compresión sin perdida

Esta técnica condensa las cadenas de código sin despreciar nada de información que forma la imagen, por lo que, esta se regenera intacta al ser descomprimida. Sin embargo, es menor la capacidad de compresión que provee este tipo de técnicas, dado que dado que su fin es presentar una exacta visualización de la imagen.

2.6.2 Compresión con perdida

Se denomina algoritmo de compresión con pérdida a cualquier procedimiento de codificación que tenga como objetivo representar cierta cantidad de información utilizando una menor cantidad de la misma, siendo imposible una reconstrucción exacta de los datos originales.

La compresión con pérdida sólo es útil cuando la reconstrucción exacta no es indispensable para que la información tenga sentido. La información reconstruida es solo una aproximación de la información original. Suele restringirse a información analógica que ha sido digitalizada (imágenes, audio, video, etc.), donde la información puede ser "parecida" y, al mismo tiempo, ser subjetivamente la misma. Su mayor ventaja reside en las altas razones de compresión que ofrece en contraposición a un algoritmo de compresión sin pérdida.

Existen dos técnicas comunes de compresión con pérdida:

- Por códecs de transformación: los datos originales son transformados de tal forma que se simplifican (sin posibilidad de regreso a los datos originales). Creando un nuevo conjunto de datos proclives a altas razones de compresión sin pérdida.

- Por códecs predictivos: los datos originales son analizados para predecir el comportamiento de los mismos. Después se compara esta predicción con la realidad, codificando el error y la información necesaria para la reconstrucción. Nuevamente, el error es proclive a altas razones de compresión sin pérdida.

2.6.3 Sistemas de compresión emergentes

Estos sistemas de compresión están diseñados para generar imágenes con múltiples resoluciones a partir de un único archivo fuente que, evidentemente, está limitado por la resolución real de la imagen archivada. Estos sistemas proporcionan una gran flexibilidad cuando se manipulan imágenes en la computadora, pero están muy ceñidos a círculos profesionales y muy especializados.

2.7 FORMATOS DE IMAGEN [16]

2.7.1 Formato BMP

Windows bitmap (.BMP) es el formato propio del programa Microsoft Paint, que provee con el sistema operativo Windows. Puede guardar imágenes de 24 bits (16,7 millones de colores), 8 bits (256 colores) y menos. Puede darse a estos archivos una compresión sin pérdida de calidad: la compresión RLE (Run-length encoding).

Los archivos con extensión .BMP, en los sistemas operativos Windows, representan la sigla BitMaP (o también Bit Mapped Picture), o sea mapa de bits. Los archivos de mapas de bits se componen de direcciones asociadas a códigos de color, uno para cada cuadro en una matriz de pixeles tal como se esquematizaría un dibujo

de "colorea los cuadros" para niños pequeños. Normalmente, se caracterizan por ser muy poco eficientes en su uso de espacio en disco, pero pueden mostrar un buen nivel de calidad. A diferencia de los gráficos vectoriales, al ser reescalados a un tamaño mayor, pierden calidad. Otra desventaja de los archivos BMP es que no son utilizables en páginas web debido a su gran tamaño en relación a su resolución.

Dependiendo de la profundidad de color que tenga la imagen cada pixel puede ocupar 1 o varios bytes. Generalmente se suelen transformar en otros formatos, como JPEG (fotografías), GIF o PNG (dibujos y esquemas), los cuales utilizan otros algoritmos para conseguir una mayor compresión (menor tamaño del archivo).

2.7.2 Formato JPG

Es un formato de compresión con pérdidas, pero que desecha en primer lugar la información no visible, por lo que las pérdidas apenas se notan.

El algoritmo JPG está basado en el hecho de que el ojo humano percibe peor los cambios de color que las variaciones de luminosidad. JPG divide la información de la imagen en dos partes: color y luminosidad y las comprime por separado.

Admite modos en escala de grises con una profundidad de 8 bits y en color hasta 24 bits. Permite la carga progresiva en un navegador, lo que lo ha convertido en el formato estándar en la web. No es un formato adecuado para imágenes con alto contraste de color.

Además, hay que tener en cuenta que la compresión se produce automáticamente cada vez que se guarda el archivo, por lo que es aconsejable guardar en este formato una única vez, cuando la imagen esté ya terminada.

2.7.3 Formato GIF

Es un formato que devuelve imágenes de tamaño muy reducido. Esa reducción se consigue indexando los colores, es decir, asimilándolos a uno de los 256 colores de su tabla. Su profundidad de color máxima, por tanto, es de 8 bits.

El formato GIF permite hacer algunas cosas curiosas: puede hacerse transparente uno de los colores indexados en la tabla, lo que permite suprimir fondos. También permite enlazar varias imágenes GIF en una secuencia, lo que se conoce con el nombre GIF animado.

El pequeño tamaño de los archivos GIF hizo que fuera el formato más extendido en los primeros tiempos de Internet. Pero su principal defecto consiste en que es un formato propietario (CompuServe Inc.), lo que ha provocado la aparición del formato libre PNG que, además, comprime mejor que GIF.

2.7.4 Formato PNG

Es el formato de más rápido crecimiento en la web, porque reúne lo mejor de JPG y GIF.

Se trata de un formato de compresión sin pérdidas, con una profundidad de color de 24 bits. Soporta hasta 256 niveles de transparencia, lo que permite fundir la imagen perfectamente con el fondo.

Entre sus inconvenientes hay que citar que no soporta animaciones y que el tamaño de los archivos PNG, debido a la capa de transparencia, siempre es mayor que el de los archivos JPG.

2.7.5 Formato TIFF

La denominación en inglés *"Tagged Image File Format"* es un formato de archivo de imágenes con etiquetas. Esto se debe a que los ficheros *TIFF* contienen, además de los datos de la imagen propiamente dicha, "etiquetas" en las que se archiva información sobre las características de la imagen, que sirve para su tratamiento posterior.

2.7.6 Otros formatos

Existen multitud de formatos: IFF, LBM, WPG, MAC, PIC, PSD, TGA, PCX, entre otros. Cada uno proviene de su peculiar programa de diseño con la ventaja de que prácticamente todos los programas de dibujo en el mercado soportan estas extensiones.

2.8 PROCESAMIENTO DIGITAL DE IMÁGENES

El procesamiento digital de imágenes, es un término usado para denominar las operaciones aplicadas sobre una imagen, para modificarlas de alguna forma u obtener información objetiva de la escena captada por una cámara, o cualquier otro dispositivo de captura.

En los últimos años, el procesamiento digital de imágenes ha sido ampliamente utilizado por diversas disciplinas tales como: Medicina, Bilogía, Física e Ingeniería.

Como aplicaciones típicas se puede mencionar: detección de la presencia de objetos, inspección visual automática, medición de características geométricas y de color, clasificación de objetos, restauración de imágenes, y mejoramiento de la calidad de las imágenes.

Entre las técnicas de procesamiento digital de imágenes podemos encontrar:

2.8.1 Filtros

Los filtros se utilizan para la modificación de imágenes ya sea para detectar los bordes de una escena o para modificar el aspecto, otra función de los filtros es, para la eliminación de ruido de la imagen.

2.8.2 Ecualización del Histograma

El histograma es un gráfico que relaciona los niveles de intensidad de una imagen y el número de píxeles que poseen tal nivel de intensidad

La ecualización del histograma se aplica cuando se desea lograr una distribución más uniforme entre el número de píxeles referido a los diferentes niveles de intensidad presentes en la imagen, ello es, un histograma que se extienda en el intervalo de niveles de gris de cero al nivel de gris máximo. La extensión de los

valores de niveles de gris se realiza utilizando información presente en la propia imagen.

El resultado de la ecualización maximiza el contraste de una imagen sin perder información de tipo estructural, esto es, conservando su Entropía (información).

2.8.3 Transformada de Hough

La Transformada de Hough, es un algoritmo empleado en reconocimiento de patrones, en imágenes que permite encontrar ciertas formas dentro de una imagen, como líneas, círculos, etc. La versión más simple consiste en encontrar líneas. Su modo de operación es principalmente estadístico y consiste en que para cada punto que se desea averiguar si es parte de una línea. Para ello se aplica una operación dentro de cierto rango, con lo que se averiguan las posibles líneas de las que puede ser parte el punto. Esto se continúa para todas los puntos en la imagen, al final se determina qué líneas fueron las que más puntos posibles tuvieron y esas son las líneas en la imagen.

2.8.4 Segmentación

La segmentación subdivide una imagen en sus partes constituyentes u objetos. El nivel al que se lleva a cabo esta subdivisión, depende del problema a resolver. Esto es, la segmentación deberá detenerse cuando los objetos de interés de una aplicación hayan sido aislados.

En general la segmentación autónoma es una de las tareas mas dificilles del procesamiento de imágenes. Esta etapa del proceso determina el eventual éxito o fracaso del análisis.

2.8.5 Binarización

La binarización de una imagen digital consiste en convertir la imagen digital en una imagen en blanco y negro, de tal manera que se preserven las propiedades esenciales de la imagen.

Uno de los métodos para poder binarizar una imagen digital es mediante el histograma de dicha imagen. A través del histograma obtenemos una gráfica donde se muestran el número de píxeles por cada nivel de gris que aparecen en la imagen. Para binarizar la imagen, se deberá elegir un valor adecuado dentro de los niveles de grises (umbral), de tal forma que el histograma forme un valle en ese nivel. Todos los niveles de grises menores al umbral calculado se convertirán en negro y todos los mayores en blanco.

(a) (b)

Figura 2.1: Binarización de imágenes: (a) Imagen en Color, (b) Imagen Binarizada.

2.8.6 Escala de grises

La escala de grises utiliza matices de negro para representar objetos. Los objetos en escala de grises tienen un valor de brillo comprendido entre el 0% (blanco) y el 100% (negro). Las imágenes producidas con escáner de blanco y negro o de escala de grises se visualizan normalmente en escala de grises.

La escala de grises también permite convertir ilustraciones en color en ilustraciones en blanco y negro de alta calidad. Los niveles de gris (matices) de los objetos convertidos representan la luminosidad de los originales.

Al convertir objetos de escala de grises a RGB, se asigna a los valores de color de cada objeto los valores de gris de los objetos originales. También se pueden convertir objetos de escala de grises en objetos CMYK.

a. b.

Figura 2.2: Imagen en escala de grises: a) Imagen en Color, b) Imagen a escala de gris.

2.8.7 Control del brillo de una imagen

En ciertas ocasiones, la apariencia de una imagen puede realzarse visualmente ajustando el brillo de la misma. Esto se logra sumando o restando un valor constante

a cada pixel de la imagen de entrada. El efecto de tal transformación sobre el histograma de la imagen, es desplazarlo hacia la derecha (zona más brillante), en caso de que se sume un valor constante o por el contrario, lo desplaza hacia la izquierda (zona más oscura) cuando se resta un valor constante.

2.9 ELEMENTOS DE UN SISTEMA DE PROCESAMIENTO DIGITAL DE IMÁGENES

Los componentes generales de un sistema de tratamiento digital de imágenes se pueden organizar de la siguiente forma:

- Digitalizador (unidad de entrada de datos)
- Procesador (unidad de proceso de datos)
- Pantalla (unidad de salida de datos)

2.9.1 Digitalizador

Es un dispositivo que convierte una imagen en una representación numérica (imagen digital). Existen multitud de aparatos para realizar esta función.

a. b. c.

Figura 2.3: Dispositivos digitalizadores: a) Cámara Digital, b) Cámara Web, c) Escáner.

2.9.2 Pantalla

La función de la pantalla es la inversa del digitalizador, esto es, convertir las matrices numéricas que representan imágenes en alguna forma de representación discernible por el ser humano a través de los sentidos.

a. b.

Figura 2.4: a) Monitor, b) Impresora.

2.10 RECONSTRUCCIÓN 3D

La reconstrucción 3D numérica es el proceso mediante el cual objetos son digitalizados en un sistema coordenado X, Y, Z, usando algún método de codificación de la topografía, manteniendo sus características físicas (dimensiones, volumen y forma). De esta manera un procedimiento de calibración o parametrización del sistema, convierte la secuencia de imágenes adquiridas y tratadas digitalmente, a escala real del objeto. Por esta razón los métodos de reconstrucción 3D son una herramienta útil en aplicaciones donde la información topográfica juega un papel importante en la toma de decisiones, como en el caso de control de calidad,

modelización 3D de objetos industriales y exploraciones funcionales de deformaciones en la espalda o en otra parte del cuerpo, entre otros.

Existen varios métodos de adquisición de objetos en 3D, pero en general dichos métodos se dividen en dos grandes grupos: Métodos de adquisición por contacto físico y sin contacto físico [18].

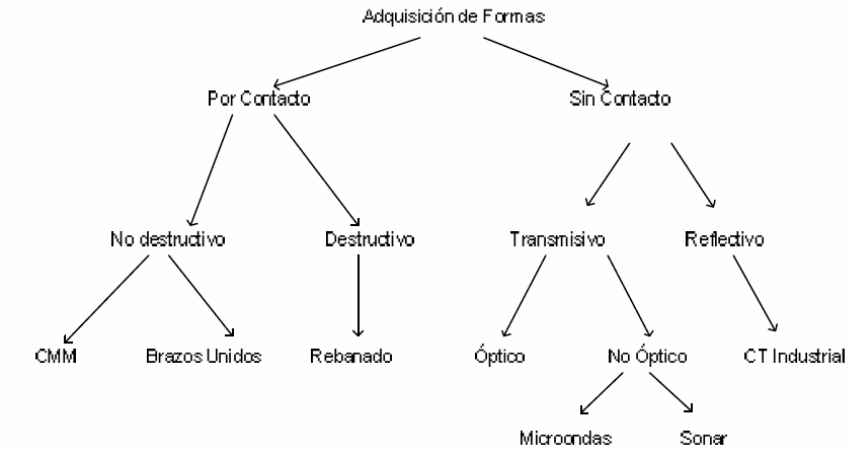


Figura 2.4: Métodos de adquisición de objetos en 3D.

2.11 TECNICAS DE ADQUISICION DE IMAGENES 3D [18]

El desarrollo de un sistema de Visión 3D requiere la resolución de una serie de aspectos ó etapas: recuperación de la estructura tridimensional de la escena, modelado y representación de objetos, reconocimiento y localización, y la interpretación de la escena.

Las diferentes técnicas existentes para la recuperación de la estructura tridimensional de la escena presentan características específicas en todos los niveles del proceso de interpretación visual, desde la etapa de formación de la imagen,

análisis e interpretación de la misma, cada método requiere tanto de equipos como de algoritmos específicos.

2.11.1 Por contacto físico

Los sistemas de adquisición de formas en 3-D por contacto físico son aquellos que ya sea por medios destructivos o no destructivos obtienen perfiles de un objeto. Esto quiere decir, que para obtener los datos de profundidad, el sistema tiene que tocar el objeto. Sistemas robóticos acercan estilógrafos al objeto o cortan al objeto en secciones para obtener varios planos en 2-D que al juntarlos generan información 3D.

Uno de los grandes problemas de los sistemas por contacto es el tiempo que utilizan para obtener todos los parámetros, ya que necesitan recorrer el cuerpo punto por punto. Además, al hacer contacto físico pueden alterar los parámetros de dicho objeto. Existen aplicaciones como la adquisición de formas de objetos históricos, en las cuales estos métodos por contacto serían inaceptables.

2.11.2 Visión Estéreo

El desarrollo de las técnicas de Visión Estéreo ha sido objeto de un gran esfuerzo de investigación en los últimos años. El objetivo de la Visión Estéreo es resolver dos problemas: el problema de correspondencia consistente en decidir para un punto del plano de imagen izquierdo, que punto en el plano de imagen derecho es correspondiente (son imágenes del mismo punto físico). El segundo problema es el problema de reconstrucción que trata de obtener, dados dos puntos correspondientes en ambos planos de imagen, las coordenadas 3D del punto en el espacio respecto a un sistema de coordenadas del mundo.

Dentro de este segundo problema subyace el problema de calibración del sistema visual que determina con qué aproximación conocemos las posiciones de los centros focales de cada sistema óptico, y la posición de los planos de imagen en el sistema de coordenadas del mundo. La solución del problema de la visión estéreo debe asumir desde el primer momento que las posiciones de los puntos en la imagen y los parámetros de calibración del sistema visual son conocidos de forma imperfecta. Por lo tanto no se puede confiar en modelos geométricos rígidos del proceso de formación y reconstrucción de la imagen.

2.11.3 Visión Activa

La mayoría de los sistemas de visión por computador consideran la Visión como un proceso reconstructivo que aborda el problema creando representaciones a diferentes niveles con un grado de abstracción progresivamente más alto. Según Marr (1982) el proceso de la visión consiste en descubrir mediante imágenes qué es lo que hay presente en una escena y dónde está ubicado. Hay que resaltar, que esta definición no capta el hecho de que la visión, habitualmente, es usada para guiar nuestras actividades, es decir, los procesos de adquisición y extracción de la información visual dependen en gran medida de la tarea a realizar, y sirven para interactuar con el entorno de una manera intencionada y activa: dirigimos la mirada y atendemos a eventos selectivamente en función de nuestras necesidades. El mayor problema que presentan los sistemas de visión artificial, surge de la enorme cantidad de datos contenidos en las imágenes convencionales muestreadas uniformemente, donde la resolución es constante en todo el campo visual; este volumen de datos hace prácticamente imposible que tales sistemas puedan procesar en tiempo real toda la información de dichas imágenes. Con el fin de acelerar y simplificar el proceso de comprensión de una escena, en los últimos años ha surgido una tendencia dentro del campo de la visión por computador denominada Visión Activa, que tiene como

objetivo el estudio y desarrollo de sistemas de percepción en tiempo real, que integrados en un robot permitan que este interaccione en un mundo complejo y dinámico. El paradigma de la visión activa toma como punto de partida de que la visión no se debe llevar a cabo de manera aislada, sino que debe servir para algún propósito y, en particular, debería permitir a la gente conocer el entorno. Los precursores de este paradigma fueron Bajcsy (1988), Aloimonos (1990) y Ballard (1992), que propusieron, inspirándose en los sistemas biológicos, actuar de forma activa en el proceso de percepción, o sea, controlar los parámetros de la cámara y los módulos de procesamiento, para simplificar la comprensión de las escenas. La visión activa es definida por Bajcsy como, un proceso inteligente de adquisición de información, consistente en elaborar estrategias de control para ajustar los parámetros del sensor de cara a mejorar el conocimiento que se tiene del entorno.

2.11.4 Técnicas de Luz Estructurada

Dentro del campo de la Visión tridimensional, existen un gran número de técnicas que hoy son empleadas con éxito en numerosas aplicaciones industriales. Entre todas ellas, se encuentra lo que se conoce como la Luz Estructurada.

Este tipo de sistema se caracteriza por ser un método directo y activo. Un método directo se caracteriza por que se pueden obtener conclusiones estudiando los datos obtenidos directamente de las imágenes. En cuanto a lo de activo, hay que tener en cuenta que este sistema emplea algún sistema generador de luz, por lo que introduce un tipo de energía al entorno donde se realiza el estudio.

Los sistemas de luz estructurada se basan en estudiar la deformación que sufre un patrón de luz al ser interceptado por cualquier objeto. Este es el problema principal de este tipo de herramientas, ya que se necesita un tipo de luz concentrada en un

punto. No valdría como sistema de iluminación, cualquiera de los sistemas normales que se emplean actualmente, como bombillas, fluorescentes, etc., ya que, están compuestos por ondas de diferentes frecuencias provocando que el haz se difumine por todo el entorno.

Una de las mejores soluciones es emplear un haz láser. Debido a sus características de coherencia, divergencia, y direccionabilidad, ésta se comporta en una luz ideal para este tipo de sistemas. Esto es así debido a que la coherencia provoca que todas las ondas del haz tengan la misma frecuencia, y junto a su poca divergencia y alta direccionabilidad permiten que se pueda dirigir un haz láser a cualquier punto que se quiera.

Una vez que ya se conoce el tipo de luz que se va a emplear, será necesario elegir un patrón adecuado. Las diferentes soluciones van desde la utilización de puntos, hasta rejillas con láseres de diferentes colores.

El emplear puntos implica tener que recorrer el objeto por toda su superficie tomando una gran cantidad de puntos y pudiendo perder algunas zonas de éste.

El empleo de un plano parece una mejor solución que la anterior. El plano iluminará un conjunto de puntos con las mismas características y que cumplen con la condición de un plano en el espacio. El uso de una rejilla con diferentes puntos de colores implica el tener toda la superficie del objeto iluminada de una vez y el problema consistiría en encontrar los diferentes puntos y su situación.

Por supuesto, además del patrón de luz, es necesario tener una cámara que recoja todas las imágenes de la deformación del plano láser. La posición de la cámara en el conjunto deberá ser aquella, que permita obtener, tanto la mejor resolución

como evitar que existan zonas oscuras, es decir, que no existan zonas del objeto que no sean iluminadas por el láser.

Una vez que se ha decidido por uno u otro patrón y la correspondiente cámara, será necesario calibrar los diferentes parámetros. Este paso es el más importante, ya que para obtener las coordenadas de diferentes puntos en el objeto se va a emplear el método de triangulación.

La triangulación consiste en obtener la situación de un punto en relación a la situación de la cámara y el plano. La figura de abajo explica el método de la triangulación:

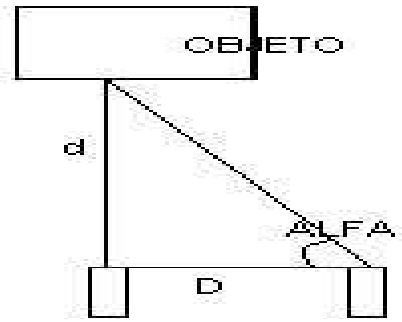


Figura 2.5: Esquema de la triangulación 3D.

Si se conoce la distancia de la cámara a un punto del objeto, que será la base del triángulo, la distancia entre la cámara y el láser, uno de los lados, y el ángulo del plano del láser, se puede conocer las coordenadas tridimensionales de dicho punto.

Pero para ello, es necesario conocer las posiciones en el espacio tanto de la cámara como del plano láser, por lo que será necesario, un proceso de calibración de ambos sistemas.

Los patrones proyectados están diseñados de manera que a cada píxel de la imagen se le pueda asignar un código bien definido. Las técnicas de proyección de estos patrones pueden clasificarse en función de la estrategia de codificación utilizada, dividiéndose en tres grandes grupos:

- Técnica de Multiplexado en tiempo.
- Técnicas de Vecindad espacial.
- Codificación directa.

En nuestro caso consideraremos únicamente las llamadas técnicas de multiplexado en tiempo una de las más extendidas en los sistemas comerciales de digitalización tridimensional por luz estructurada.

Dentro de las técnicas de multiplexado temporal podemos encontrar:

2.11.4.1 Técnicas binarias

En este caso, se utilizan únicamente dos niveles de iluminación, codificados por 0 ó 1, cada píxel del patrón tiene su propia codificación compuesta por una secuencia de 0s u 1s. Este tipo de patrones son sencillos de codificar y de procesar.

2.11.4.2 Técnicas de N-ary códigos

Estas técnicas reducen el número de patrones a proyectar sobre la pieza, sin embargo, utilizan un mayor número de niveles de iluminación (niveles en escala de gris/color incrementando así el nivel de información codificada en estos patrones.

2.11.4.3 Técnicas de codificación en grises y desplazamiento de fase.

La técnica consiste en la proyección de un conjunto de patrones binarios que producen una división del objeto en diferentes regiones. Además de este conjunto de patrones, se proyecta un patrón complementario que se desplaza con el fin de aumentar la resolución (Fig. 1.3).

2.11.5 Telemetría Láser

La telemetría láser consiste en medir el tiempo de recorrido de un rayo luminoso (láser) hasta la superficie de medida. Se puede medir de dos formas: con la medida del tiempo de vuelo y el cálculo por diferencia de fase. En el primer caso los datos se obtienen midiendo el tiempo entre la emisión del impulso luminoso y la observación del retorno. En el segundo se regula el impulso luminoso siguiendo una frecuencia determinada y se mide el desfase entre el rayo emitido y la luz retornada.

2.11.6 Holografía

La holografía es una técnica especial de producción de fotografías tridimensionales de un objeto. El termino holograma fue acuñado por el inventor de la holografía, el científico húngaro Dennis Gabor, a partir de las palabras “grama” (mensaje), y “halos” (toda, completa). En realidad un holograma contiene más información sobre la forma de un objeto que una fotografía simple, ya que permite verla en relieve, y variando la posición del observador, obtener diferentes perspectivas del objeto holografiado.

2.12 CALIBRACIÓN DE CÁMARAS [19]

El proceso de calibración de una cámara es el primer paso para resolver aplicaciones en las que sea necesario obtener datos cuantitativos de la imagen. Aunque es posible obtener información de la escena a partir de imágenes tomadas con cámaras sin calibrar, el proceso de calibración resulta esencial cuando se trata de obtener medidas de las mismas. La calibración precisa de la cámara permite obtener distancias en el mundo real a partir de las imágenes tomadas del mismo. Con esta información es posible resolver aplicaciones industriales de ensamblado de piezas, evitar obstáculos en la navegación de un robot, controlar un brazo robot o realizar una planificación de trayectorias. Si por el contrario nos centramos en la reconstrucción 3D de objetos, cada punto en la imagen determina un rayo óptico que pasa a través del centro óptico de cámara hacia la escena. El manejo de varias imágenes de una misma escena en la cual no exista movimiento, permite relacionar ambos rayos ópticos para obtener la posición de los puntos 3D en la escena. En este caso es necesario resolver el paso previo de las correspondencias de un objeto en las diferentes imágenes. Una vez se ha podido realizar la reconstrucción 3D del objeto, éste se puede comparar con un modelo almacenado para determinar las imperfecciones del mismo fruto del proceso de fabricación, lo que implica una mejoría importante respecto a la inspección humana.

La calibración de la cámara consiste en la estimación de los parámetros intrínsecos de la misma los cuales modelan la geometría interna de la cámara y las características ópticas del sensor. Los parámetros extrínsecos miden la posición y la orientación de la cámara respecto al sistema de coordenadas establecido para la escena. Estos dan la relación respecto al sistema de coordenadas del usuario en lugar del sistema de coordenadas de la cámara.

Existen diversas técnicas de calibración de cámara. De acuerdo a la literatura estas pueden ser clasificadas en dos grandes categorías: Calibración fotogramétrica y autocalibración.

2.12.1 Calibración fotogramétrica

Se realiza mediante la observación de patrones cuya geometría en el espacio 3D es conocida con un buen nivel de precisión. Los patrones de calibración normalmente están posicionados en dos o tres planos ortogonales entre ellos. En algunos casos, basta con un único plano, cuya traslación es perfectamente conocida. Este tipo de calibración requiere una configuración elaborada, pero sus resultados son eficientes

2.12.2 Autocalibración

Este método se basa en el movimiento de la cámara observando una escena estática, a partir de sus desplazamientos y usando únicamente la información de la imagen. La rigidez de la escena impone en general restricciones sobre los parámetros de cámara. Tres imágenes tomadas por una misma cámara con parámetros intrínsecos fijos son suficientes para obtener tanto los parámetros extrínsecos como intrínsecos. Aunque esta técnica es muy flexible, aun no se encuentra madura.

2.12.3 Modelo pin-hole

El modelo de cámara que tradicionalmente se utiliza para pasar de coordenadas reales 3D a coordenadas 2D pertenecientes a la imagen captada, suele ser el modelo de proyección perspectiva denominado modelo *pin-hole*. En dicho modelo todos los rayos provenientes de un cierto objeto atraviesan un fino agujero para impactar en el sensor imagen. Dado que las lentes no tienen este comportamiento lineal, el modelo

pin-hole debe ser corregido con un valor de distorsión, es decir, debe ser complementado con parámetros que corrigen su comportamiento ideal y lo acercan, lo más posible, al comportamiento real del objetivo.

El sistema de referencia de la cámara se sitúa en el centro de la proyección, coincidiendo el eje *z* de dicho sistema con el eje óptico, también denominado eje axial. En esta disposición de ejes, el plano imagen, de coordenadas *u*,*v*, se encuentra situado a una distancia igual a la longitud focal del objetivo de forma perpendicular al eje óptico. La intersección del eje óptico con el plano imagen se denomina punto principal.

El centro de proyección *C* de la cámara se supone constante pero es a priori desconocido. El plano imagen normalmente se sitúa por delante del centro de proyección *C* para tener una imagen sin inversión. En la Figura A-2 se muestra un esquema explicativo del modelo *pin-hole*.

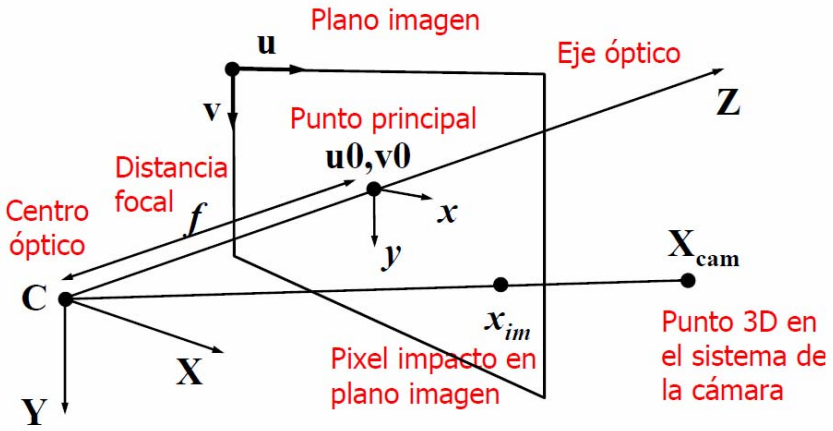


Figura 2.6: Diagrama explicativo del modelo pin-hole.

2.12.4 Parámetros de calibración: intrínsecos y extrínsecos

El modelo *pin-hole* posee dos grupos de parámetros diferenciados: parámetros de calibración intrínsecos y extrínsecos.

La matriz RT_{ext} (matriz de parámetros extrínsecos) transforma las coordenadas de un objeto de la escena dadas en el sistema de referencia externo a coordenadas en el sistema de referencia de la cámara. Dicha matriz tiene 6 parámetros independientes y se denominan extrínsecos, 3 para la matriz de rotación y otros 3 para el vector de traslación. Si se modifica la posición relativa entre un objeto de la escena y la cámara, la matriz extrínseca se ve modificada.

El resto de parámetros son inherentes a la óptica y al ajuste concreto en una cámara, por lo que se denominan parámetros intrínsecos Los parámetros intrínsecos son constantes en tanto no varíen las características y posiciones relativas entre la óptica y el sensor imagen.

Dentro del modelo *pin-hole*, los parámetros "distancia focal" y "factor de escala" siempre tienen la misma interpretación. No ocurre así con la corrección de la distorsión y la posición elegida para el punto principal (*u0*,*v0*) del modelo para el cual existen numerosas terminologías e interpretaciones. Se puede ver un análisis exhaustivo llegando a obtener 13 interpretaciones distintas para el punto principal: centro de distorsión radial, punto medio del sensor imagen, foco de expansión, etc. Todas estas interpretaciones del punto principal pueden llegar a ser válidas a la hora de obtener un modelo de formación de las imágenes al modificar convenientemente los términos de corrección de distorsión.

2.12.5 Aberraciones y distorsiones ópticas

Cuando no se utiliza una cámara oscura con óptica *pin-hole,* sino que se utiliza una óptica con lentes, la imagen obtenida diferirá de la que se hubiera obtenido con una *pin-hole* de distancia focal equivalente. Podemos dividir estas diferencias en varios tipos: las diferencias de enfoque o simplemente aberraciones, y las puramente geométricas o distorsiones; además de un caso particular de aberración conocido como aberración cromática.

En teoría, los rayos de luz procedentes de un punto de la escena deberían incidir sobre un único punto del plano de formación de la imagen. Cuando esto no ocurre, tenemos aberraciones. Los casos más clásicos son la aberración esférica y la coma. El término aberración esférica suele emplearse incorrectamente para referimos a lo que realmente se llama distorsión radial. La aberración esférica ocurre cuando la luz procedente de un punto se proyecta sobre un área circular del sensor, en lugar de un único punto. Este problema se da cuando tenemos mal enfocada la cámara, y se correspondería a la alteración que sufre un miope. La coma es una aberración similar, en la que la luz se proyecta sobre una superficie en forma de cola de cometa (de ahí su nombre) creando un efecto de halo.

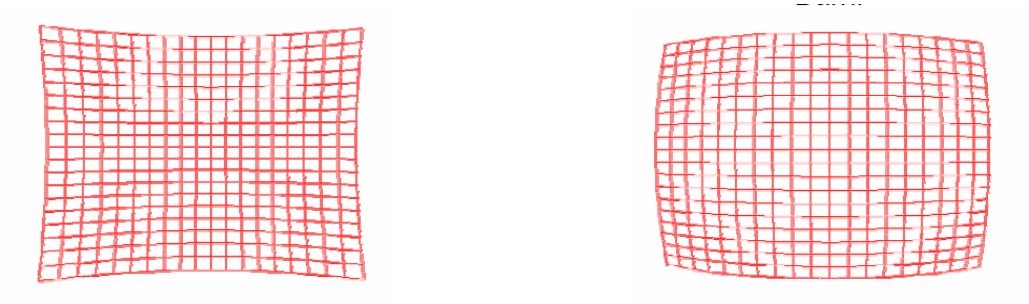
Tratar con las aberraciones suele dejarse en manos de las ópticas. Otro punto a tratar son las distorsiones geométricas

Se pueden considerar dos tipos de distorsiones, presentes en mayor o menor grado en las ópticas estándar más habituales. Estas distorsiones son las llamadas distorsión radial y distorsión tangencial, así como la combinación de las dos, conocida como distorsión de descentrado. Muchos autores ignoran la distorsión tangencial, basándose en que empíricamente se comprueba que suele ser mucho más

importante el efecto de la distorsión radial, y prácticamente despreciable el de la tangencial.

2.12.5.1 La distorsión radial

Está principalmente causada por la curvatura de las lentes. Bajo esta distorsión, los puntos de la imagen se desplazan hacia el interior (distorsión negativa) o hacia el exterior (distorsión positiva) desde su situación teórica, es decir, desde la situación en que se encontrarían si la cámara se ajustara perfectamente al *modelo pin-hole.*



a. b.

Figura 2.7: a) Distorsión negativa, b) Distorsión positiva.

2.12.5.2 La distorsión tangencial

A menudo es muy poco significativa, está principalmente provocada por defectos en el montaje de la óptica, como en la ya mencionada distorsión de descentrado, o porque el eje óptico no es perpendicular al plano sensor. La distorsión tangencial deformará la imagen en dirección perpendicular al llamado eje de máxima distorsión,

siendo la magnitud de este desplazamiento función de la distancia hasta el centro óptico de la imagen.

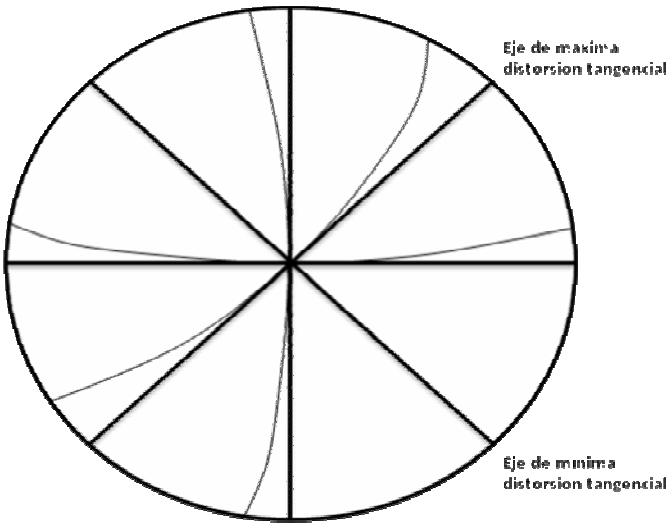


Figura 2.8: Efecto de la distorsión tangencial. Las líneas finas son la imagen distorsionada de las líneas gruesas ideales.

CAPITULO III
FASE DE INICIO

3.1 INTRODUCCION

A través del proceso unificado se logra desarrollar ordenada y metódicamente sistemas bien elaborados y que se ajustan a las necesidades del usuario final ya que su propósito está orientado a guiar el desarrollo en la implementación y distribución eficiente de dicho sistema.

El proceso unificado se apoya en una serie de ciclos que se repiten a lo largo del desarrollo del sistema, estos ciclos representan la vida del sistema y cada uno de ellos está constituido por cuatro fases que ayudan al desarrollo eficiente de la aplicación y cada una de ellas se subdividen en iteraciones que representan partes del sistema.

Las fases con las que cuenta este proceso son:

- Inicio.
- Elaboración.
- Construcción.
- Transición.

Las iteraciones hacen referencia a pasos de flujos de trabajos fundamentales: requisitos, análisis, diseño, implementación y prueba como se muestra en la figura. 3.1.

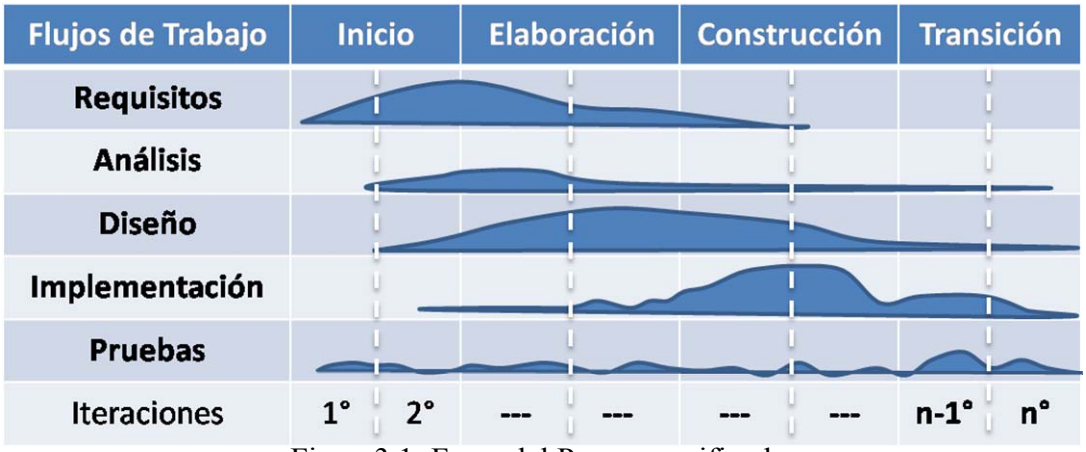


Figura3.1: Fases del Proceso unificado.

El objetivo de este capítulo es representar la fase de inicio aplicada al desarrollo de un software de reconstrucción 3D que utiliza técnicas de visión artificial, a través de tres flujos de trabajo típicos de esta fase: flujo de requisitos, flujo de análisis y flujo de diseño. Durante el flujo de requisitos de esta primera fase, se llevara a cabo el diagrama de modelo de dominio que servirá para comprender el contexto del sistema y los elementos que forman parte de él y el diagrama de casos de uso que nos dará una visión general de su funcionamiento.

En el flujo de análisis se llevara a cabo el análisis de los requisitos para estructurarlos y refinarlos en un modelo de objeto que sirve como primera iteración para el modelo de diseño.

Durante la etapa de diseño de la fase de inicio, se bosqueja un modelo inicial de diseño, el cual se lleva a cabo con los requisitos del sistema representados a través de los casos de uso.

3.2 CONTEXTO DEL SISTEMA

Es muy importante dar a conocer a los posibles usuarios finales el contexto del sistema y eso se logra dando una descripción del alcance de la aplicación. A través del modelo de dominio se pueden representar los aspectos más relevantes del entorno del sistema representándolos como objetos del mundo real.

El modelo de dominio da una visión del contexto del sistema de modo que se puede ver con claridad cómo está estructurado el sistema y lo que ocurre en el.

En la figura 3.2 se muestra el modelo de dominio, que representa el contexto del sistema, se observa como el usuario inicializa el sistema, este activa el proyector enviándole una secuencia de patrones que son capturados por la cámara, la cual le proporciona al software las imágenes necesarias para que las procésese mediante algoritmos de visión artificial y reconstrucción 3D y así genere un archivo el cual contendrá la información de las coordenadas de los puntos del objeto reconstruido llamado archivo de nube de puntos.

Los patrones de calibración se utilizan para determinar los parámetros intrínsecos de la cámara que se esté utilizando, en este proceso el usuario activa el proceso de calibración de cámara y este a su ves activa la cámara para que capture las imágenes de la plantilla de calibración colocada frente a la cámara en diferentes posiciones. Las imágenes capturadas el modulo de calibración las procesa para calcular las distorsiones en la lente de la cámara y así generar el archivo de parámetros intrínsecos necesarios para así corregir en las imágenes del objeto a ser reconstruido y eliminar error de distorsión introducido por la lente.

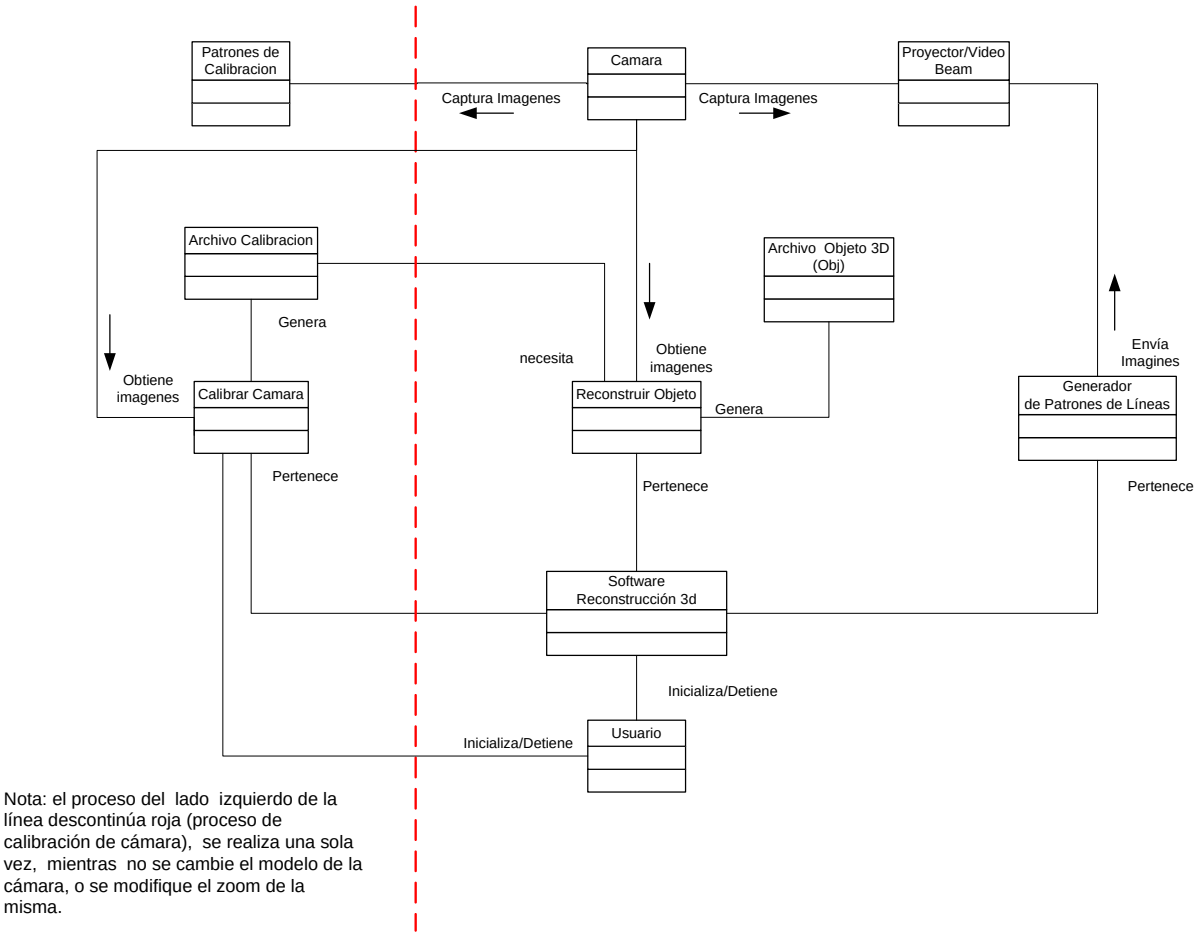


Figura 3.2: Modelo de Dominio del sistema.

Fuente: Los Autores.

3.3 RIESGOS CRITICOS DEL SISTEMA

Cuando se de desarrollar cualquier aplicación surgen situaciones que entorpecen el desarrollo del sistema y pueden ocasionar fallas que hagan que el sistema no cumpla con el propósito por el cual fue creado.

En esta fase de inicio se estudian los casos que tiene mayor complejidad y que lleva inherente un alto factor de riesgo que puede afectar el éxito del proyecto.

Lista de riesgos del sistema.

- No contar con algoritmos necesarios para la calibración de la cámara ya que estos son necesarios para corregir los errores introducidos por la lente.
- No contar con las ecuaciones para el cálculo de las coordenadas (x, y, z), de los puntos del objeto a reconstruir.
- No contar con la iluminación apropiada. La iluminación es un factor fundamental a la hora de la captura de las imágenes, por tal razón se debe contar con una iluminación apropiada para que las propiedades de las imágenes no se vean afectada y poder tener una buena imagen para su posterior procesamiento.
- No contar con el archivo de parámetros intrínsecos de la cámara ya que este es fundamental a la hora de realizarse la reconstrucción 3D del objeto.

3.4 ITERACION I

Se aplicó la fase de inicio, al sistema de reconstrucción 3D mediante la primera iteración que incluye, los flujos de trabajo, requisitos, análisis y diseño del software, con esto se obtiene una primera arquitectura del software.

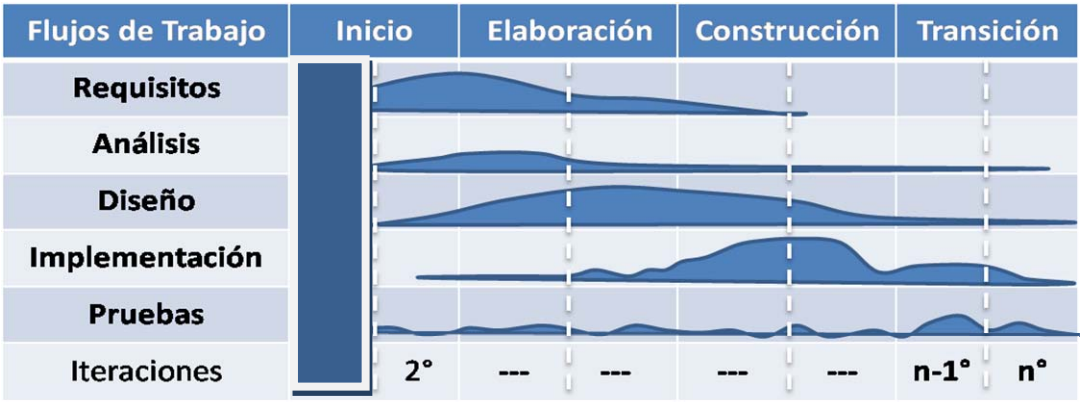


Figura 3.3: Iteración I (Fases del Proceso Unificado).

3.4.1 Requisitos

En esta parte del capítulo presentaremos los requisitos del sistema de reconstrucción 3D.

- Capturar imágenes de buena calidad que sirvan de entrada al sistema.
- Contar con algoritmos eficientes para el tratamiento de imágenes.
- Calibración de los parámetros intrínsecos de la cámara.
- Introducción de los parámetros físicos del sistema.
- Calibración del campo de visión de la cámara.
- Generar la matriz de proyección del plano.
- Generar la matriz de proyección del objeto.
- Generar la matriz de corrimiento a partir de la matriz de proyección del plano y la matriz de proyección del objeto.
- Contar con las ecuaciones matemáticas necesarias para la reconstrucción 3D.
- Generar la nube de puntos del objeto (archivo en formato *.obj).

3.4.1.1 Representación de los requisitos como casos de usos

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Un caso de uso representa los requisitos funcionales del sistema ya que definen su función y la acción que este realiza.

Para representar los requisitos como casos de uso se realizan las siguientes actividades:

- Determinar y describir los actores
- Determinar y describir los casos de uso.
- Describir el modelo de casos de uso.

3.4.1.2 Actores de del sistema

Un actor se puede definir como el rol que asume una persona o sistema que interactúa con el sistema que se está desarrollando. Tiene la propiedad de ser externo al sistema y no necesariamente representa a una persona en particular si no mas bien la labor que realiza él.

En el sistema se encuentran presentes los siguientes actores:

- Usuario: persona que interactúa con el sistema.
- Cámara: dispositivo que captura las imágenes que serán usadas por el sistema.
- Proyector/Video Beam: dispositivo que emite parones de luz enviados por el sistema, para ser capturados por la cámara.
- Manejador de archivos: almacena y/o consulta los archivos de configuración, de imágenes y de nube de puntos.

3.4.1.3 Casos de uso del sistema

Un caso de uso es una descripción de la secuencia de iteraciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica y proporciona uno o más escenarios que indica como debería interactuar el sistema con el usuario o con otras sistemas para conseguir un objetivo.
En el sistema se encuentran los siguientes casos de uso:

Tabla 3.1: Casos de uso del Sistema (1/2).

Caso de Uso: Configurar Sistema.	
Descripción:	Se encarga de recibir las variables externas del sistema por medio del usuario, verificar la validez de los datos introducidos, y realizar cálculos inherentes al sistema.
Actor(es):	Usuario, Manejador de archivos.
Objetivos:	Generar el archivo de configuración del sistema.
Caso de Uso: Calibrar Cámara	
Descripción:	Se encarga de procesar las imágenes de la plantilla de calibración
Actor(es):	Usuario, Manejador de archivos.
Objetivos:	Calcular los parámetros intrínsecos de la cámara y generar el archivo de calibración
Caso de Uso: Capturar Imagen	
Descripción:	Este caso de uso se encarga de capturar las imágenes de los patrones de código gray y de la plantilla de calibración a través del actor cámara.
Actor(es):	Cámara, Manejador de archivos
Objetivos:	Captura imágenes del ambiente del sistema.
Caso de Uso: Generar Nube de Puntos	
Descripción:	Se encarga de procesar la matriz de proyección y calcular las coordenadas de los puntos del objeto en reconstrucción.
Actor(es):	Usuario, Manejador de Archivos
Objetivos:	Generar el archivo de nube de puntos (archivo *.OBJ)
Caso de Uso: Proyectar patrón Código Gray	
Descripción:	Este caso de uso se encarga de generar los patrones de luz que serán proyectados sobre el plano y el objeto y entregarlos al actor Video Beam
Actor(es):	Video Beam
Objetivos:	Generar patrones de líneas en código gray.

Tabla 3.1: Casos de Uso (2/2).

Caso de Uso: Generar Matriz de Proyección	
Descripción:	Se encarga de procesar las imágenes de patrones de código gray tanto del plano como del objeto.
Actor(es):	Ninguno.
Objetivos:	Generar la matriz de proyección del plano y la matriz de proyección del objeto.
Caso de Uso: Abrir Archivo	
Descripción:	Se encarga de proporcionar una interfaz para que el usuario pueda localizar el archivo de nube de puntos, una vez encontrado entregarle la ruta del archivo al sistema para q pueda ser procesado.
Actor(es):	Usuario, Manejador de archivos.
Objetivos:	Mostrar el archivo de nube de puntos.
Caso de Uso: Guardar Archivo	
Descripción:	Se encarga de proporcionar una interfaz para q el usuario pueda proporcionarle un nombre y una dirección al archivo de nube de puntos para luego ser almacenarlo.
Actor(es):	Usuario, Manejador de archivos.
Objetivos:	Guardar el archivo de nube de puntos.

Fuente: Los Autores.

3.4.1.4 Modelo del caso uso

Un modelo de caso de uso muestra distintas operaciones que se esperan de una aplicación o sistema y como se relaciona con su entorno (usuario u otras aplicaciones).

Un modelo de caso de uso es una descripción del sistema desde el punto de vista del usuario. Para los desarrolladores de sistema, esta es una herramienta valiosa,

ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista del usuario.

En la figura 3.4 se muestra el modelo del caso de uso del sistema, como se puede observar el usuario activa el caso de uso Calibrar Cámara para calcular los parámetros intrínsecos de la cámara, este incluye al caso de uso Capturar imagen el cual se encarga de capturar las imágenes de donde se obtendrán dichos parámetros por medio del actor Cámara para luego interactuar con el actor Manejador de Archivos para almacenar un archivo de texto plano los datos obtenidos.

Luego de calibrada la cámara, el usuario activa el caso de uso Configurar Sistema en donde se introducen los datos externos de sistema y este interactúa con el actor Manejador de Archivos para almacenar un archivo con los datos introducidos.
Después de realizada la calibración del sistema el usuario activa el caso de uso Generar Nube de Puntos quien calcula y genera el archivo de nube de puntos en formato .OBJ. Este a su vez interactúa con los casos de uso Proyectar patrones Código Gray y Generar Matriz de Proyección. Proyectar patrones Código Gray es el encargado de generar los patrones de líneas que serán proyectados en el plano de proyección por medio del actor Video Beam y, de activar el caso de uso capturar imagen para almacenar los patrones de líneas generados, Generar Matriz de Proyección se encarga de procesar las imágenes anteriormente y entregarle sus resultados al caso de uso que lo llamo para que este se encargue de ejecutar el resto de sus acciones correspondientes.

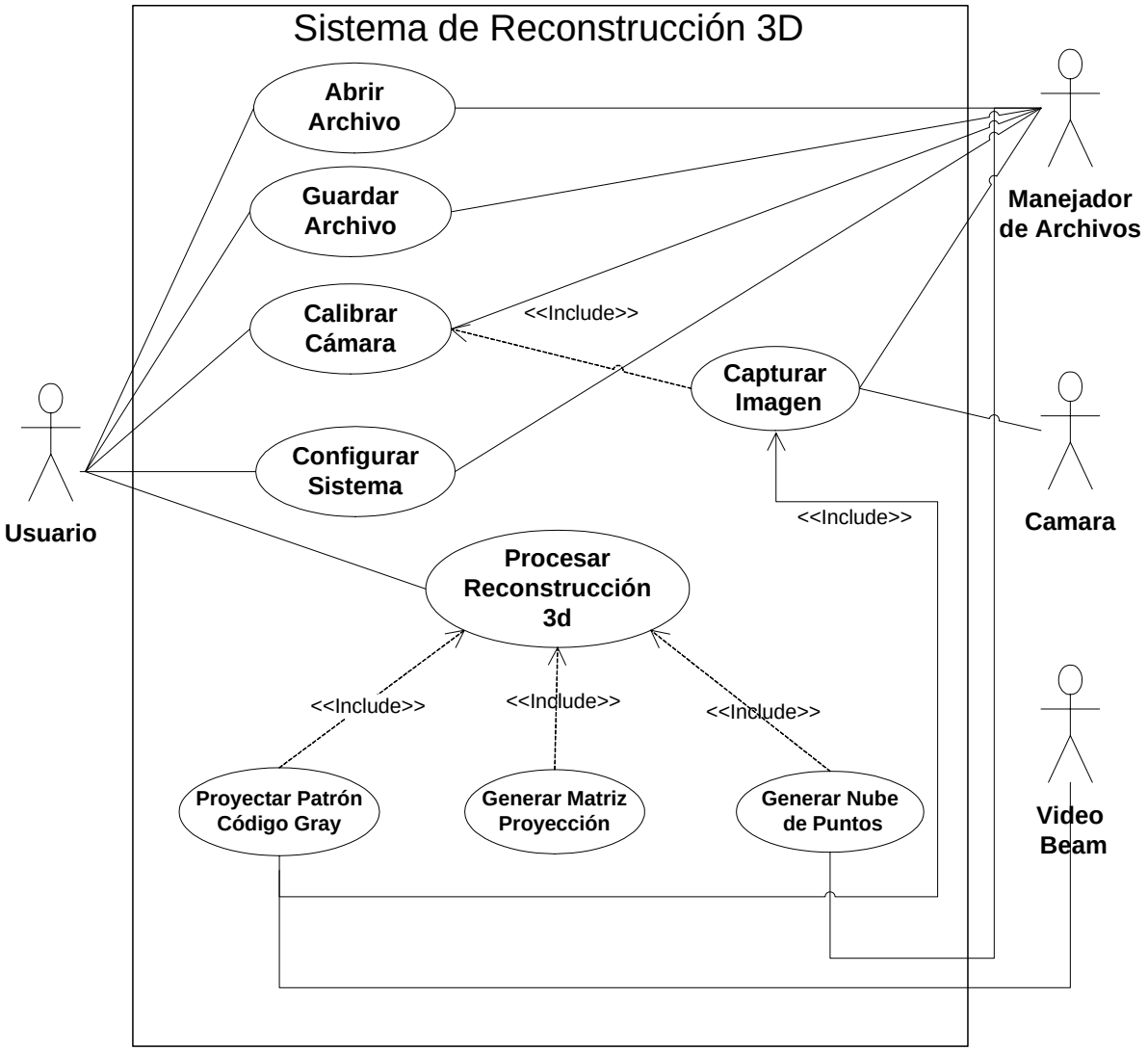


Figura 3.4: Modelo de Casos de uso del sistema.

Fuente: Los Autores.

3.4.2 ANALISIS

El objetivo de este flujo de trabajo, es analizar los requisitos, refinarlo y estructurarlos en un conjunto de objetos conceptuales que se denomina Modelo de Análisis, que permite identificar los elementos significativos del sistema desde un punto de vista de la arquitectura con vista a la construcción a un modelo de diseño.

3.4.2.1 Análisis de caso de uso

El análisis de caso de uso se realiza para identificar las clases de análisis cuyos objetivos son necesarios para llevar a cabo el flujo de sucesos del caso de uso, y distribuir su comportamiento entre los objetos de análisis.

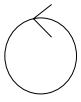
3.4.2.1.1 Identificación de las clases de análisis

En este punto se identifican las clases de control, de entidad y de interfaz para realizar los casos de usos y esbozar sus nombres, responsabilidades, atributos y relaciones.

Existen tres estereotipos pata identificar las clases de análisis: clases de control, clases de entidad y clases de interfaz.

La Clase de Control modela funcionalidad que no corresponde a ningún objeto en particular y que se presenta en algunos casos de uso. Estos objetos generalmente operan sobre varios objetos entidad, realizan algún algoritmo y retornan algún resultado a un objeto de interfaz. Esta representa:

- Coordinación.
- Secuencia.
- Transacción.
- Control de otros objetos.

Su estereotipo es: 

En el sistema se identificaron las siguientes clases de control:

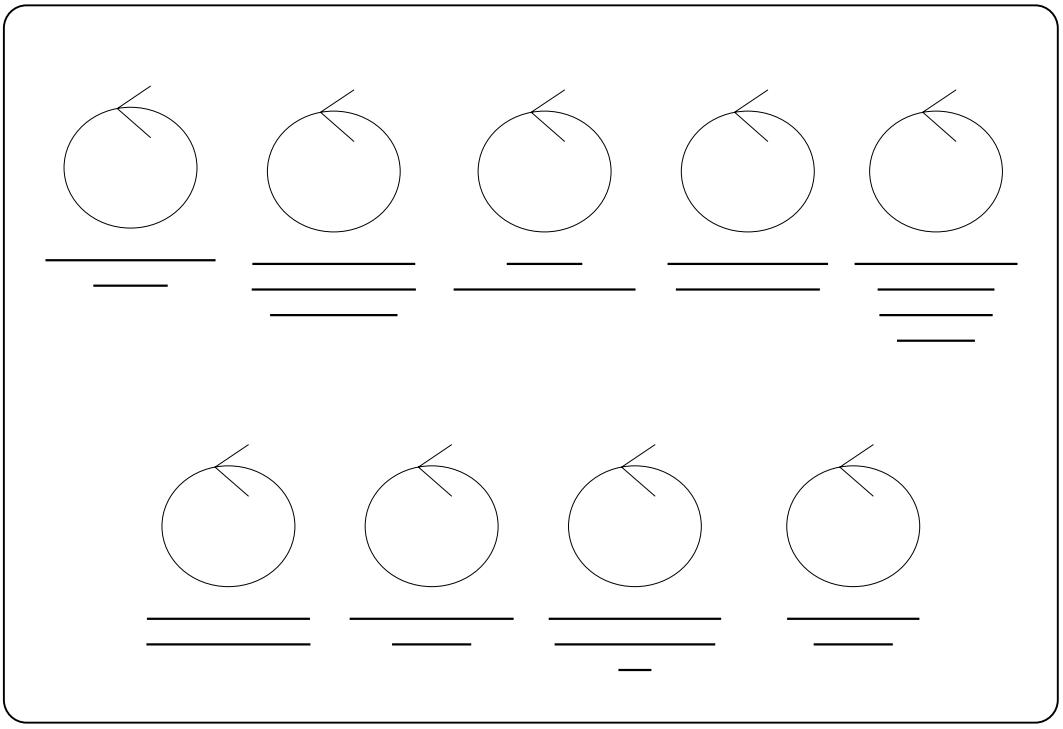


Figura 3.5: Clases de análisis del sistema.
Fuente: Los Autores.

:Gestor Capturar Imagen
:Gestor Generar Patrones Líneas Código Gray
Mat

En la tabla siguiente se muestra una descripción de las clases de control identificadas en el sistema:

Tabla 3.2: Clases de análisis del sistema (1/2).

Gestor Capturar Imagen	
Descripción:	Activa la cámara para tomar la imagen del ambiente
Gestor Generar Patrones de Líneas Código Gray	
Descripción:	Generar los distintos patrones de líneas en código gray a proyectar sobre el objeto y el plano.
Gestor Matriz de Proyección	

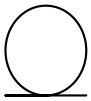
Tabla 3.2: Clases de análisis del sistema (2/2).

stor Guard
Nube De Puntos
Archivo

Descripción:	La clase encarga de procesar las diferentes imágenes de patrones de líneas en código gray para generar la matriz de proyección de plano y la matriz de proyección del objeto
Gestor Generar Nube de Puntos	
Descripción:	Es la encargada de procesar la matriz de proyección del plano y la matriz de proyección del objeto y generar el archivo de nube de puntos
Gestor Archivo Configuración	
Descripción:	Se encarga de verificar los datos introducidos y generar el archivo de configuración del sistema
Gestor Obtener Parámetros Intrínsecos Cámara	
Descripción:	Se encarga de procesar las imágenes de las plantillas de calibración y generar el archivo de parámetros intrínsecos.
Gestor Procesar Reconstrucción 3D	
Descripción:	Se encarga de controlar la ejecución de los Gestores: Generar Patrones de líneas en código Gray, Matriz de proyección, Generar Nube de puntos.
Gestor Guardar Archivo	
Descripción:	Activa la entidad correspondiente para que este se encargue de las operaciones de almacenamiento.
Gestor Abrir Archivo	
Descripción:	Activa la entidad correspondiente para que este se encargue de las operaciones de lectura.

Fuente: Los Autores.

La Clase de Entidad se utiliza para modelar la información que posee una vida larga o que debe sobrevivir cierto periodo de tiempo.

Su estereotipo es: 

En el sistema se identificaron las siguientes clases de entidad:

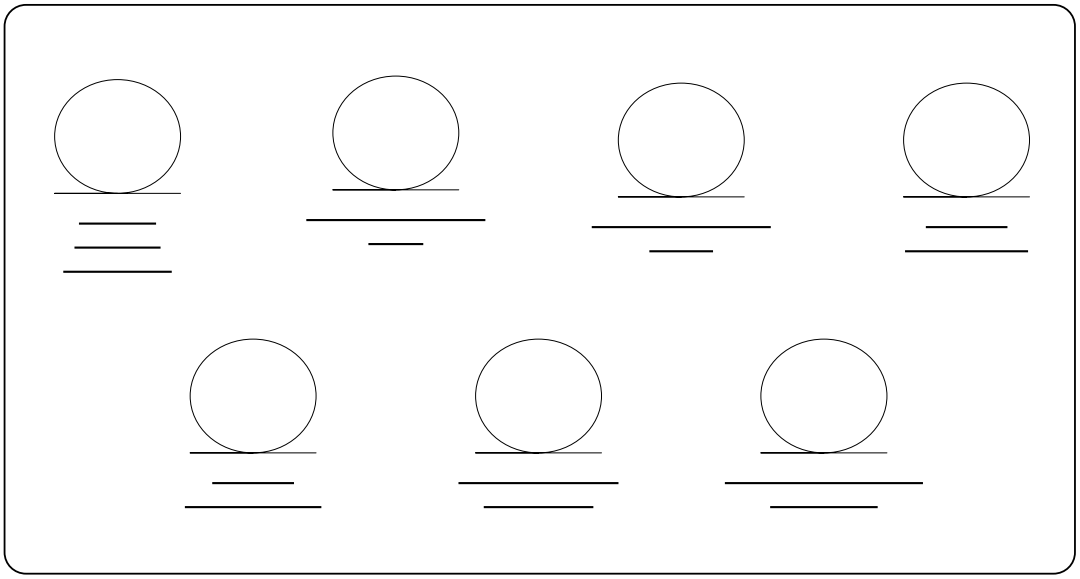


Figura 3.6: Clases de Entidad del sistema.

Fuente: Los Autores.

Tabla 3.3: Clases de Entidad del sistema (1/2).

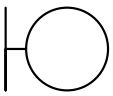
Imagen Patrones Proyección	
Descripción:	Almacena temporalmente las imágenes de de los patrones de proyección en código gray del plano y del objeto.
Matriz Proyección Plano	
Descripción:	Contiene el resultado del procesamiento de los patrones de proyección en código gray del plano.
Matriz Proyección Objeto	
Descripción:	Contiene el resultado del procesamiento de los patrones de

Tabla 3.3: Clases de entidad del sistema (2/2).

	proyección en código gray del objeto.
Archivo Nube de Puntos	
Descripción:	Almacena las coordenadas de la nube de puntos generada por el gestos Generar Nube de Puntos.
Archivo Configuración	
Descripción:	Almacena la información externa del ambiente del sistema que el usuario introduce a este.
Imagen Plantilla Calibración	
Descripción:	Contiene las imágenes de las plantillas de calibración.
Archivo Parámetros Intrínsecos	
Descripción:	Almacena el resultado de las imágenes de plantilla de calibración que el gestor Calibrar cámara obtuvo.

Fuente: Los Autores.

La Clase de Interfaz de Usuario (IU) se utiliza para modelar información y comportamiento que es dependiente de la interfaz actual del sistema, también modela la interacción entre el sistema y sus actores, esto implica recibir información i peticiones de los usuarios y los sistemas externos. Estas clases representan abstracción de ventanas, formularios, paneles, interfaces de comunicación, interfaces de impresoras, sensores, etc.

Su estereotipo es: 

En el sistema se identificaron las siguientes clases de Interfaz de Usuario:

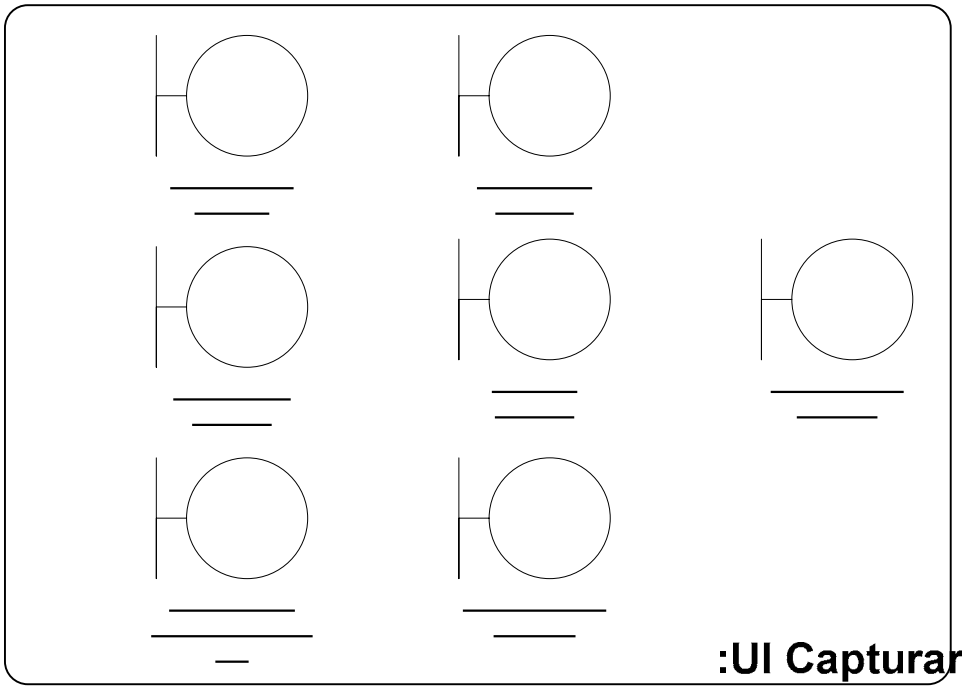


Figura 3.7: Clases de Interfaz del sistema.

Fuente: Los Autores.

Tabla 3.4: Clases de Interfaz del sistema (1/2).

IU Capturar Imagen	
Descripción:	Permite activar la cámara para la captura de las imágenes.
IU Calibrar Cámara	
Descripción:	Permite al usuario Iniciar el proceso de calibración de la cámara.
IU Procesa Reconstrucción 3D	
Descripción:	Permite al usuario Iniciar el proceso para de reconstrucción 3D.
IU Configurar Sistema	
Descripción:	Permite al usuario introducir los valores externos del sistema.
IU Proyectar Imagen	
Descripción:	Permite conectarse con el proyector para enviar los patrones de líneas.
IU Abrir Archivo	

Tabla 3.4: Clases de Interfaz del sistema (2/2).

Descripción:	Ofrece al usuario una interfaz para localizar un archivo de nube de puntos
IU Guardar Archivo	
Descripción:	Ofrece al usuario una interfaz para almacenar un archivo de nube de puntos

Fuente: Los Autores.

3.4.2.1.2 Diagrama de Clases de análisis del sistema

Un diagrama de clases de análisis representa las clases que son posibles clases dentro del sistema y las relaciones que existen entre ellas.

A continuación se muestran los diagramas de clases más importantes de sistema.

3.4.2.1.2.1 Diagrama de clases de análisis del Caso de Uso Calibrar Cámara

En la figura 3.8 podemos observar cómo lleva a cabo mediante la interacción de varias clases de análisis el caso de uso Calibración Cámara, este es activado por el usuario a través de la IU Calibrar Cámara para posteriormente ingresar las imágenes de la plantilla de calibración que servirán para la calibración de la cámara.

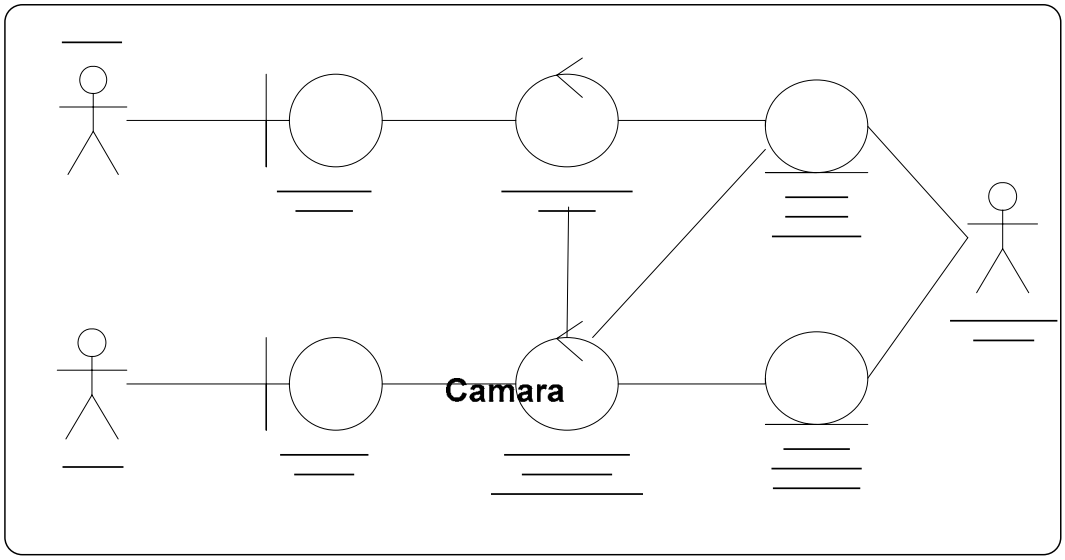


Figura 3.8: Diagrama de clases de análisis del Caso de Uso Calibrar Cámara.

Fuente: Los Autores.

:IU Capturar Imagen

3.4.2.1.2.2 Diagrama de clases del Caso de Uso Configurar Sistema

En la figura 3.9 podemos observar cómo lleva a cobo mediante la interacción de varias clases de análisis el caso de uso Configurar Sistema, el usuario introduce a través de la IU Configuración los parámetros necesarios para el óptimo funcionamiento del sistema, estos parámetros serán almacenados en un archivo de configuración.

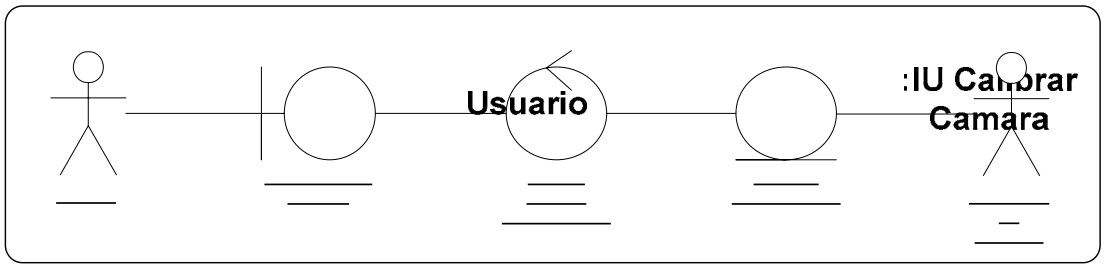


Figura 3.9: Diagrama de clases de análisis del Caso de Uso Configurar Sistema.

Fuente: Los Autores.

3.4.2.1.2.3 Diagrama de clases del Caso de Uso Procesar Reconstrucción 3D

Para la realización de este caso de uso se llevan a cabo varias tareas y se encuentran relacionados tanto actores como otros casos de uso.

Este caso de uso comienza al usuario activar la IU Procesar Reconstrucción 3D quien indica al sistema que está listo para empezar el proceso de captura de imágenes con los patrones de líneas en código gray.

Una vez obtenidas las imágenes del plano y el objeto este comienza el proceso para generar las matrices de proyección del plano y de proyección de objeto las cuales serán almacenadas para su posterior procesamiento por el gestor Generar Nube de Puntos quien se encarga de realizar los cálculos para generar el archivo de nube de puntos.

En la figura 3.10 se observa el diagrama de clases de análisis del caso de uso Generar Nube de Puntos, donde se muestra de manera grafica todas las clases de análisis y actores involucrados para que se lleve a cabo

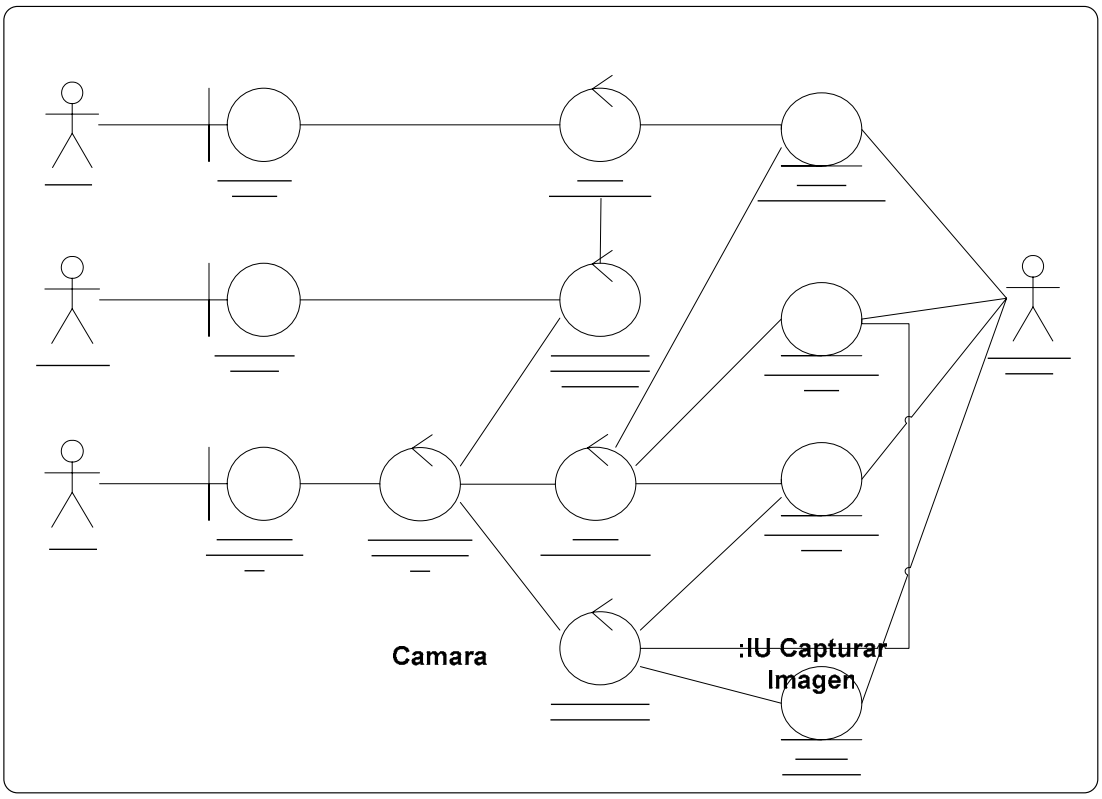


Figura 3.10: Diagrama de clases de análisis del Caso de Uso Procesar Reconstrucción 3D.

Fuente: Los Autores.

Video Beam :IU Proyectar Imagen

3.4.2.1.2.4 Diagrama de clases del Caso de Uso Guardar Archivo

En la figura 3.11 podemos observar cómo lleva a cobo mediante la interacción de varias clases de análisis el caso de uso Guardar Archivo, el usuario introduce a través de la IU Guardar Archivo la ruta y el nombre del archivo que va ser almacenado.

Usuario :IU Procesar Reconstrucción 3D :Gestor Procesar Reconstrucción 3D

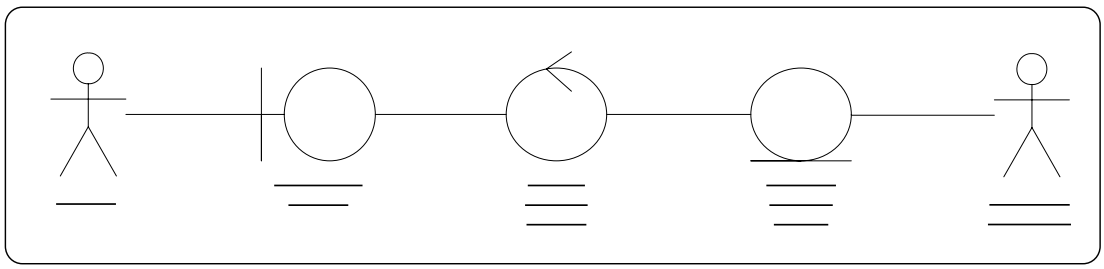


Figura 3.11: Diagrama de clases de análisis del Caso de Uso Guardar Archivo.

Fuente: Los Autores.

3.4.2.1.2.5 Diagrama de clases del Caso de Uso Abrir Archivo

En la figura 3.11 podemos observar cómo lleva a cobo mediante la interacción de varias clases de análisis el caso de uso Abrir Archivo, el usuario interactúa con la IU Abrir Archivo en busca del archivo que se quiere mostrar.

Usuario

:IU Guardar Archivo

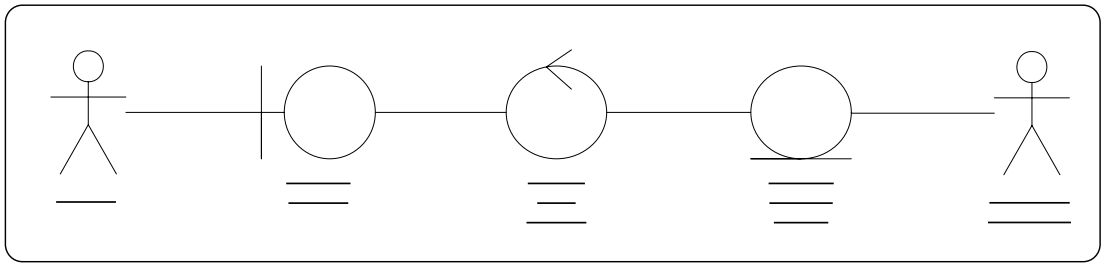


Figura 3.12: Diagrama de clases de análisis del Caso de Uso Abrir Archivo**.**

Fuente: Los Autores.

3.4.2.1.3 Descripción de interacciones entre objetos y análisis

Una vez obtenidas las clases candidatas para la realización de un caso de usos pueden representar sus interacciones a través de un diagrama de colaboración para ilustrar su ejecución y observar adecuadamente la interacción de un objeto con respecto de los demás.

A continuación se muestran los diagramas de colaboración de los casos de uso más importantes del sistema.

En la figura 3.13 ilustra la realización del caso de uso Calibrar Cámara, el proceso empieza con la activación de la IU Calibrar Cámara (flujo 1), para así activar el gestor de obtener parámetros intrínsecos de la cámara (flujo 2) y que a su vez este active el gestor capturar imagen para que prepare la cámara para capturar las imágenes de la plantilla de calibración (flujo 3,4,5), luego de capturadas las imágenes de las plantillas es almacenado a través del Manejador de Archivos (flujo 6,7) y usado por el gestor obtener parámetros intrínsecos para hacer los cálculos de los parámetros intrínsecos de la cámara (flujo 8), estos datos serán almacenados en un archivo de texto por a través del Manejador de Archivos(flujo 9,10).

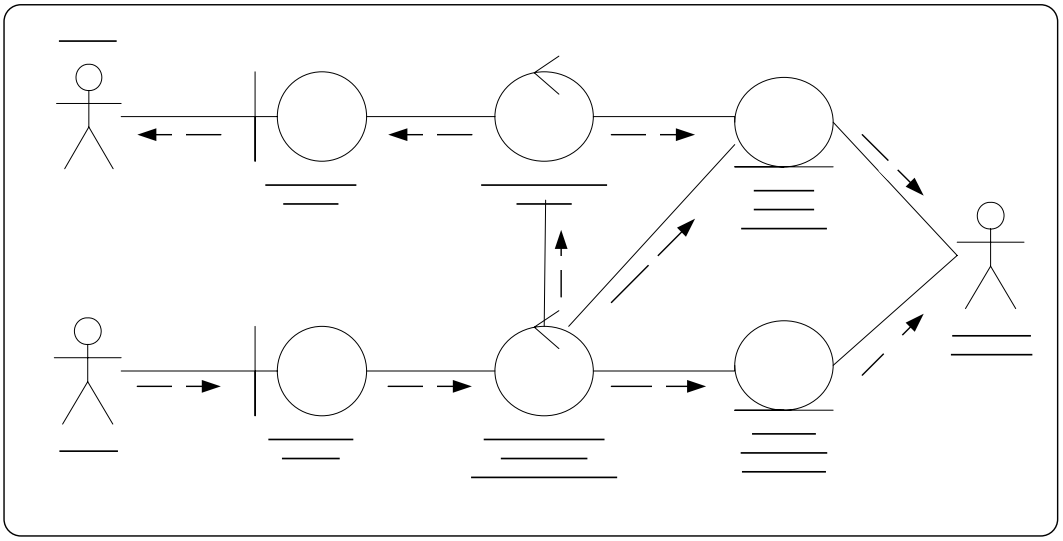


Figura 3.13: Diagrama de colaboración del Caso de Uso Calibrar Cámara.

Flujo de trabajo

1. Solicita Calibrar Cámara 2. Activa Obtener Parámetros Intrínsecos Cámara 3. Inicia la captura de las imágenes 4. Activa la Cámara 5. Capturar Imagen	6. Almacena Imagen de la plantilla 7. Activar Manejador de Archivos 8. Obtiene Imágenes de la plantillas 9. Almacena el archivo de parámetros intrínsecos 10. Activar Manejador de Archivos

Camara

5 **4**

:IU Capturar Imagen **:C**

En la figura 3.14 se muestra la realización del caso de uso Configuración. El usuario solicita configurar el sistema (flujo 1), introduce datos de configuración que serán validados por el gestor de archivo de configuración (flujo 2) y que serán almacenados a través del Manejador de Archivos en un archivo de texto (flujo 3,4).

Flujo de trabajo

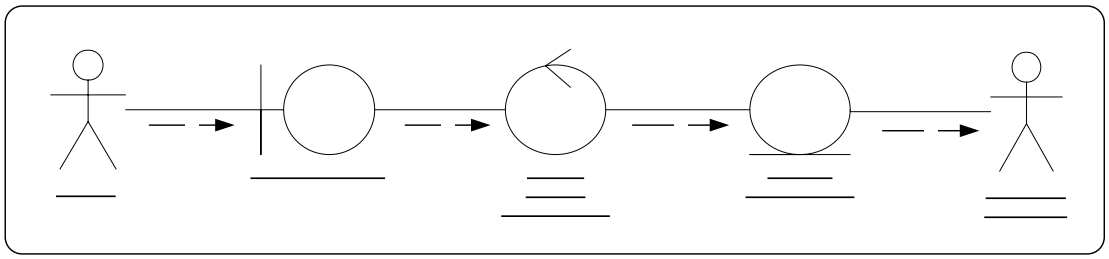


Figura 3.14: Diagrama colaboración del Caso de Uso Configurar Sistema.

Fuente: Los Autores.

Flujo de trabajo

1. Solicita Configurar Sistema
2. Envía Datos Configuración
3. Almacena datos Configuración
4. Activa el Manejador de Archivos

En la figura 3.15 se muestra la realización del caso de uso Procesar Reconstrucción 3D. El proceso comienza con la solicitud del usuario de procesar la reconstrucción 3D (flujo 1) para así activar el al gestor Procesar Reconstrucción 3D lo que activa el gestor Generar Patrones Líneas Código Gray el cual se encarga de generar y enviar los patrones de líneas a video beam a través de la IU Proyectar Imagen (flujo 2,3,4,5), y que as ves este active al Gestor Capturar Imagen para que prepare la cámara para capturar las imágenes de los patrones de líneas del plano y el objeto (flujo 6,7,8), las imágenes de los patrones son almacenados a través de manejador de archivos (flujo 9,10), luego de almacenar todas las imágenes, el Gestor Procesar Reconstrucción 3D activa al Gestor Matriz de Proyección (flujo 11), este obtiene las imágenes de los patrones de líneas en código gray a través del manejador de archivo para procesarlas y generar las matrices plano proyección y objeto

1

2

IU Configuración

Usuario

C

proyección las cuales serán almacenadas a través del Manejador de Archivos (flujo 12,13,14,15,16,17), a continuación se activa el Gestor Generar Nube de Puntos el cual obtiene las matrices de proyección plano y objeto a través de Manejador de Archivos (flujo 18,19,20,21,22), calcula, genera y almacena el archivo de nube de puntos a través del manejador de archivos (flujo 23,24).

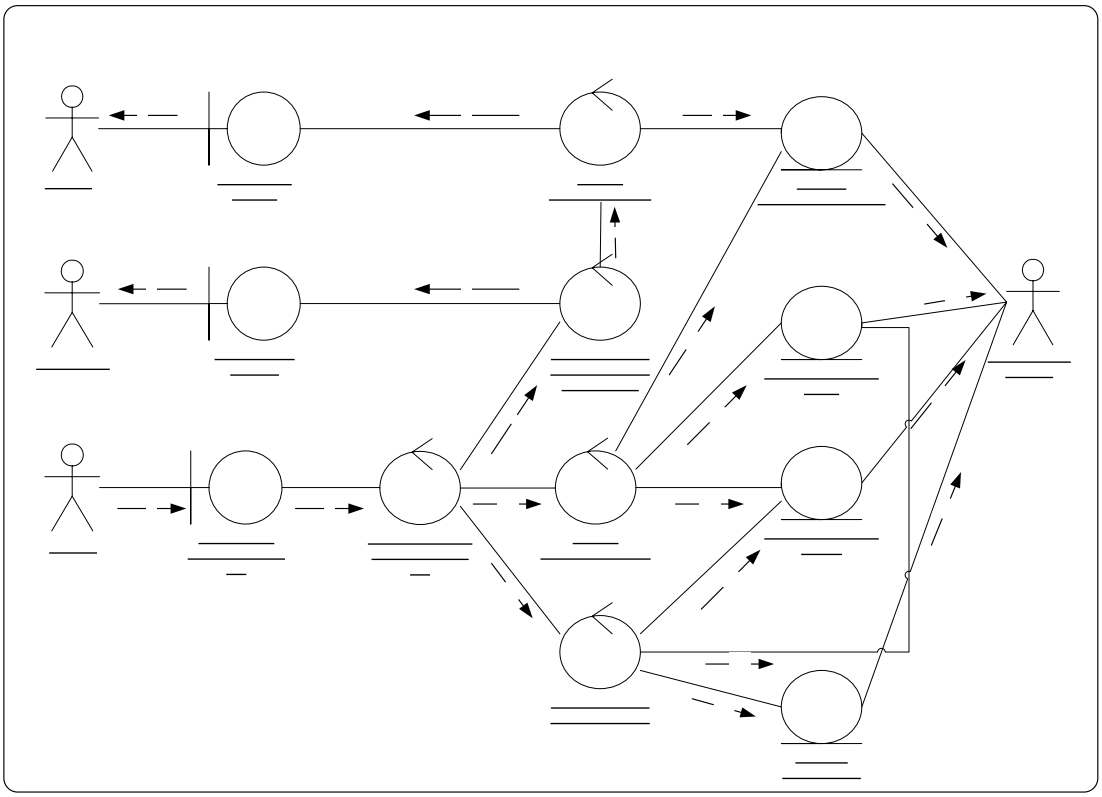


Figura 3.15: Diagrama colaboración del Caso de Uso Generar Nube de Puntos.

Fuente: Los Autores.

Flujo de trabajo

1. Solicita Procesar Reconstrucción 3D 2. Activa Gestor Reconstrucción 3D 3. Activa al Gestor Patrones de líneas Código Gray 4. Envía los patrones 5. Ordena al Video Beam	13. Activa el manejador de Archivos 14. Genera y almacena la Matriz de Proyección Plano 15. Activa el manejador de Archivos 16. Genera y almacena la Matriz de proyección Objeto

Proyectar los patrones 6. Activa al Gestor Capturar Imagen 7. Envía Orden para activar la cámara 8. Activa la Cámara 9. Almacena la imagen capturada **Flujo de trabajo** 10. Activa el manejador de Archivos 11. Activa el Gestor Matriz de Proyección 12. Obtiene las imágenes de los patrones de proyección	17. Activa el manejador de Archivos 18. Activa el Gestor Nube de Puntos 19. Obtiene Matriz Proyección Objeto 20. Activa el manejador de Archivos 21. Obtiene Matriz Proyección Plano 22. Activa el manejador de Archivos 23. Genera y almacena el archivo de Nube de Puntos 24. Activa el manejador de Archivos

En la figura 3.16 se muestra el flujo de trabajo del Caso de Uso Guardar Archivo, el usuario activa el IU Guardar Archivo (flujo 1), este llama al Gestor Guardar Archivo para que solicite el nombre y la ruta del archivo (flujo 2,3), se activa el Manejador de Archivos (flujo 4).

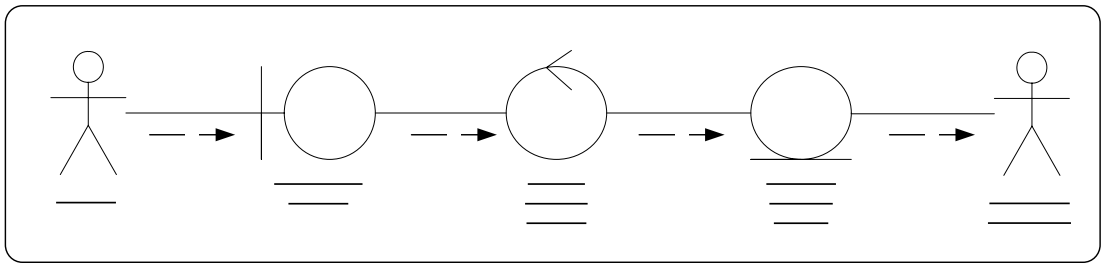


Figura 3.16: Diagrama colaboración del Caso de Uso Guardar Archivo.

Fuente: Los Autores.

Flujo de trabajo

1. Solicita Guardar Archivo 2. Activa el Gestor Guardar Archivo 3. Almacena el Archivo de Nube de Puntos 4. Activa el Manejador de Archivos

En la figura 3.17 se muestra la realización del Caso de Uso Abrir Archivo, el usuario activa el IU Abrir Archivo (flujo 1), este este llama al gestor Abrir archivo quien se encarga de localizar al archivo de nube de puntos por medio del Manejador de Archivos (flujo 2,3,4).

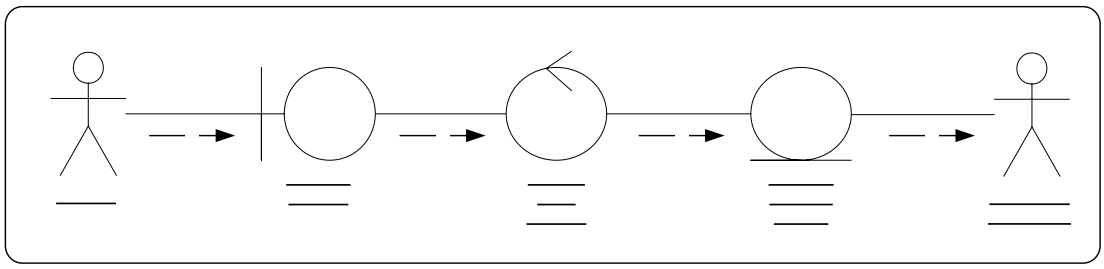


Figura 3.17: Diagrama colaboración del Caso de Uso Abrir Archivo.

Fuente: Los Autores.

Flujo de trabajo

1. Solicita Abrir Archivo
2. Activa el Gestor Abrir Archivo
3. Encuentra y abre el Archivo de Nube de Puntos
4. Activa el Manejador de Archivos

3.4.2.2 Análisis de la arquitectura

El propósito de analizar la arquitectura es esbozar el modelo de análisis y la arquitectura mediante la identificación de paquetes de análisis de las clases de análisis que se muestran de manera general basándose en los requisitos del sistema.

1 2

3.4.2.2.1 Identificación de paquetes de análisis

Usuario :IU Abrir Archivo

Para identificar los paquetes de análisis se pueden agrupar algunos casos de uso en un paquete y dar la funcionalidad correspondiente dentro del paquete.

El sistema identifica sus paquetes a partir del modelo de casos de uso. Los casos de uso: Generar Nube de Puntos, Configurar Sistema, Generar Matriz de Proyección, Proyectar Patrón Código Gray están agrupados en un mismo paquete de análisis que se denomina gestión de procesamiento por estar implicados en un mismo proceso de negocios.

Los casos de uso Calibrar Cámara y Capturar Imagen son casos de uso que no están implicados en el proceso de negocios anteriormente descrito si no, que cada uno según su funcionalidad se describe en procesos de negocios distintos: Gestión Calibración de cámara y Gestión Captura de imagen.

En la figura 3.16 se observan los distintos casos de uso que agrupados en paquetes de análisis según su funcionalidad.

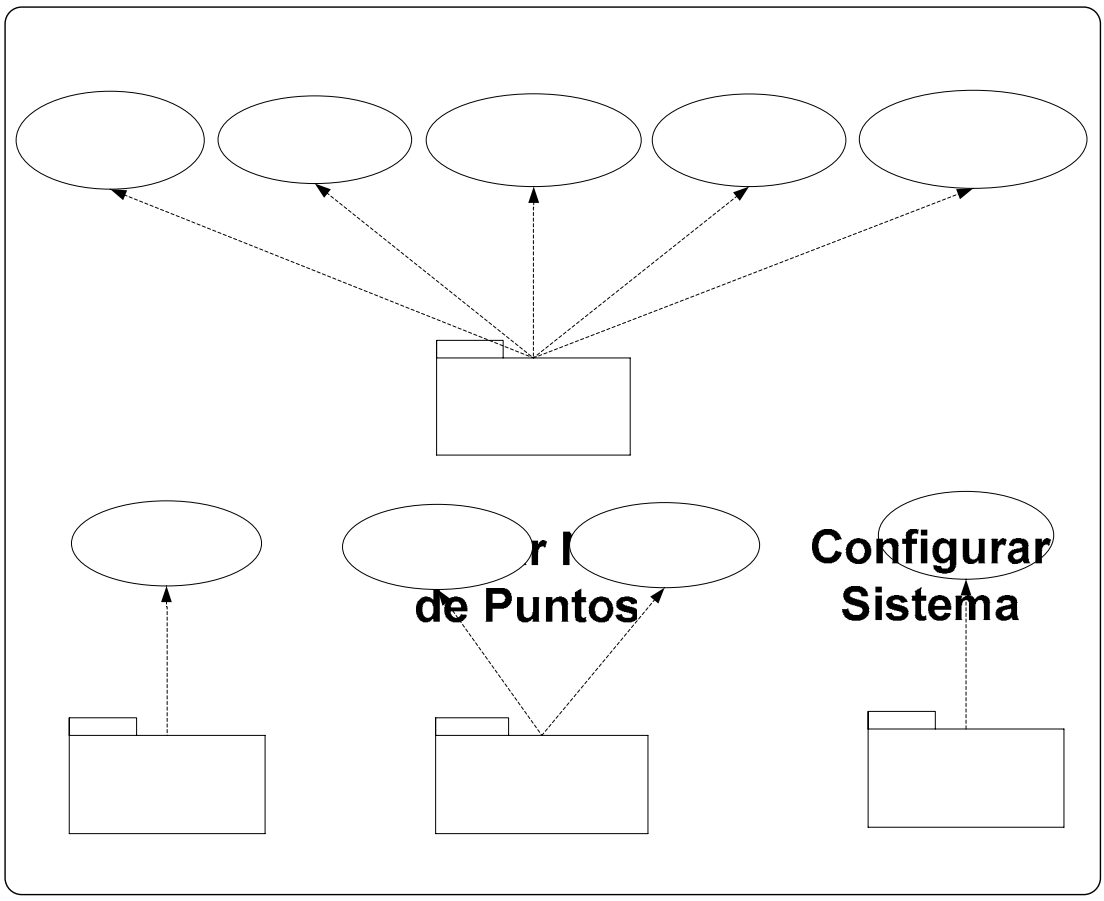


Figura 3.16: Diagrama de paquetes de análisis del sistema.

Fuente: Los Autores.

3.4.3 DISEÑO

En este flujo se define el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso u un sistema, con suficientes detalles como para permitir su interpretación y realización física.

3.4.3.1 Diseño de la arquitectura

En esta parte de la clase de diseño se modela el diseño de la arquitectura mediante la identificación de las clases de diseño relevantes y la identificación de las clases de diseño a partir de las clases de análisis.

3.4.3.1.1 Identificación de las clases de diseño relevantes apara la arquitectura

Para comenzar con la identificación de las clases de diseño lo primero que hay que hacer identificar las clases de análisis y resaltar las más relevantes y a partir de ellas identificar las clases de diseño.

3.4.3.1.2 Identificación de las clases de diseño a partir de las clases de análisis

A partir de las clases de análisis se identificaron las más significativas para esbozar las clases de diseño de realización de los casos de uso.

Las clases encontradas durante el análisis son:

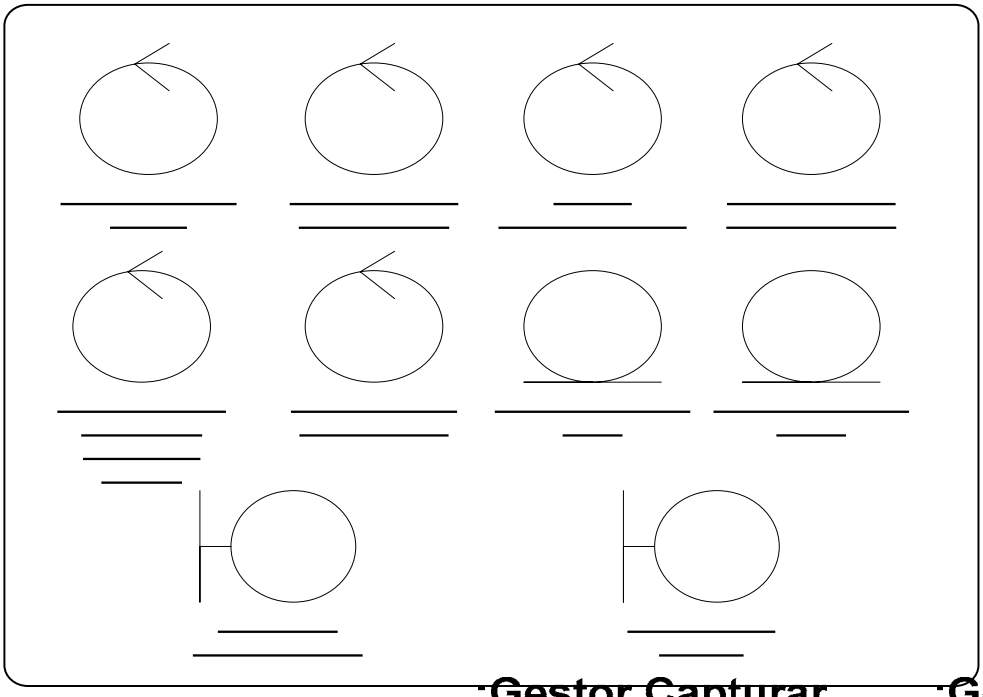


Figura 3.15: Clases de análisis más significativas del sistema

Fuente: Los Autores

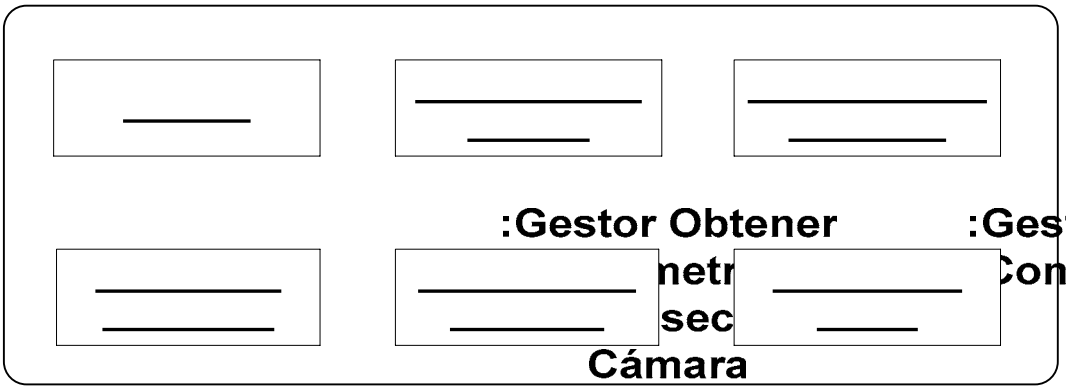


Figura 3.16: Clases de diseño del sistema

Fuente: Los Autores

:UI Generar
Nube De Puntos

3.5 RESULTADOS OBTENIDOS EN LA ITERACION I

Durante esta iteración se llevaron a cabo los flujos de trabajo, requisitos, análisis y diseño obteniendo como resultado, un modelo de caso de uso para el sistema que representa la arquitectura candidata.

Durante la etapa de análisis, se esbozaron las clases de análisis, los paquetes de análisis y los diagramas de colaboración que son la entrada esencial en el diseño.

Y finalmente en la etapa de diseño, se identificaron las clases de diseño relevantes que se irán refinando en la siguiente fase.

3.6 CONCLUSIONES

En el transcurso de la fase de inicio se logro obtener una visión global del sistema propuesto a través del modelo dominio y así establecer lo que debe cubrir la arquitectura. Además se identificaron en detalle los riesgos críticos del sistema y se llevaron a cabo algunas acciones para disminuirlas.

Como conclusión de esta fase, se obtiene una primera versión del modelo de casos de uso, del modelo de análisis y el modelo de diseño, los cuales se trabajaran con más detalle en la fase de elaboración que es la siguiente en este proceso.

CAPITULO IV
FASE DE ELABORACION

4.1 INTRODUCCION

El principal objetivo de la fase de elaboración es definir una arquitectura de software sólida que sirva de base para el desarrollo del sistema a través de los flujos de trabajo: requisitos, análisis y diseño, y garantizar la entrada a la siguiente fase que tiene por nombre fase de construcción.

Durante el desarrollo de esta fase se deba completar la recopilación de los requisitos que quedaron pendientes en la fase de inicio, e identificar los casos de uso faltantes, para mejorar el modelo de caso de uso y así construir una interfaz de usuario que cubra todos los requerimientos del sistema, esperando completarla con éxito en una sola iteración.

Durante esta fase se tienen planificadas las siguientes actividades:

- Determinar una arquitectura estable partiendo de la arquitectura de la fase de inicio, para ello se deba tomar en cuenta aquellos requisitos que en la fase de inicio no fueron vistos, y mejorar el modelo de caso de usos partiendo de ellos.
- Describir una arquitectura candidata mediante las vistas de todos los modelos, detallando los casos de uso para que sirvan de entrada a la nueva fase, y llevar el modelo de análisis hasta el punto que se alcance una arquitectura sólida.

4.2 ITERACION I: SISTEMA DE SOFTWARE

En la fase de elaboración se llevara a cabo la mayor parte del trabajo durante el flujo de requisitos, donde se describirán en detalle nuevos casos de uso, mientras que en el flujo de análisis se identificaran las clases faltantes en la fase de inicio y finalmente en el flujo de diseño se identificaran las clases de diseño para alcanzar una arquitectura final que servirá de entrada a la fase de construcción.

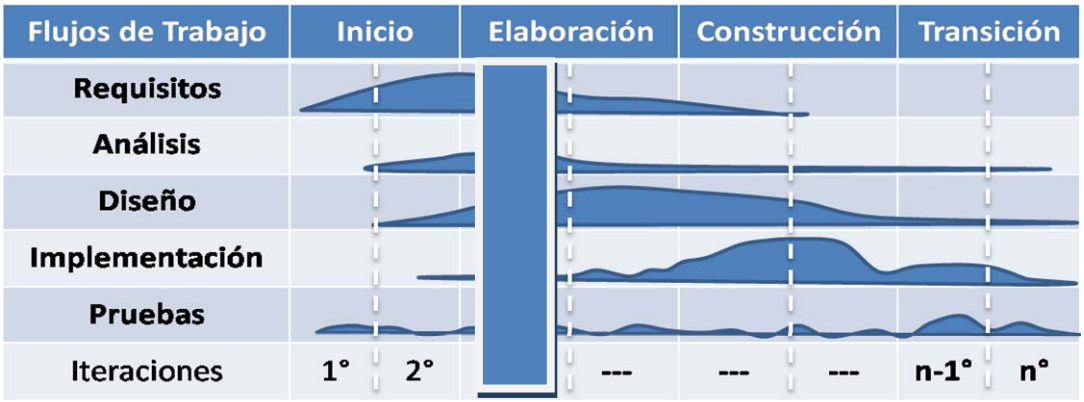


Figura 4.1: Flujo de trabajo durante la fase de elaboración del sistema.

4.2.1 Requisitos

Durante este flujo de trabajo se identificaron nuevos casos de uso presentes en el sistema. Se establecerá la arquitectura final que servirá de entrada a la siguiente fase, fase de construcción.

4.2.1.1 Representación de los Requisitos como Casos de Uso

Se identificaron nuevos casos de uso para el sistema, que desde ahora formaran parte de su arquitectura, y además, se mantuvieron los mismos actores que en la fase anterior

4.2.1.1.1 Determinar y describir sus Actores

No se identificaron nuevos actores en la fase de elaboración.

4.2.1.1.2 Determinar y describir sus Casos de Uso

Analizando el modelo del Casos de Uso expuestos en la fase inicio, se obtuvieron nuevos casos de uso que se describen en la tabla 4.1, teniendo como resultado el fortalecimiento de la arquitectura del sistema.

Tabla 4.1: Nuevos casos de uso del sistema (1/4).

Caso de Uso: Generar Matriz Proyección Plano	
Descripción:	Se encarga de procesar las imágenes de patrones de código gray del plano.
Actor(es):	Ninguno.
Objetivos:	Generar la matriz de proyección del plano.
Caso de Uso: Generar Matriz Proyección Objeto	
Descripción:	Se encarga de procesar las imágenes de patrones de código gray del objeto.
Actor(es):	Ninguno.
Objetivos:	Generar la matriz de proyección del objeto.
Caso de Uso: Convertir a Escala de grises	

Descripción:	Se encarga de convertir a escala de grises las imágenes, tanto del plano como del objeto para que sean usadas por otros casos de uso.
Actor(es):	Ninguno.
Objetivos:	Convertir a escala de grises las imágenes de entrada.
Caso de Uso: Binarizar Imagen	
Descripción:	Se encarga de convertir en blanco y negro las imágenes, tanto del plano como del objeto para que sean usadas por otros casos de uso.
Actor(es):	Ninguno.
Objetivos:	Convertir en blanco y negro las imágenes (Binarizar).
Caso de Uso: Configurar Macara	
Descripción:	Capturar una imagen del objeto y configurar el umbral de binarizacion para obtener una imagen para el proceso de segmentación
Actor(es):	Ninguno.
Objetivos:	Crear la máscara binaria del objeto.
Caso de Uso: Procesar Parámetros de Configuración	
Descripción:	Recoge los parámetros externos del sistema introducidos por el usuario.
Actor(es):	Ninguno.
Objetivos:	Generar un archivo de configuración.
Caso de Uso: Segmentación	
Descripción:	este caso de uso se encarga de tomar una región de interés de la imagen utilizando la máscara de donde se sacara el elemento que será reconstruido
Actor(es):	Ninguno.
Objetivos:	Aislar una parte de la imagen para reconocer el objeto a reconstruir.
Caso de Uso: Calculo Matriz Corrimiento	

Tabla 4.1. Nuevos Casos de Uso del Sistema (2/4).

Descripción:	Se encarga de tomar la matriz de proyección plano y la matriz de proyección objeto y restar sus valores para generar la matriz de corrimiento.
Actor(es):	Ninguno.
Objetivos:	Calcular la Matriz de Corrimiento.
Caso de Uso: Visor GUI 3D	
Descripción:	Esta caso de uso es el encargado de la visualización en un espacio 3D de la nube de puntos del objeto reconstruido.
Actor(es):	Ninguno.
Objetivos:	Mostrar la nube de puntos del objeto en el visor.

Tabla 4.1: Nuevos Casos de Uso del Sistema (3/4).

Caso de Uso: Control GUI 3D	
Descripción:	Este caso de uso de encarga de controlar las acciones que ejerce el usuario sobre el visor GUI 3D
Actor(es):	Ninguno.
Objetivos:	Controlar el GIU 3D.
Caso de Uso: Limpiar GUI	
Descripción:	Se encarga de eliminar el la nube de puntos del visor 3D
Actor(es):	Ninguno.
Objetivos:	Elimina la nube de puntos del visor.
Caso de Uso: Mover Izquierda	
Descripción:	Desplaza hacia la izquierda la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Mueve hacia la izquierda la nube de puntos.
Caso de Uso: Mover Derecha	
Descripción:	Desplaza hacia abajo la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Mueve hacia la derecha la nube de puntos.
Caso de Uso: Mover Arriba	

Descripción:	Desplaza hacia arriba la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Mueve hacia arriba la nube de puntos.
Caso de Uso: Mover Abajo	
Descripción:	Desplaza hacia abajo la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Mueve hacia abajo la nube de puntos.
Caso de Uso: Girar Izquierda	
Descripción:	Gira hacia la izquierda la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Gira hacia la izquierda la nube de puntos.

Tabla 4.1: Nuevos Casos de Uso del Sistema (4/4).

Caso de Uso: Gira Derecha	
Descripción:	Gira hacia abajo la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Gira hacia la derecha la nube de puntos.
Caso de Uso: Gira Arriba	
Descripción:	Gira hacia arriba la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Gira hacia arriba la nube de puntos.
Caso de Uso: Gira Abajo	
Descripción:	Gira hacia abajo la nube de puntos en el visor.
Actor(es):	Ninguno.
Objetivos:	Gira hacia abajo la nube de puntos.
Caso de Uso: Reiniciar Posición	
Descripción:	Este caso de uso se encarga de colocar en la posición inicial a la nube de puntos luego de ser desplazada o girada.
Actor(es):	Ninguno.
Objetivos:	Coloca en la posición inicial al a la nube de puntos.

Fuente: Los Autores.

4.2.1.1.3 Modelo de caso de uso

Con la aparición de nuevos casos de uso, fue necesario modificar la arquitectura del sistema hallado durante la fase de inicio, desarrollándose así, un nuevo modelo que contiene los casos de usos identificaos en la fase de elaboración y que se ilustran en la figura 4.2, resaltados en borde naranja.

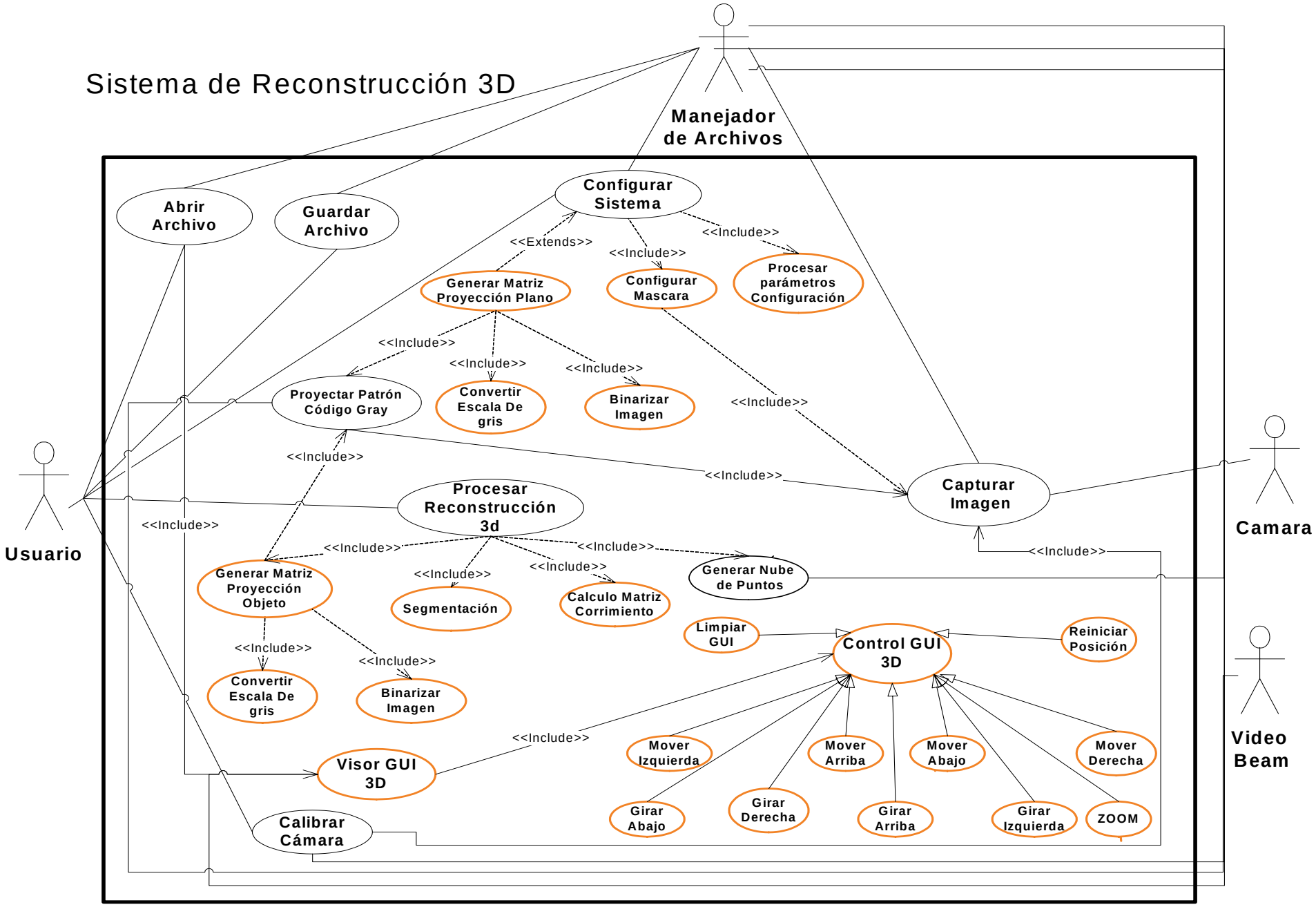


Figura 4.2: Nuevo Modelo de Caso de Uso del Sistema.

4.2.2 Análisis

En la fase de inicio se desarrolla se desarrolla el modelo preliminar de análisis del sistema que contemplaba un bosquejo inicial de las clases de análisis, ahora sobre las bases de estos modelos iniciales se abordaran las actividades de análisis de la fase de elaboración con el fin de ejecutar la línea base de la arquitectura ejecutable del sistema.

4.2.2.1 Identificación de las Clases de Análisis

En el sistema se identificaron las siguientes clases de entidad.

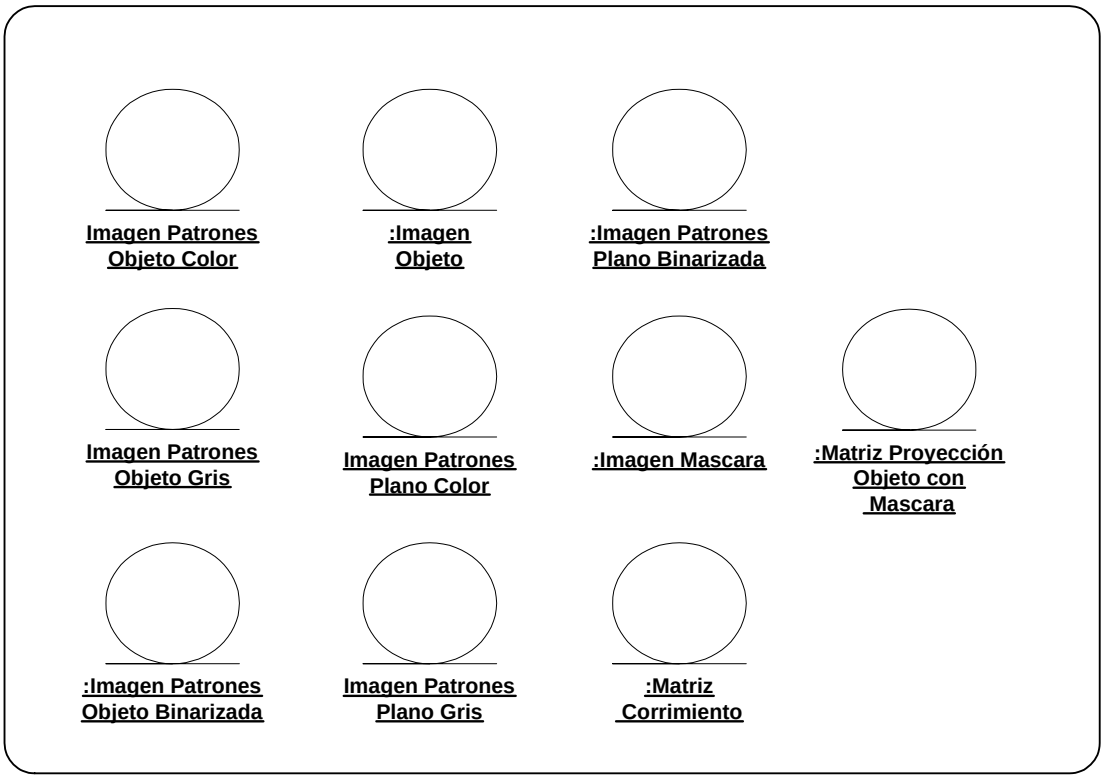


Figura 4.3: Nuevas clases de entidad del sistema.

Fuente: Los Autores.

A continuación en la tabla 4.2 se muestra una breve descripción de las clases de entidad

Tabla 4.2: Descripción de las nuevas clases de entidad del sistema.

Imagen Patrones Objeto Color	
Descripción:	Almacena la imagen de los patrones del objeto en color.
Imagen Objeto	
Descripción:	Almacena la imagen del objeto.
Imagen Patrones Plano Binarizada	
Descripción:	Almacena las imágenes de los patrones del plano en blanco y negro.
Imagen Patrones Objeto Gris	
Descripción:	. Almacena la imagen de los patrones del objeto en escala de grises.
Imagen Patrones Plano Color	
Descripción:	Almacena la imagen de los patrones del plano en color.
Imagen Mascara	
Descripción:	Almacena una imagen bizarizada de objeto según un umbral escogido por el usuario .
Imagen Patrones Objeto Binarizada	
Descripción:	Almacena las imágenes de los patrones del objeto en blanco y negro.
Imagen Patrones Plano Gris	
Descripción:	Almacena la imagen de los patrones del plano en escala de grises.
Matriz Corrimiento	
Descripción:	Almacena el resultado de la resta entre la matriz de proyección objeto y la matriz de proyección plano.
Matriz Proyección Objeto con Mascara	
Descripción:	Almacena la imagen como resultado de la fusión entre la matriz de corrimiento y la imagen mascara.

Fuente: Los Autores.

Para el sistema se identificaron las siguientes clases de control, las cuales permitirán que el sistema tenga el funcionamiento adecuado.

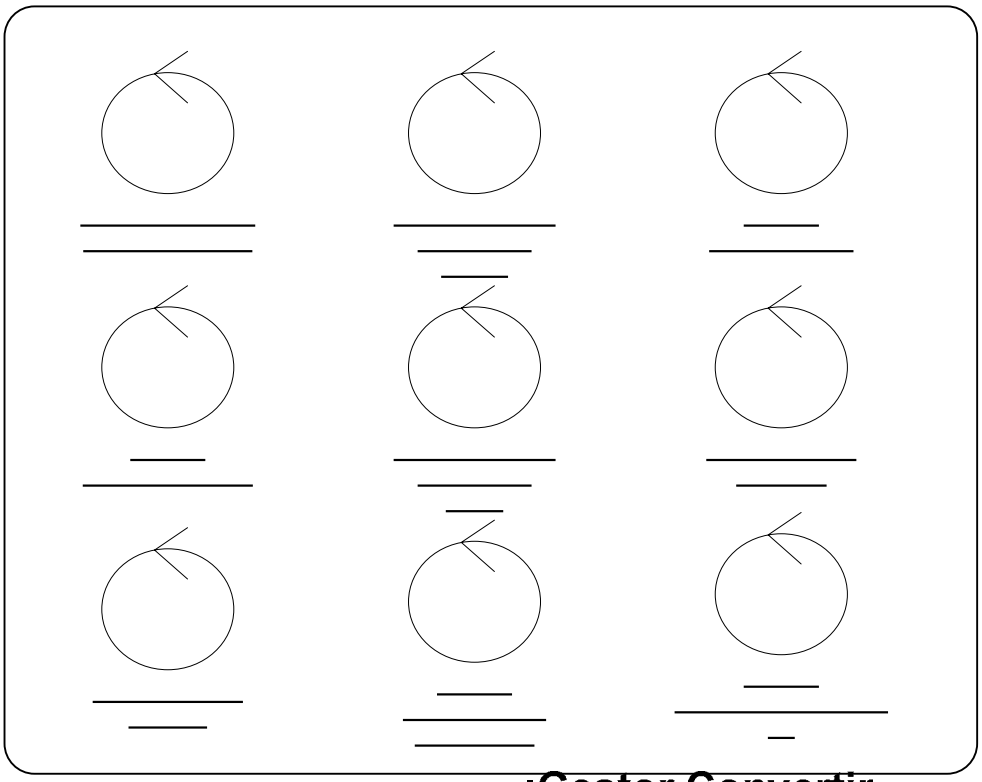


Figura 4.4: **Nuevas clases de control del sistema.**

Fuente: Los Autores.

En la tabla 4.3 se muestra la descripción de las clases de control del sistema.

Tabla 4.3: Descripción de las nuevas clases de control del sistema (1/2).

Gestor Convertir Escala de Grises	
Descripción:	Esta clase de control convierte las imágenes de plano y objeto a escala de grises
Gestor Generar Proyección Objeto	
Descripción:	Toma las imágenes de de patrones de objetos binarizadas y genera la matriz de proyección del objeto.
Gestor Segmentación	
Descripción:	Segmenta la imagen para reconocer el objeto a reconstruir.

Tabla 4.3: Descripción de las nuevas clases de control del sistema (2/2).

Gestor Binarizar Imagen	
Descripción:	Toma las imágenes del plano y el objeto en escala de grises y las convierte a blanco y negro.
Gestor Generar Proyección Plano	
Descripción:	Toma las imágenes de de patrones del plano binarizadas y genera la matriz de proyección del plano.
Configuración Mascara	
Descripción:	Coordina las operaciones del Gestor Capturar Imagen y Configuración Umbral.
Gestor Configurar Umbral	
Descripción:	Es el gestor encargado de obtener la imagen capturada y aplicar el valor de umbral introducido por el usuario.
Gestor Calculo Matriz de Corrimiento	
Descripción:	Se encarga de procesar de la resta entre la matriz de proyección objeto y la matriz de proyección plano.
Gestor Calculo Coordenadas 3D	
Descripción:	Se encarga procesar la matriz de corrimiento y generar la matriz de corrimiento.

Fuente: Los Autores.

4.2.2.2 Diagrama de Clases de Análisis del sistema

A continuación se describen los diagramas de clases de análisis de los casos de uso identificados en esta fase.

4.2.2.2.1 Diagrama de Clases de Análisis del Caso de Uso Segmentación

Mediante la figura 4.5 se observa la realización del caso de uso Segmentación, el cual se inicia con la activación de la clase de control segmentación el cual se encarga de fusionar la imagen mascara con la matriz de proyección objeto y generar la matriz de proyección con máscara.

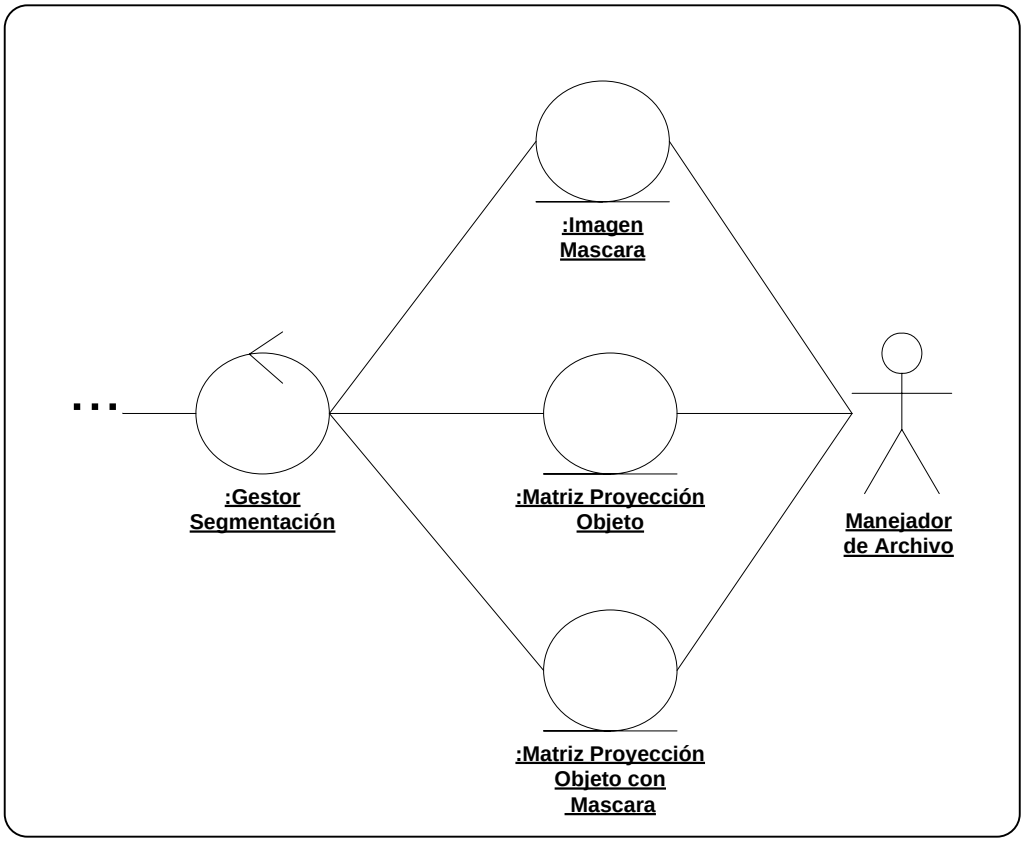


Figura 4.5: Diagrama de clases de análisis del Caso de Uso Segmentación.

Fuente: Los Autores.

4.2.2.2.2 Diagrama de Clases de Análisis del Caso de Uso Calculo de Matriz de Corrimiento

Para el siguiente caso de uso: Calculo de Matriz de Corrimiento, se identifico una clase de control llamada Gestor Matriz de Corrimiento el mismo, procesa la Matriz de Proyección Objeto con Mascara y la Matriz Proyección Plano para generar la Matriz de Corrimiento. En la figura 4.6 se muestra la interacción entre las clases de análisis del caco de uso Cálculo de Matriz de Corrimiento.

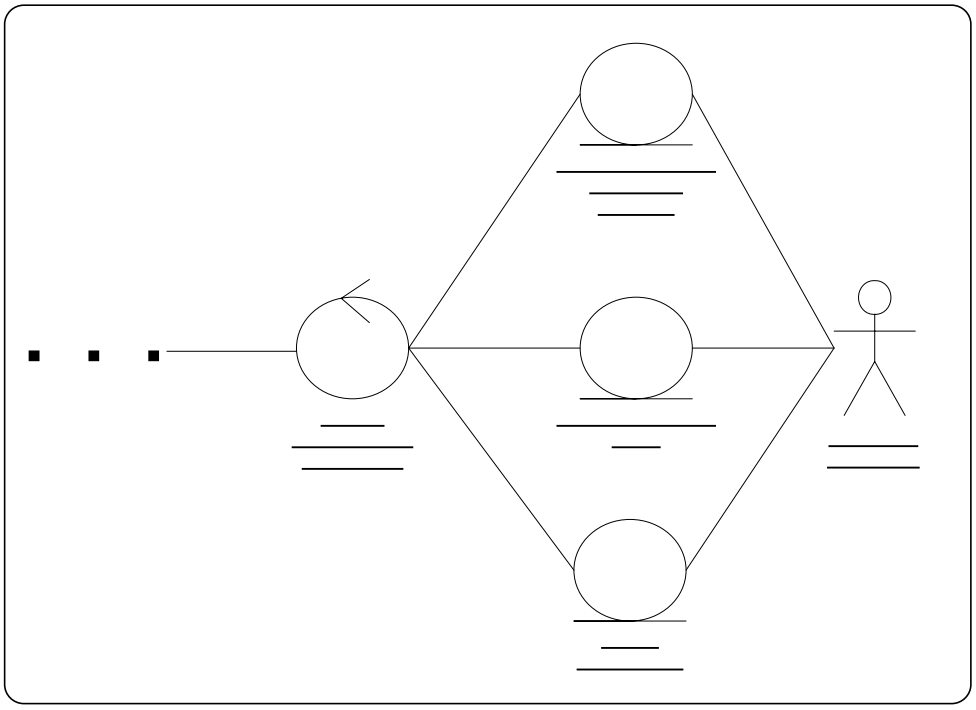


Figura 4.6: Diagrama de clases de análisis del Caso de Uso Calculo de Matriz de Corrimiento.

Fuente: Los Autores.

4.2.2.2.3 Diagrama de Clases de Análisis del Caso de Uso Configurar Mascara

En la figura 4.7 el usuario activa el gestor configurar mascara, a través de la interfaz IU Configurar Sistema, este gestor activa el proceso de capturar imagen, para luego activa el gestor Configuración Umbral para que este aplique el umbral de binarizacion introducido por el usuario, a la Imagen Objeto y cree la Imagen Mascara

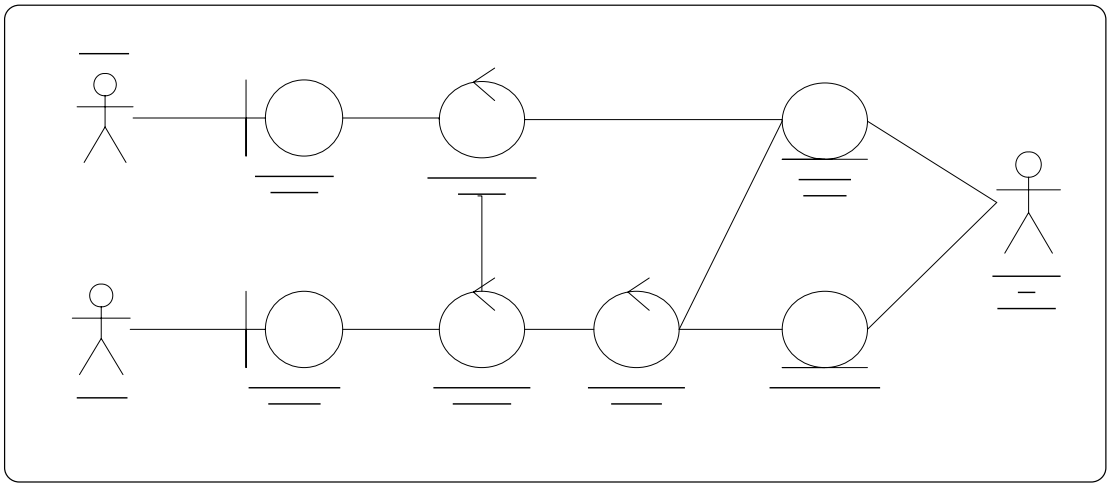


Figura 4.7: Diagrama de clases de análisis del Caso de Uso Calculo de Configurar Mascara.

Fuente: Los Autores.

4.2.2.2.4 Diagrama de Clases de Análisis del Caso de Uso Generar Matriz Proyección Plano

Camara

en la figura 4.8 se explica el funcionamiento del Caso de uso Generar Matriz de Proyección Plano, en este caso de uso el Gestor Matriz de Proyección Plano activa los gestores de la siguiente manera: Gestor Capturar Imagen el cual se encarga de capturar la imagen y almacenarla con el nombre Imagen Patrón Plano Color, a

:IU Capturar Imagen

:Gestor Captu Imagen

continuación activa el Gestor Convertir a Escala de Grises el cual procesa la imagen anterior y almacena la Imagen Patrón Plano Gris, seguidamente es activado el Gestor Binarizar Imagen el cual procesa la Imagen Patrones Gris y crea la Imagen Patrón Plano Binarizada y por ultimo activa el Gestor Generar Proyección Plano, este toma las Imágenes Patrón Plano Binarizadas las procesa crea y almacena la Matriz Proyección Plano.

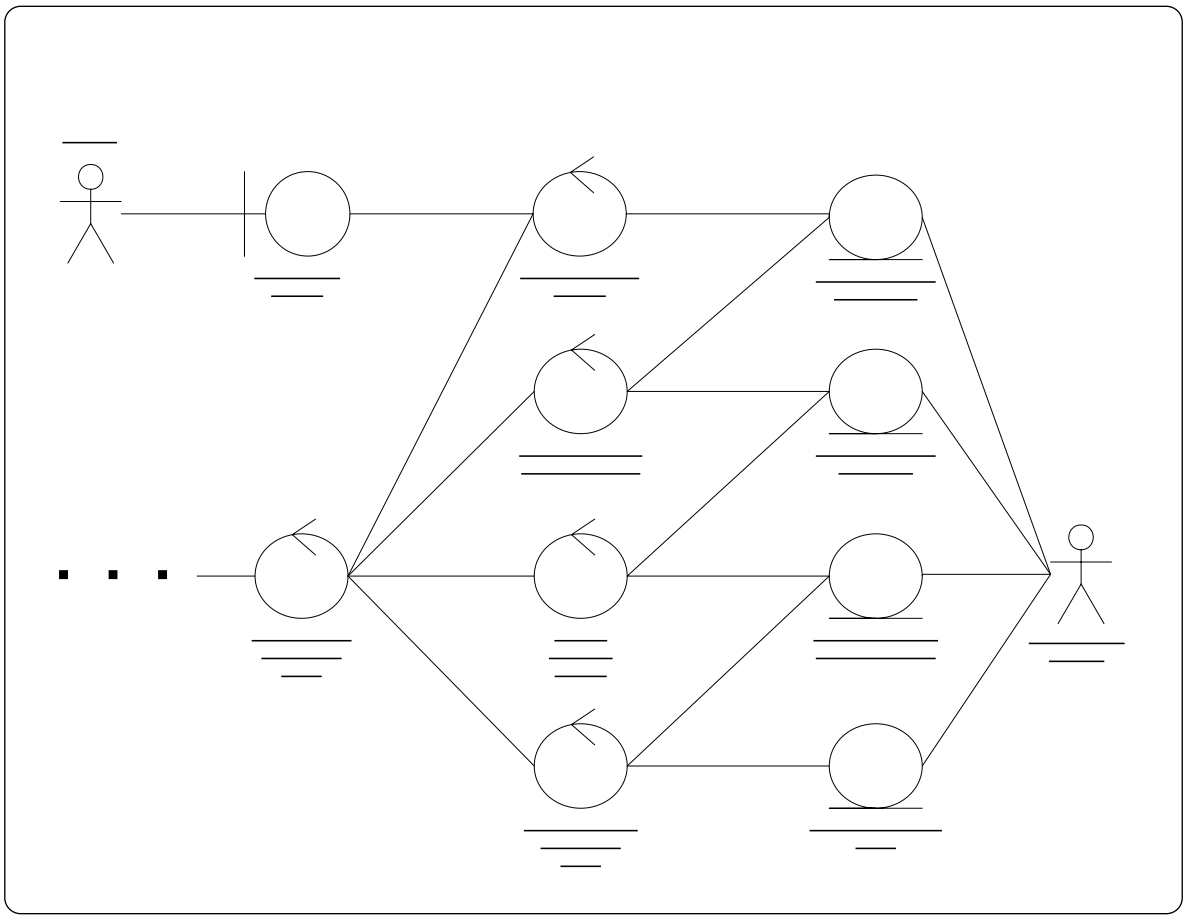


Figura 4.8: Diagrama de clases de análisis del Caso de Uso Generar Matriz de Proyección Plano.

Fuente: Los Autores.

4.2.2.2.5 Diagrama de Clases de Análisis del Caso de Uso Generar Matriz Proyección Objeto

En la figura 4.9 se explica el funcionamiento del Caso de uso Generar Matriz de Proyección Objeto, en este caso de uso el Gestor Matriz de Proyección Objeto activa los gestores de la siguiente manera: Gestor Capturar Imagen el cual se encarga de capturar la imagen y almacenarla con el nombre Imagen Patrón Objeto Color, a continuación activa el Gestor Convertir a Escala de Grises el cual procesa la imagen anterior y almacena la Imagen Patrón Objeto Gris, seguidamente es activado el Gestor Binarizar Imagen el cual procesa la Imagen Patrones Gris y crea la Imagen Patrón Objeto Binarizada y por ultimo activa el Gestor Generar Proyección Objeto, este toma las Imágenes Patrón Plano Binarizadas las procesa crea y almacena la Matriz Proyección Objeto.

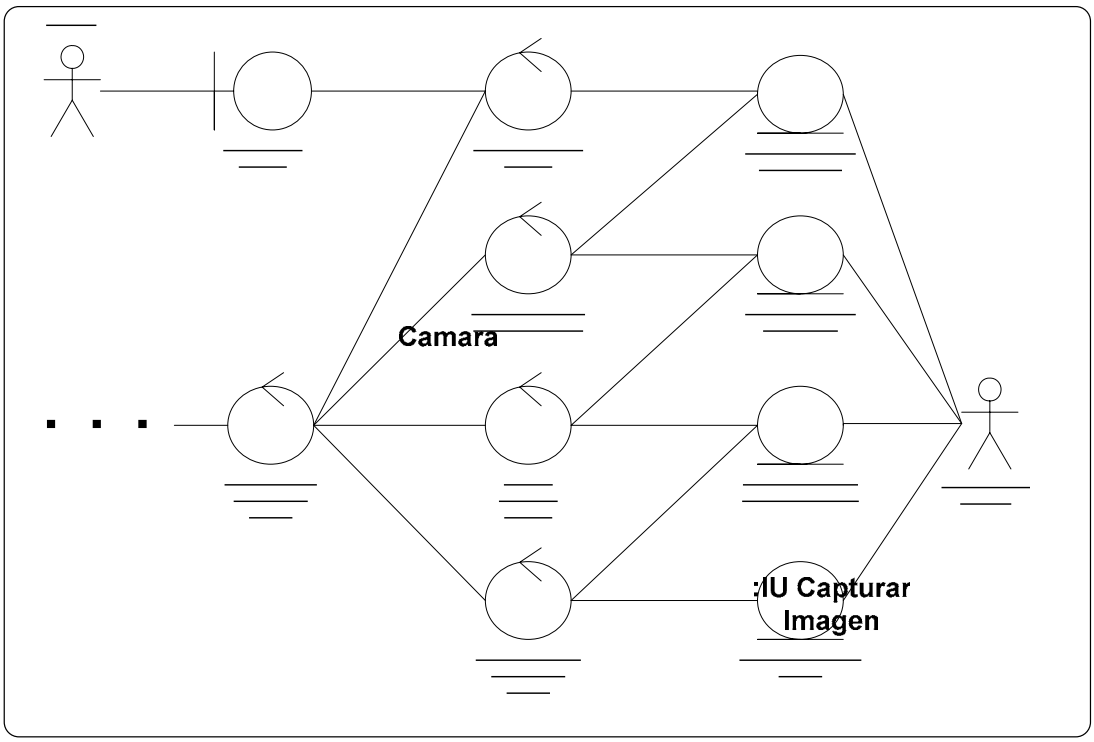


Figura 4.9: Diagrama de clases de análisis del Caso de Uso Generar Matriz de Proyección Plano.

Fuente Los Autores.

4.2.2.3 Diagrama de Colaboración del Sistema

A continuación se describen detalladamente los diagramas de clases de colaboración de los casos de uso identificados en esta fase.

4.2.2.3.1 Diagrama de Clases de Colaboración del Caso de Uso Segmentación

El siguiente diagrama, figura 4.10 se muestra detalladamente las secuencia de acciones que se llevan a cabo durante la segmentación de la imagen, el proceso se inicia con la solicitud de una clase externa para llevar a cabo el proceso de segmentación (flujo 1), luego el gestor segmentación obtiene a través del manejador de archivos la Imagen Mascara (flujo 2,3), también obtiene la matriz de proyección Objeto a través del

manejador de archivos (flujo 4,5) y luego almacena la Matriz de Proyección Objeto con Mascara que es el resultado de la combinación de las dos imágenes anteriores a través del manejador de archivos (flujo 6,7).

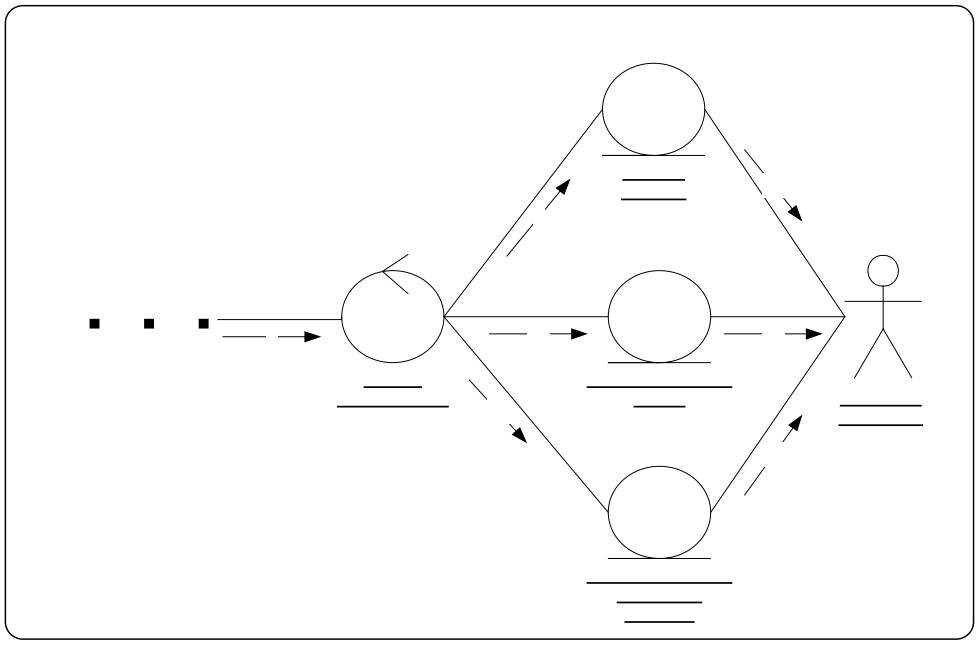


Figura 4.10: Diagrama de clases de Colaboración del Caso de Uso Segmentación.

Fuente: Los Autores.

Flujos de trabajo

8. Solicita proceso de segmentación de las imágenes 9. Obtiene Imagen Mascara 10. Activa Manejador de Archivos 11. Obtiene Matriz de Proyección Objeto	12. Activa Manejador de Archivos 13. Almacena Matriz de Proyección Objeto con Mascara 14. Activa Manejador de Archivos

8

4.2.2.3.2 Diagrama de Clases de Colaboración del Caso de Uso Calculo Matriz Corrimiento

:Gestor Segmentació

El siguiente diagrama, figura 4.11 se muestra detalladamente las secuencia de acciones que se llevan a cabo en el caso de uso Matriz de Corrimiento, el proceso se inicia con

la solicitud de una clase externa para llevar a cabo el proceso de Cálculo de Matriz de Corrimiento (flujo 1), luego el gestor Calculo de Matriz de Corrimiento obtiene a través del manejador de archivos la Matriz Proyección Objeto con Mascara (flujo 2,3), también obtiene la Matriz de Proyección Plano a través del manejador de archivos (flujo 4,5) y luego almacena la Matriz Corrimiento que es el resultado de la resta de Matriz Proyección Objeto con Mascara y Matriz de Proyección Plano través del manejador de archivos (flujo 6,7).

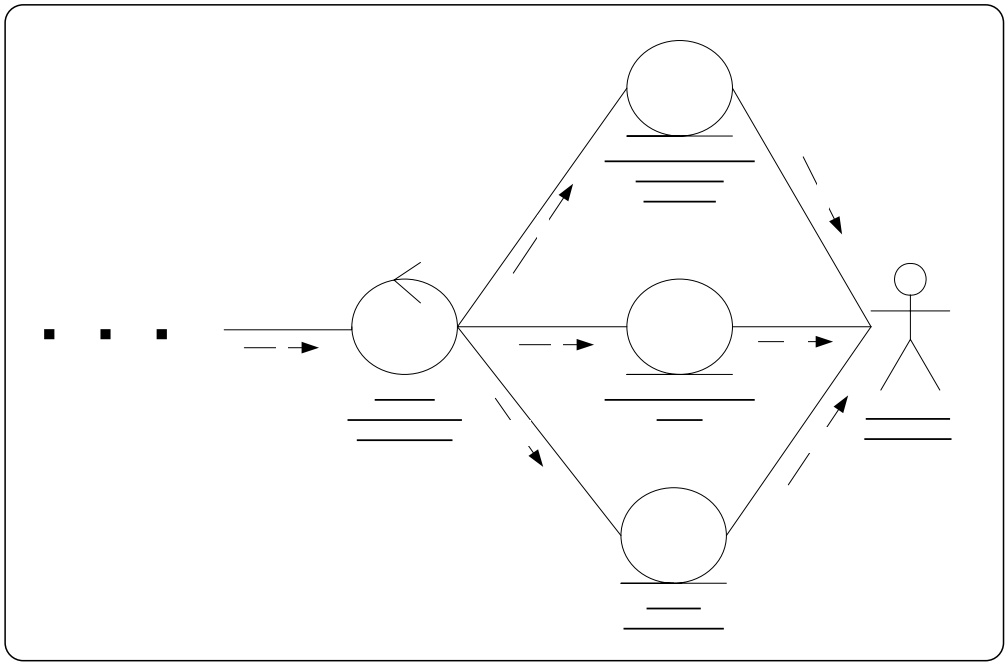


Figura 4.11: Diagrama de clases de Colaboración del Caso de Uso Calculo Matriz Corrimiento.

Fuente: Los Autores.

Flujos de trabajo

6. Solicita proceso de Cálculo de Matriz de Corrimiento 7. Obtiene la Matriz de Proyección Objeto con Mascara 8. Activa Manejador de Archivos	10. Activa Manejador de Archivos 11. Almacena Matriz de Corrimiento 12. Activa Manejador de Archivos

9. Obtiene Matriz de Proyección Plano	

4.2.2.3.3 Diagrama de Clases de Colaboración del Caso de Configurar Mascara

El siguiente diagrama, figura 4.12 se muestra detalladamente las secuencia de acciones que se llevan a cabo en el caso de uso Configurar Mascara, este comienza cuando el usuario solicita la activación de Configurar Mascara a través de la IU Configurar Sistema (flujo1,2), esta solicitud activa el Gestor de Configuración mascara el cual a su vez activa el Gestor Capturar Imagen que se encarga de activar la cámara a través de la IU Capturar Imagen y almacenarla usando usuario al Manejador de Archivos (flujo 3,4,5,6,7), luego el Gestor Configurar Mascara activa el Gestor Configurar Umbral y este obtiene la Imagen Objeto por medio del Manejador de Archivos para aplicarle el umbral introducido por usuario (flujo 8,9,10), luego almacena la imagen mascara usando el Manejador de Archivos (flijo 11,12).

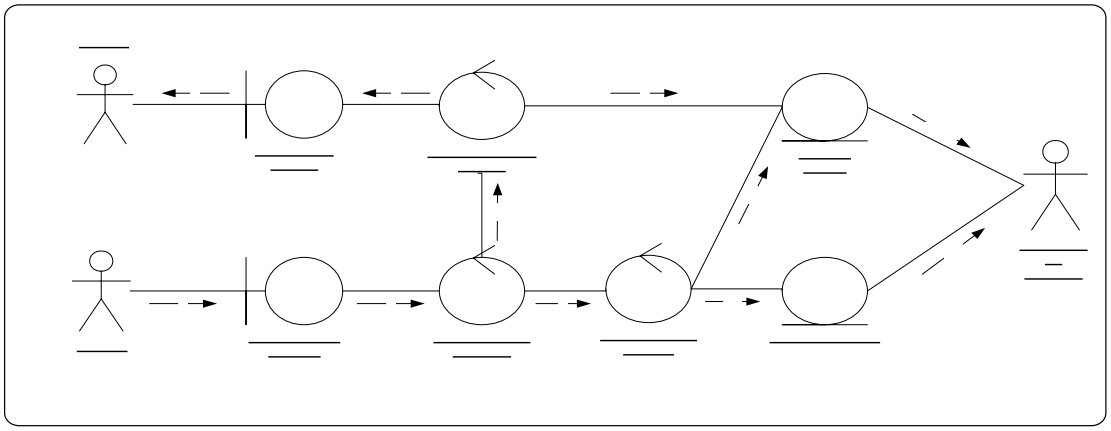


Figura 4.12: Diagrama de clases de Colaboración del Caso de Uso Configurar Mascara.

Fuente: Los Autores.

Flujos de trabajo

1. El usuario solicita la IU Configura Sistema 2. Activa el Gestor Configuración Mascara 3. Activa el Gestor Capturar Imagen 4. Solicita Capturar la imagen 5. Activa la cámara 6. Almacena la Imagen	7. Activa el manejador de archivos 8. Activa el Gestor Configurar umbral 9. Obtiene la Imagen Objeto 10. Activa el Manejador de Archivos 11. Aplica el umbral y almacena la imagen 12. Activa el Manejador de Archivos

4.2.2.3.4 Diagrama de Clases de Colaboración del Caso de Generar Matriz de Proyección Plano

La clase Gestor Matriz de Proyección Plano es activada por una clase externa (flujo 1), está activa el Gestor Capturar Imagen que se encarga de activar la cámara a través de la IU Capturar Imagen y almacenar la Imagen Patrón Plano Color usando usuario al Manejador de Archivos (flujo 2,3,4,5,6), luego es activado el Gestor Convertir a Escala de Grises, este obtiene la Imágenes Patrón Plano Color a través de Manejador de Archivos para convertirlas a escala de grises y almacenar la imagen Patrones Plano Gris a través del Manejador de Archivos (flujo 7,8,9,10,11), ahora es activado el Gestor Binarizar Imagen, este obtiene la Imágenes Patrón Plano Gris a través de Manejador de Archivos para convertirlas en imágenes de blanco y negro (Imagen Patrones Plano Binarizadas) y almacenarlas usando del Manejador de Archivos (flujo 12,13,14,15,16), seguidamente es activado el Gestor Generar Proyeccion Plano, este obtiene la Imágenes Patrón Plano Binarizadas a través de Manejador de Archivos

procesarlas y generar la Matriz de Pryeccion Plano la cual es almacenada a través del manejador de archivos (flujo 17,18,19,20,21).

Este proceso se explica gráficamente en la figura 4.13.

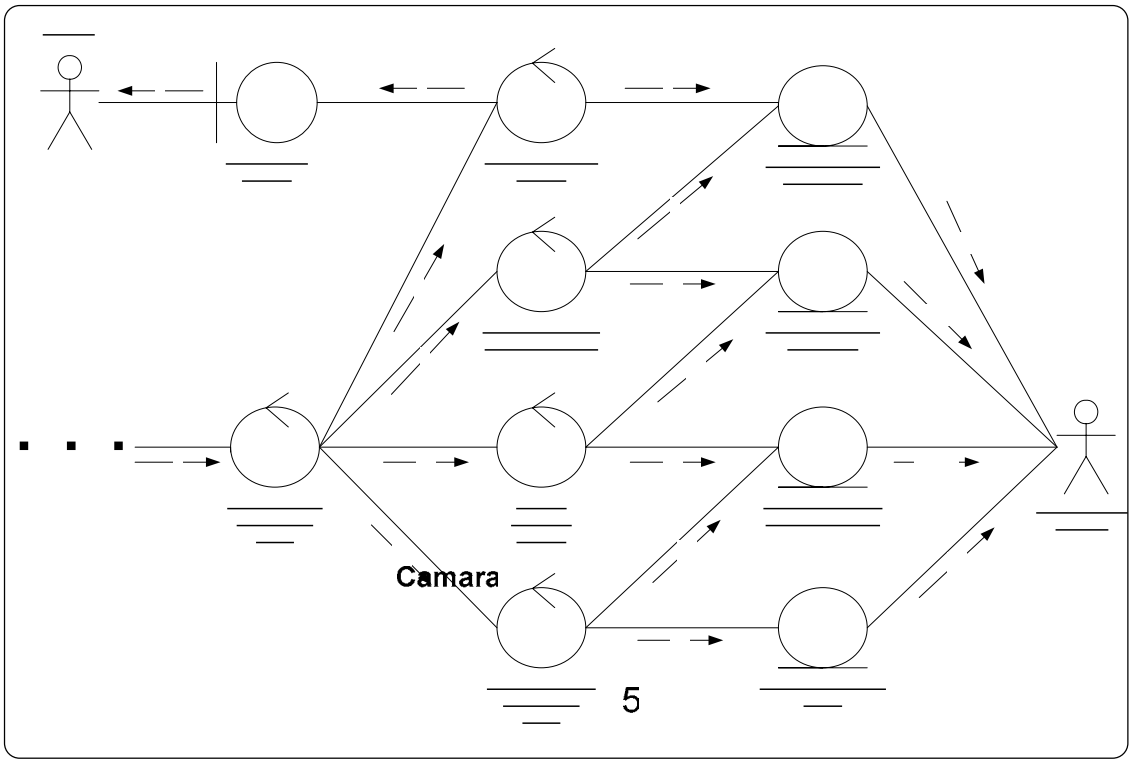


Figura 4.13: Diagrama de clases de Colaboración del Caso de Uso Generar Matriz Proyección Plano.

Fuente: Los Autores.

Flujos de Trabajo

2. Activa el Gestor Matriz Proyección Plano 3. Activa el Gestor Capturar Imagen 4. Solicita Capturar la imagen 5. Activa la cámara 6. Almacena la Imagen 7. Activa el manejador de archivos 8. Activa al Gestor Convertir	13. Activa el Gestor Binarizar Imagen 14. Obtiene Imagen Patrones Plano Gris 15. Activa el Manejador de Archivos 16. Binariza y almacena las Imágenes Patrones Plano Binarizadas 17. Activa el Manejador de

Escala de Grises 9. Obtiene Imágenes Patrones Plano Color 10. Activa el Manejador de Archivos 11. Convierte las imágenes en escala de grises y almacena Imagen Patrones Plano Gris 12. Activa al Manejador de Archivos	Archivos 18. Activa al gestor Generar Proyeecion Plano 19. Obtiene Imágenes Patrones Plano Binarizada 20. Activa Manejador de Archivos 21. Genera y almacena la Matriz de Proyección Plano 22. Activa el Manejador de Archivos

4.2.2.3.5 Diagrama de Clases de Colaboración del Caso de Generar Matriz de Proyección Objeto

La clase Gestor Matriz de Proyección Objeto es activada por una clase externa (flujo 1), está activa el Gestor Capturar Imagen que se encarga de activar la cámara a través de la IU Capturar Imagen y almacenar la Imagen Patrón Objeto Color usando usuario al Manejador de Archivos (flujo 2,3,4,5,6), luego es activado el Gestor Convertir a Escala de Grises, este obtiene la Imágenes Patrón Objeto Color a través de Manejador de Archivos para convertirlas a escala de grises y almacenar la imagen Patrones Objeto Gris a través del Manejador de Archivos (flujo 7,8,9,10,11), ahora es activado el Gestor Binarizar Imagen, este obtiene la Imágenes Patrón Objeto Gris a través de Manejador de Archivos para convertirlas en imágenes de blanco y negro (Imagen Patrones Objeto Binarizadas) y almacenarlas usando del Manejador de Archivos (flujo 12,13,14,15,16), seguidamente es activado el Gestor Generar Proyeccion Objeto, este obtiene la Imágenes Patrón Objeto Binarizadas a través de Manejador de Archivos procesarlas y generar la Matriz de Pryeccion Objeto la cual es almacenada a través del manejador de archivos (flujo 17,18,19,20,21).

Este proceso se explica gráficamente en la figura 4.14.

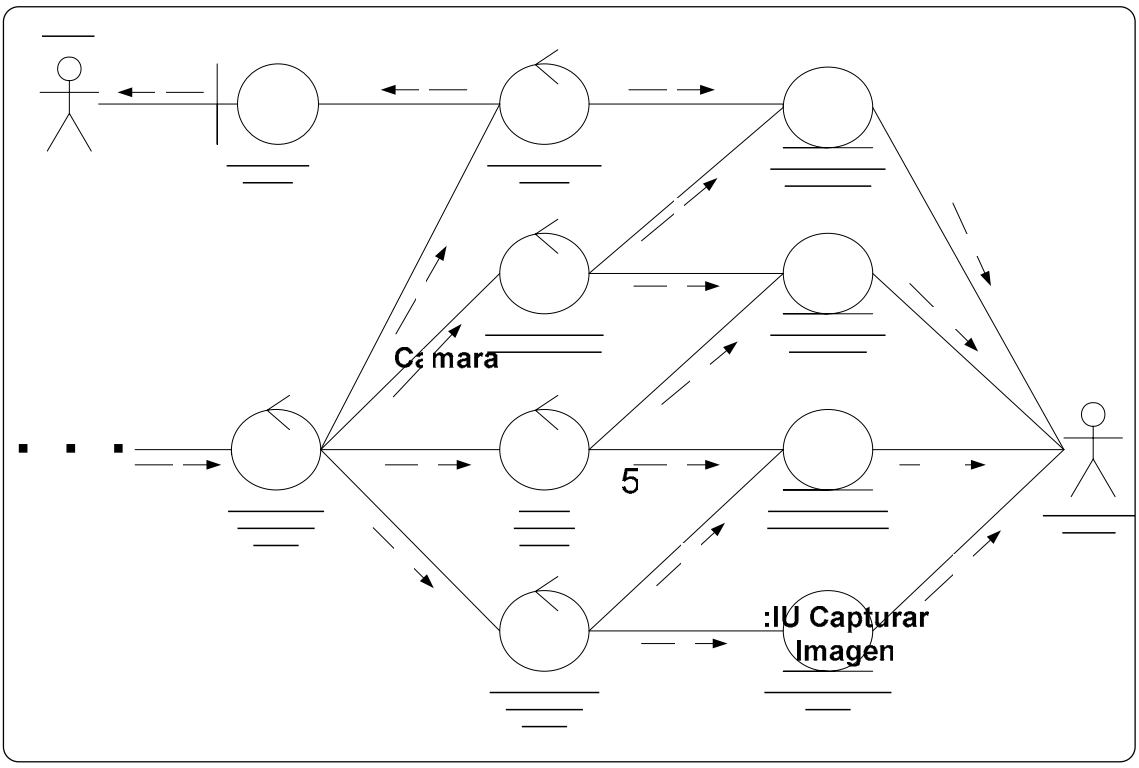


Figura 4.14: Diagrama de clases de Colaboración del Caso de Uso Generar Matriz Proyección Objeto.

Fuente: Los Autores

Flujos de trabajo

2. Activa el Gestor Matriz Proyección Objeto 3. Activa el Gestor Capturar Imagen 4. Solicita Capturar la imagen 5. Activa la cámara 6. Almacena la Imagen 7. Activa el manejador de archivos 8. Activa al Gestor Convertir Escala de Grises 9. Obtiene Imágenes Patrones Objeto Color 10. Activa el Manejador de Archivos 11. Convierte las imágenes en	13. Activa el Gestor Binarizar Imagen 14. Obtiene Imagen Patrones Objeto Gris 15. Activa el Manejador de Archivos 16. Binariza y almacena las Imágenes Patrones Objeto Binarizadas 17. Activa el Manejador de Archivos 18. Activa al gestor Generar Proyeecion Objeto 19. Obtiene Imágenes Patrones Objeto Binarizada 20. Activa Manejador de Archivos

escala de grises y almacena Imagen Patrones Objeto Gris 12. Activa al Manejador de Archivos	21. Genera y almacena la Matriz de Proyección Objeto 22. Activa el Manejador de Archivos

4.2.3 Diseño

El flujo de trabajo diseño de la fade de elaboración tiene por objetivo principal, modelar el sistema y encontrar una arquitectura de software estable que soporte los requisitos, partiendo de los resultados obtenidos en el flujo de análisis ya que proporcionan una compresión detallada de los requisitos del sistema, de manera que podamos describir la realización física de loscasoso de uso mediante el conjunto de diagramss típicos de este flujo.

4.2.3.1 Diseño de la Arquitectura

4.2.3.1.1 Diagrama de Clases de Diseño del Sistema

En las figuras 4.15 y 4.16 se ilustra el diagrama de clases y el diagrama de clases detallado del sistema.

La clase ***VentanaPrincipal*** esta clase es la que permita la interacción del usuario con el sistema, esta contiene interfaces graficas tales como botones, menús, ventanas, etiquetas, entre otras, para brindar al usuario un ambiente fácil y cómodo de manejar para interactuar con el software.

La clase ***FiltrodeObjeto,*** esta clase está asociada a la clase *VentanaPrincipal,* esta, permite que el explorador muestre solo los archivos con extensión ".OBJ", estos archivos contienen información del objeto reconstruido.

La clase ***PanelConfiguracion*** esta clase está asociada a la clase *VentanaPrincipal,* esta clase se encarga de gestionar el funcionamiento de las clases asociadas a ella, y una de las funciones más importantes en esta clase es la de generar el archivo de configuración del sistema.

La clase ***CalibrarCamara*** esta clase está asociada a la clase *PanelConfiguración*, está encargada de obtener los parámetros intrínsecos de la cámara y corregir las distorsiones en las imágenes asociadas a la lente de la cámara.

La clase ***PantallaMascara*** esta clase está asociada a la clase *PanelConfiguración*, esta, proyecta una pantalla en blanco sobre el objeto, captura una imagen de este, para que luego el usuario aplique un valor de umbral adecuado sobre la imagen capturada.

La clase ***ArchivoConfiguracion*** esta clase está asociada a la clase *PanelConfiguración*, esta, se encarga de recoger los valores de configuración de panel de configuración y dar formato al archivo para luego almacenarlo en un archivo de texto.

La clase ***VisualizadorImagen*** esta clase está asociada a la clase *VentanaPrincipal,* esta, es una ventana interna que permite al usuario visualizar el objeto a escanear y también muestra la máscara con el valor de umbral aplicado del objeto.

La clase ***RastreadorImagen*** esta clase está asociada a la clase *VisualizadorImagen,* esta clase asegura que la imagen sea cargada correctamente en el visualizador de imagen.

La clase ***GeneradorDePatrrones*** esta clase está asociada a la clase *VentanaPrincipal*, esta clase genera los patrones en código gray que son proyectados sobre el objeto y el plano.

La clase ***CapturaImagen*** esta clase está asociada a la clase *GeneradorDePatrones* y la clase *PantallaMascara*, esta encarga de de capturar y almacenar las imágenes de cada uno de los patrones generador por la clase generador de parones.

La clase ***GUI3D*** esta clase está asociada a la clase *VentanaPrincipal*, por medio de esta clase el usuario puede visualizar el objeto reconstruido.

La clase ***PanelControlVU*** esta clase está asociada a la clase *GUI3D*, y permite al usuario interactuar con el objeto reconstruido, dándole efectos como: traslación, rotación, zoom, entre otras.

La clase ***RecontruirObjeto*** esta clase está asociada a la clase *VentanaPrincipal*, esta clase es la encargada de realizar todos los procesos inherentes para la reconstrucción de la forma en 3D del objeto escaneado.

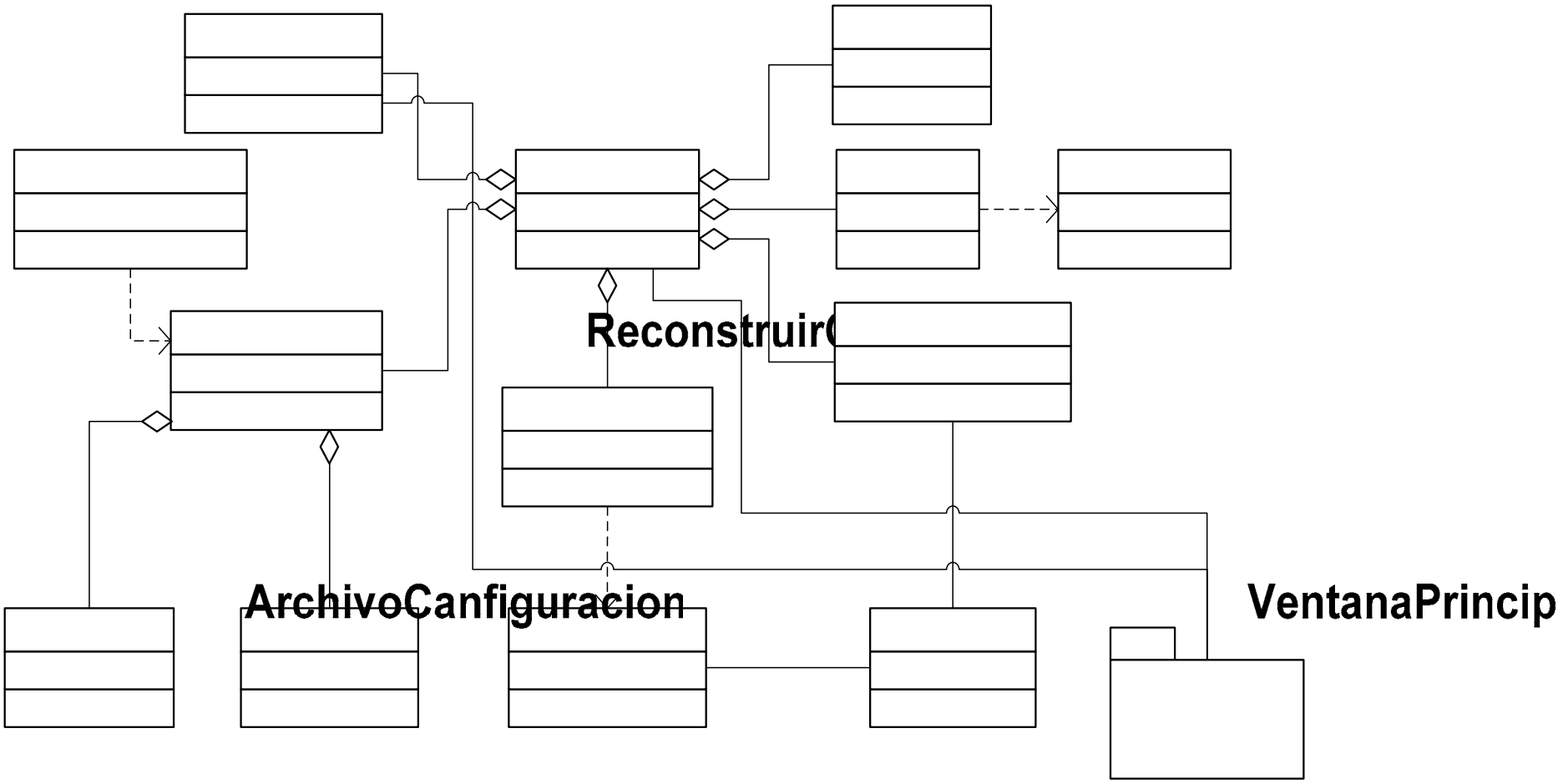


Figura 4.15: Diagrama de Clases de Diseño del Sistema.

Fuente: Los Autores.

PanelConfiguracion

VisualizadorImag

El diagrama de clases detallado del caso de uso del sistema se obervan los atributos y métodos de cada una de las clases. A continuación se explicara brevemente la utilidad de cada uno de los métodos de las clases más importantes existentes en el diagrama de clases detallado del sistema de la figura 3.15.

La clase ***ReconstruirObjeto*** tiene los métodos *sumaPatrones()* se encarga de hacer la sumatoria de las imágenes binarizadas de los patrones del objeto y el plano para en código gray del plano y el objeto. *calculoMpo()* se encarga generar una imagen de la diferencia entre la imagen de código gray de plano y el objeto llamada matriz de corrimiento. *reconstrucción3D()* utiliza la matriz de corrimiento para generar la nube de punto 3D del objeto.

La clase ***ArchivoConfiguracion*** contiene los métodos *ArchivoConfuguracion():* constructor de la clase, *cargarArchivosParametrosSistema():* da formato y almacena los parámetros de configuración, *leerArchivosParametrosSistema():* leer los parámetros del archivo de configuración y se los entrega al sistema, *lerArchivosParametrosCamara():* lee los datos del archivo de parámetros intrínsecos de la cámara y los entrega al sistema.

La clase C***alibrarCamara*** contiene los métodos: *CalibrarCamara():* constructor de la clase, *capturaFotos():* se encarga de capturar y almacenar las imágenes de los patrones de calibración, *obtenerParametrosCamara():* utiliza las imágenes capturadas por el método *capturaFotos():* y obtener los parámetros intrínsecos de la cámara y los almacena en el archivo de parametrosIntrinceco, *normalizarFotos():* lee los datos del archivo de parametrosIntrincecos y elimina la distorsión en las imágenes causadas por la lente de la cámara, *iniciarCuenta() y detenerCuenta():* se encarga de llevar el control de la barra de progreso que se tarda el algoritmo en obtener los parámetros de calibración.

La clase ***PantallaMascara()*** contiene los métodos *PantallaMascara()* constructor de la clase, *pintar():* se encardo de proyectar la pantalla en blanco para capturar la imagen de la máscara.

La clase ***PanelConfiguracion()*** contiene los métodos: *EsNumero(), EsNumeroFloat():* se encarga de validar los campos numéricos en el panel de configuración, *cargarDatosAlArchivo() y cargarDatosDesdeArchivo():* utiliza los métodos de la clase *ArchivoConfiguracion().*

La clase ***GeneradorDePatrones*** contiene los métodos: *GeneradorDePatrones:* constructor de la clase, *pintar():* se encarga de generar los patrones en código gray.

La clase ***CapturaImagen*** contiene los métodos *capturadorPatron():* captura los patrones generador por la clase *GeneradorDePatrones, capturaMascara()* captura imagen generada por la clase *PantallaMascara.*

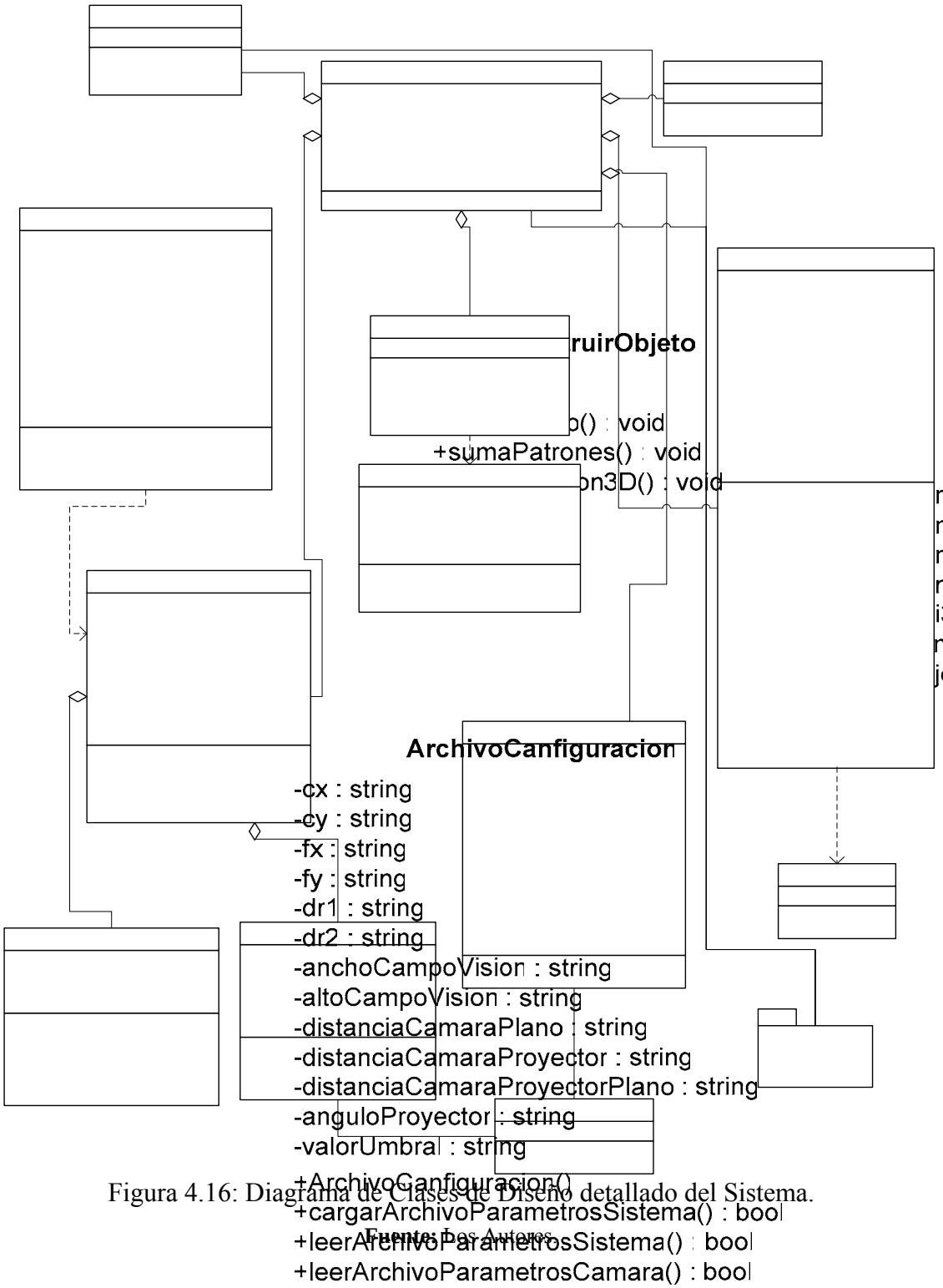


Figura 4.16: Diagrama de Clases de Diseño detallado del Sistema.

Fuente: Los Autores

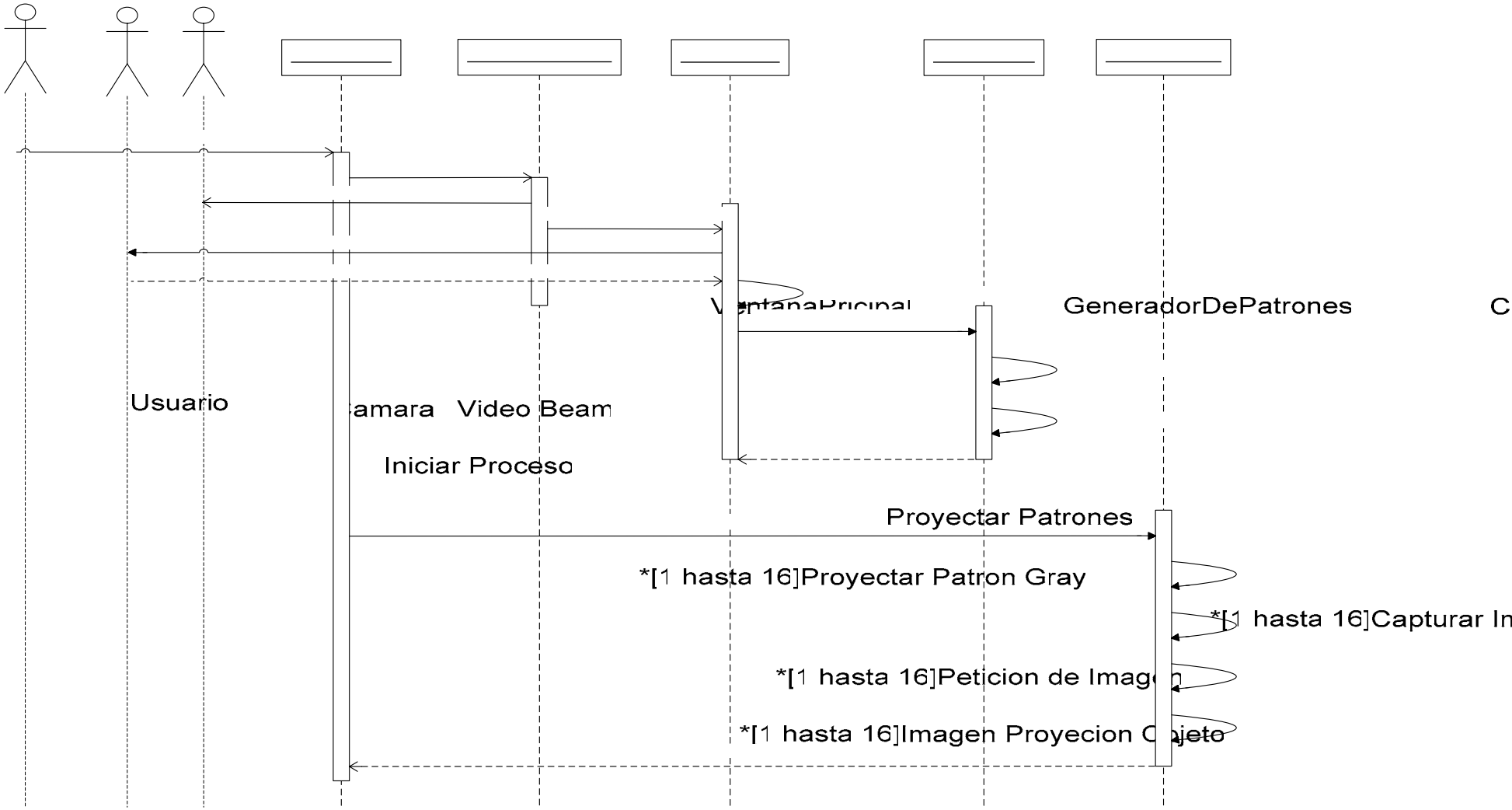
4.2.3.2 Diagrama de Secuencia del Caso de Uso Procesar Reconstrucción 3D

En el diagrama de secuencia se muestra cronológicamente el paso de mensajes entre los objetos de las clases de diseño, se observan las instancias de los actores y la interacción de estos con los objetos de diseño.

En la figura 4.17 se ilustra el diagrama de secuencia correspondiente al caso de uso Procesar Reconstrucción 3D. se observa que el usuario es quien inicia el proceso al solicitarle a *VentaPrincipal* el inicio del proceso, este a su vez le solicita a *GeneradorDePatrones* proyectar patrones.

GeneradorDePatrones solicita proyectar patrón gray al Video Beam, luego se le solicita capturar imagen a *CapturarImagen,* este hace la petición de captura a la cámara, la cámara responde con imagen proyección objeto, *CapturarImagen* almacena las imagen, luego se le solicita eliminar distorcion a *CalibrarCamara* esta elimina la distorcion con el procedimiento *normalizarFotos(),* almacena la imagen sin distorsión, y responde con distorsión eliminada. Este proceso se realiza de en un ciclo de dieciséis veces.

Una vez almacenadas y calibradas las dieciséis de los patrones con el objeto *VentanaPrincipal* hace una petición a *ReconstruirObjeto*, este, ejecuta el procedimiento *sumaDePatrones(), calculoMpo() y reconstruccion3D()*, que permiten generar el archivo del objeto reconstruido que posteriormente será almacenado.



Fi

gura 4.17: Diagrama de Secuencia del Caso de Uso Procesar Reconstrucción 3D.

Fuente: Los Autores.

4.2.3.3 Diagrama de Secuencia del Caso de Uso Configurar Mascara.

En la figura 4.18 se ilustra el diagrama de secuencia correspondiente al caso de uso Configurar Mascara.

Se observa que el usuario es quien inicia el proceso al solicitarle a *VentaPrincipal* mostrar el Panel de Configuración,

El usuario envía la orden de capturar imagen al *PanelConfiguracion,* este pide a *PantallaMascara* proyectar una imagen en blanco y esta a su vez envía la orden al viedo beam de proyectar la pantalla en blanco.

CapturarImagen activa la cámara y capta la imagen, luego el usuario introduce un valor de umbral y aplica el valor, se le envía una señal a PantallaMascara para que binarice la imagen, esta esta aplica el valor de umbral introducido por el usario y le indica *PanelConfiguracion* que la imagen ha sido binarizada y esta la muestra.

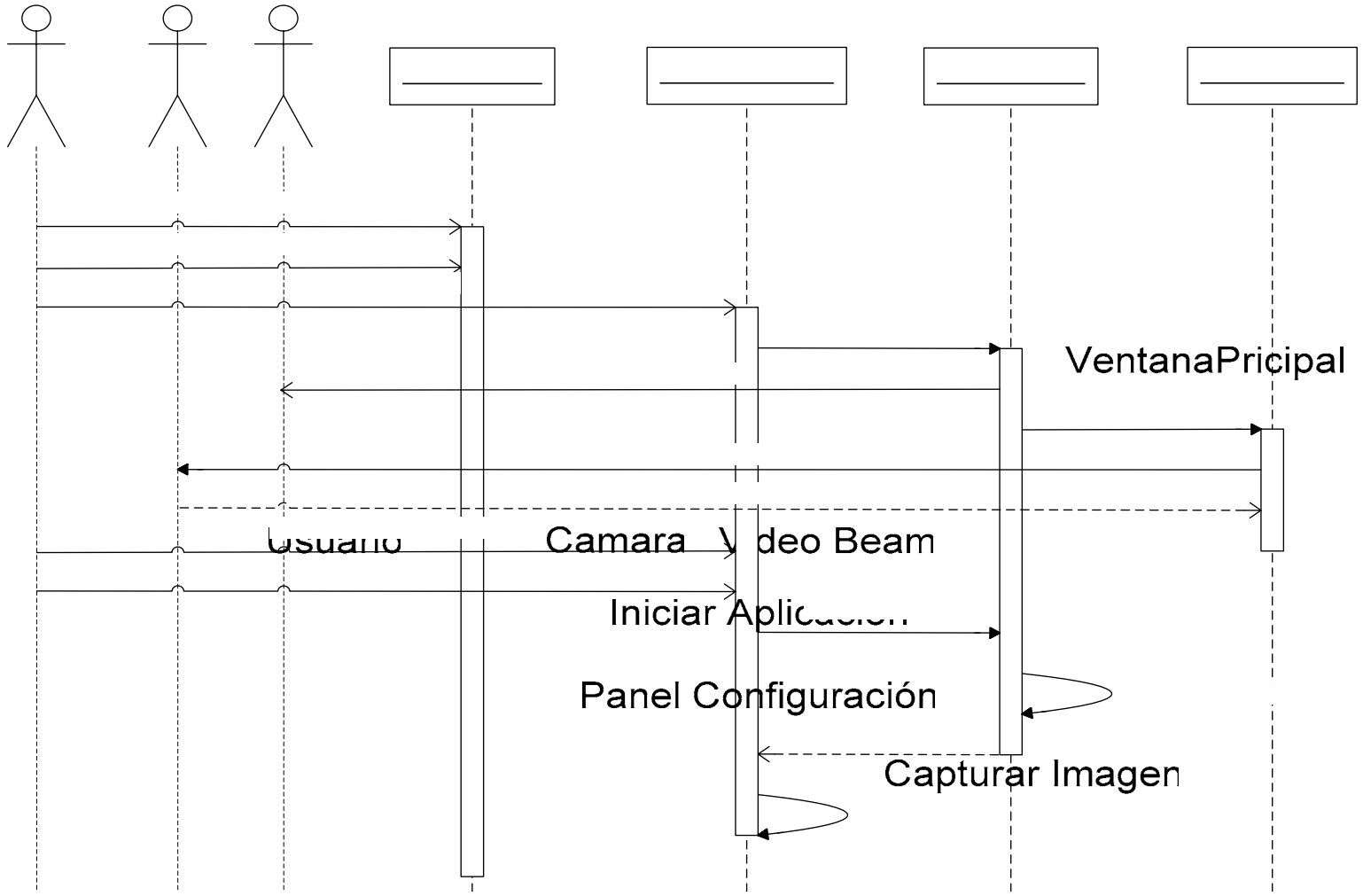


Figura 4.18: Diagrama de secuencia del caso de uso Configurar Mascara.

Fuente: Los Autores.

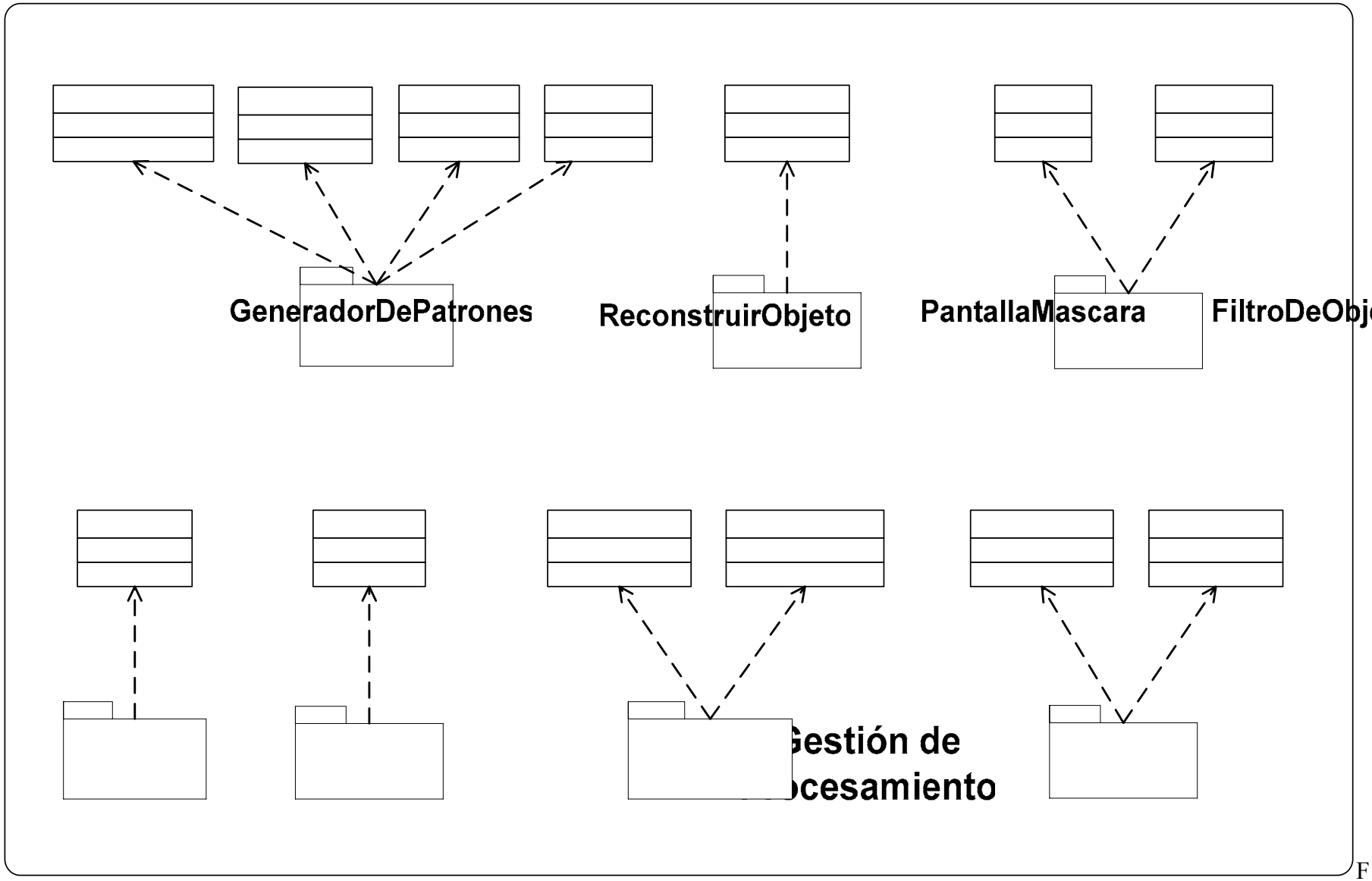
4.2.3.4 Identificación de Paquetes de Diseño

En la figura 4.19 se muestra el diagrama de paquetes del sistema correspondiente a la fase de elaboración, en estafase se añadieron nuevos casos de uso a este diagrama, estos casos de uso son: *Procesar Parámetros Configuración, Generar Matriz Proyección Plano, Generar Matriz Proyección Objeto, Calculo Matriz Corrimiento, Convertir a Escala de Gris, Binarizar Imagen, Configurar Mascara, Segmentación, Visor GUI3D, Control GUI3D.*

El sistema identifica sus paquetes a partir del modelo de Casos de Uso. Los casos de uso: *Procesar Parámetros Configuración, Generar Matriz Proyección Plano, Generar Matriz Proyección Objeto, Calculo Matriz Corrimiento,* están agrupados en un mismo paquete de análisis que se denomina *Gestión de Procesamiento*, por estar implicados en un mismo proceso de negocios. Este paquete contiene los casos de uso encargados de llevar a cabo todo el procesamiento para generar la nube de puntos del objeto a reconstruir.

Los casos de uso: *Convertir a Escala de Gris, Binarizar Imagen, Configurar Mascara, Segmentación,* se encuentran implicados en un mismo proceso de negocios y se encuentran agrupados en un mismo paquete de análisis llamado *Gestión de Procesamiento de Imagen.* Son los encargados de llevar a cabo las modificaciones en las imágenes necesarias para poder ser procesadas por los caso de uso del paquete *Gestión de Procesamiento.*

Los casos de uso: *Visor GUI3D, Control GUI3,* se encuentran implicados en un mismo proceso de negocios y están agrupados en e un mismo paquete llamado *Gestión de Procesamiento 3,* estos casos de uso están encargados de la visualización y de interpretar los movimientos del objeto 3D que el usuario realice sobre el mismo.



F

Figura 4.19: Nuevo Diagrama de Paquetes del Sistema.

Fuente: Los Autores

4.2.3.4.1 Identificación del Subsistema de Aplicación

Los subsistemas constituyen un medio para organizar el modelo de diseño en piezas manejables, estos subsistemas son: *capa especifica de aplicación, capa general de la aplicación, capa intermedia, capa de software del sistema.*

En esta etapa se identifican los subsistemas de la capa especifica de la aplicación y de la capa general de la aplicación.

En la capa específica de aplicación se identifico el paquete principal que representa todo el sistema y además contiene todos los subsistemas e interfases necesarias para el funcionamiento del sistema. Este paquete es el encargado de permitir la iteración del usuario con el sistema.

La capa general de aplicaciones contiene todos aquellos paquetes que ilustran la funcionalidad del sistema. Entre ellas tenemos: *Gestión de Procesamiento, Gestión de Procesamiento de Imágenes, Gestión de Procesamiento 3D, entre otras.*

La capa especifica y general de la aplicación se muestra en detalle en la figura 4.18.

4.2.3.4.2 Identificación de los Subsistemas intermedios y de Software del Sistema

El software del sistema y la capa intermedia establece las raíces de un sistema computacional ya que todas las funcionalidad descansa sobre software como sistemas operativos *(Windows, Linux),* intérpretes como la máquina virtual de java (JVM) y compiladores como C y C++.

En la figura 4.20 se muestra el Diagrama de Capas del Sistema.

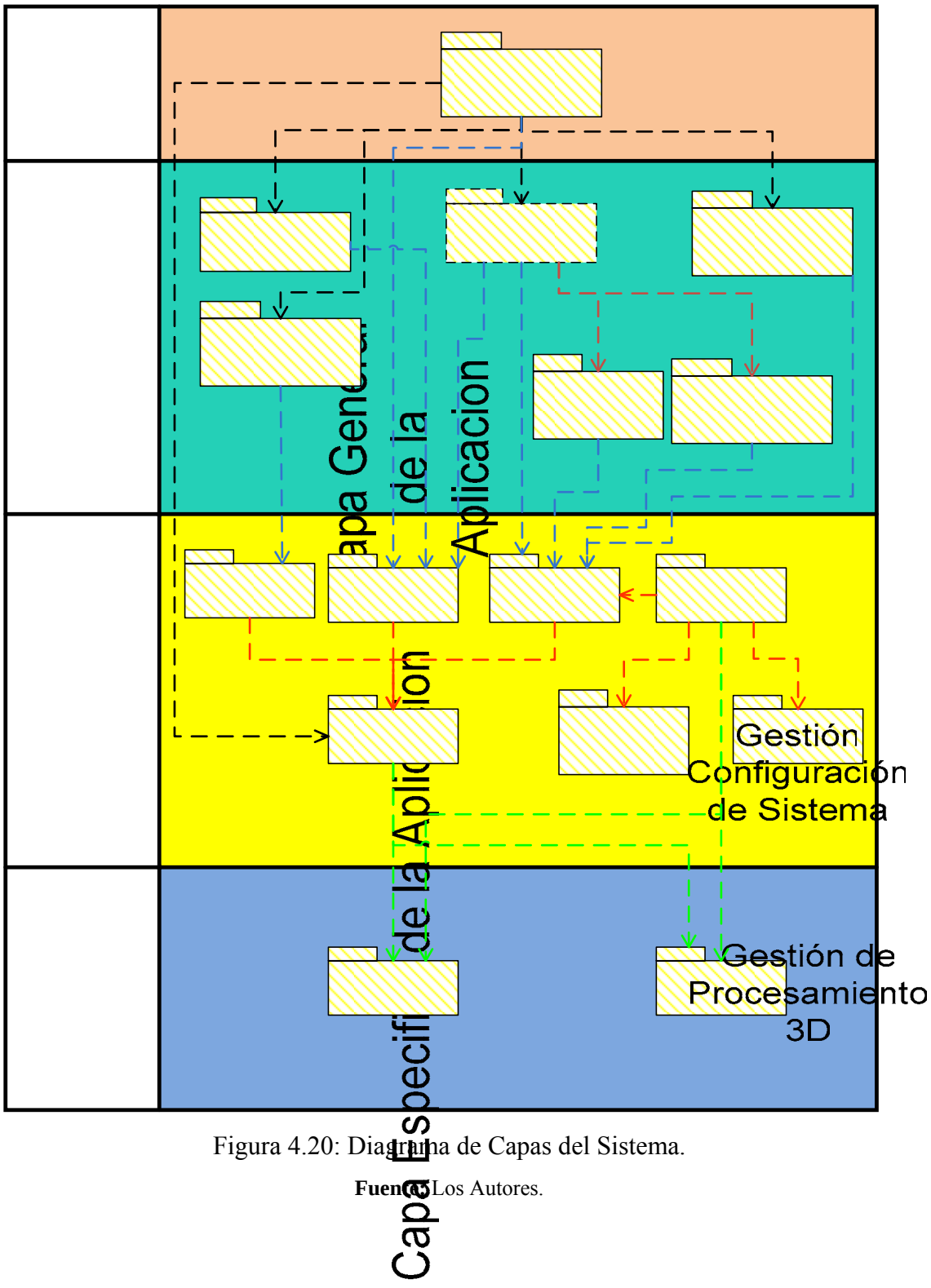


Figura 4.20: Diagrama de Capas del Sistema.

Fuente: Los Autores.

4.3 IMPLEMANTACION

Este flujo de trabajo se muestra los componentes de software que ya han sido programados hasta esta fase, hay que recalcar que esto solo es una pequeña parte del sistema que ha sido desarrollada hasta ahora, en el capitulo siguiente se muestra la estructura del sistema completa y se detallan cada uno de los componentes pertenecientes a este.

El la figura 4.21 se muestra el Diagrama de Componentes parcial desarrollado hasta esta fase.

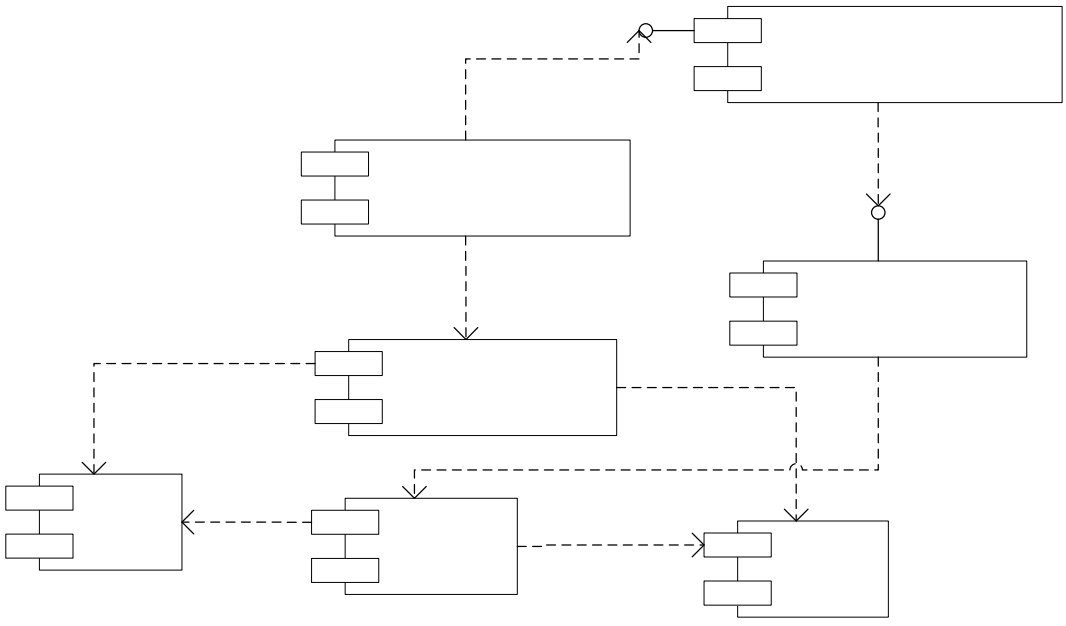


Figura 4.21: Diagrama de Componentes parcial, Fase de Elaboración.

Fuente: Los Autores.

4.4 RESULTADOS OBTENIDOS EN LA ITERACION I

Durante la fase de elaboración se aplicaron 3 flujos de trabajo: Requisitos, Análisis y Diseño.

El flujo de requisitos permitió la recopilación de los requisitos faltantes en la fase de inicio, pudiendo representar como nuevos casos de uso que se añadieron al modelo de casos de uso de esta fase, fortaleciendo la arquitectura de software del sistema.

Durante el flujo de análisis se estudiaron los nuevos casos de uso para obtener los diagramas de análisis que sirvieron de entrada al flujo de diseño para establecer las clases reales que serán utilizadas en la fase de construcción que serán utilizadas en la fase de construcción del prototipo de software.

El diagrama de secuencia del caso de uso Procesar reconstrucción 3D permitió visualizar detallada y cronológicamente los procesos que allí se llevan a cabo así como las clases utilizadas, los objetos instanciados y los métodos que allí se manejan.

Por otra parte, el diagrama de capas realizado en esta fase permitió organizar los diferentes paquetes de acuerdo a su función, con la finalidad de hacer un modelo diseñado en piezas manejables.

4.5 CONCLUSIONES

Durante la fase de elaboración se llevó a cabo la recopilación de la mayor parte de los requisitos. Esto ayudo a establecer una arquitectura sólida para el sistema. Esta arquitectura se obtuvo a través de un profundo análisis y diseño de los requisitos.

El análisis permitió obtener los diferentes modelos de análisis entre los cuales se encuentran: el diagrama de clases que representa el comportamiento de los casos de uso, el diagrama de colaboración que es más específico que el anterior, ya que el expresa las interacciones entre sus clases a través de los flujos de trabajo.

En el diseño se llevó a cabo la realización de los diagramas de clases, diagramas de secuencias y diagrama de capas, los cuales representan el funcionamiento e interacción entre las clases y paquetes.

Para finalizar es importante destacar que se lograron los objetivos planteados en esta fase y que se tiene como resultado una arquitectura sólida, que será la entrada principal para la fase de construcción.

CAPITULO V
FASE DE CONSTRUCCION

5.1 INTRODUCCION

La fase de construcción es la fase donde se implementa el sistema haciendo énfasis en la acumulación del conocimiento que se obtuvo en las fases anteriores. El objetivo de la fase de construcción, es desarrollar un producto software en su versión operativa inicial o versión "Beta", faltando solo las pruebas para ser una versión estable.

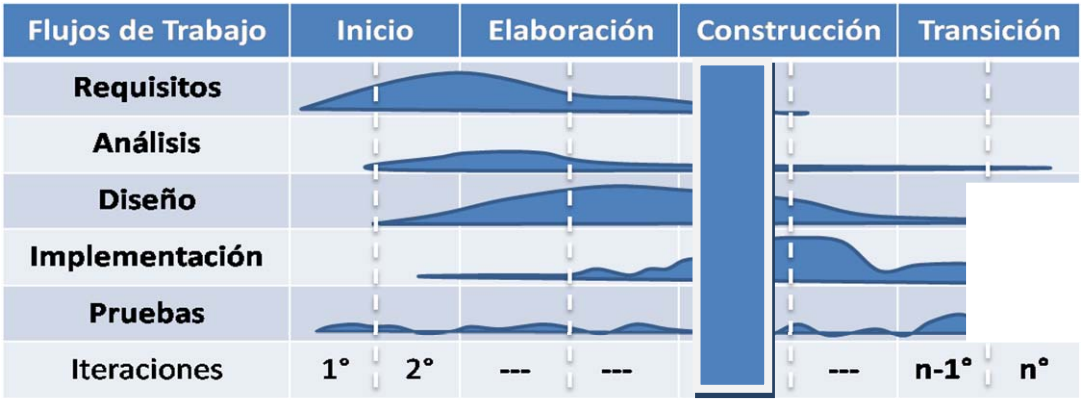


Figura 5.1: Flujo de trabajo durante la Fase de Construcción.

5.2 ESCOGENCIA DEL TIPO DE DATO

La entrada al sistema es una imagen capturada a través de una cámara y esta constituida como una matriz donde cada una de sus posiciones representa un punto en la imagen llamado pixel.

La matriz que contiene la imagen representa el tipo de dato fundamental de entrada al sistema, esta es almacenada en el formato de imagen "BMP" ya que es uno de los formatos de imágenes digitales que permite el estudio de la misma como matrices sin pérdida de información.

5.3 CALIBRACION DE LA CAMARA [19]

Las cámaras o sensores de imagen son los elementos encargados de captar la información luminosa de la escena y transmitirla al computador como una señal analógica o digital. Una vez que la imagen ha sido recogida por la cámara de video, la tarjeta de adquisición y procesamiento de imágenes recibe la señal analógica enviada por ella, para convertirla en una señal digital que será procesada.

El proceso de calibración de una cámara es un paso necesario para obtener medidas de la escena a partir de imágenes de la misma. La exactitud de la calibración determinará posteriormente la precisión de las medidas que se realicen a partir de las imágenes. Es por este motivo que es imprescindible realizar la calibración de la cámara con plenas garantías de que los parámetros obtenidos son los más parecidos a los reales. Este compromiso implica tanto la elección del método de calibración así como la correcta utilización del mismo.

5.3.1 Escogencia del Método de Calibración de Cámara

Existen diferentes métodos de calibración de cámara como también diferentes formas de clasificarlos; en este apartado se referencian algunos de los métodos más utilizados en la actualidad, entre estos estan:

- Método de calibración de Tsai

- Auto Calibración
- Método de calibración de Zhang

5.3.1.1 Método de calibración de Tsai

El método de Tsai, representa un proceso clásico de calibración basado en las medidas de las coordenadas de los puntos de una plantilla 3D respecto a un punto de referencia fijo. Este muestra excelentes resultados, siendo necesaria una sola imagen de la plantilla de calibración, a pesar de que para conseguirlos es necesaria una precisión importante de los datos de entrada.

El método de Tsai obtiene una estimación precisa de los parámetros de la cámara si los datos de entrada están poco contaminados con ruido. Teniendo en cuenta que son necesarios al menos cien puntos en la plantilla y que las coordenadas han de estar referidas a un origen de coordenadas fijo, además la orientación y posición de la cámara con referencia a unos ejes de coordenadas, estos parámetros son: Rx, Ry, Rz, que representan los ángulos de rotación para la transformada entre los ejes del mundo y de la cámara, Tx, Ty, Tz. Que son los componentes del vector de traslación para la transformada entre los ejes de cámara y el mundo. Adicionalmente, se necesitan cinco parámetros intrínsecos relacionados con constantes de la cámara que son conocidos, pues sólo basta con buscar en la información que el fabricante da con la cámara debe suministrar. Estos otros parámetros constantes son: Ncx. número de elementos sensores en dirección horizontal, Nfx. número de pixeles en la dirección horizontal de la cámara, dx. ancho de cada elemento del sensor de la cámara (mm/el), dy. alto de cada elemento del sensor de la cámara (mm/el), dpx.- ancho efectivo de un pixel en la cámara (mm/pix), dpy. alto efectivo de un pixel en la cámara (mm/pix).

La plantilla 3D consiste dos o tres planos ortogonales entres si, lo que conlleva a una realización laboriosa y costosa, teniendo en cuanta que es imprescindible un adecuado diseño de la plantilla de calibración además de una medida exacta de las coordenadas de los puntos, para así obtener buenos resultados al final del procedimiento. A pesar de todo, las posibilidades de cometer errores en las medidas son altas.

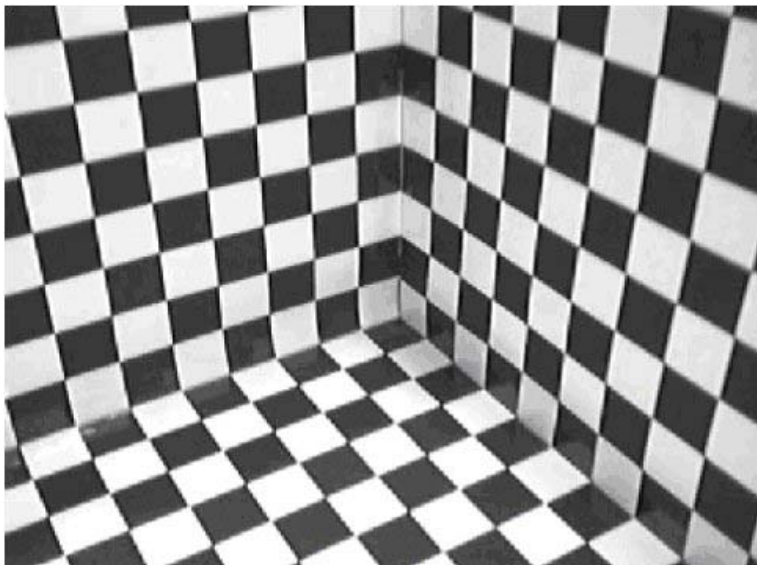


Figura 5.2: Plantilla de Calibración 3D.

5.3.1.2 Auto Calibración

Estas son técnicas que no utilizan ningún objeto de calibración, ya que sólo es necesario establecer correspondencias de un punto en diferentes imágenes. Solamente moviendo la cámara en una escena estática, la rigidez de la escena provoca en general dos restricciones dentro de los parámetros intrínsecos de la cámara Por lo tanto con varias imágenes de la misma escena tomadas con los mismos parámetros intrínsecos, las correspondencias entre tres imágenes son suficientes para calcular tanto los parámetros intrínsecos como extrínsecos. En estos casos aunque no es necesaria una

plantilla, es necesario calcular un gran número de parámetros, resultando en un problema matemático bastante complejo. Debido a la dificultad para iniciar la búsqueda los métodos de auto calibración tienden a ser inestables. También es necesario tener en cuenta la familia de algoritmos que calculan los parámetros que modelan la distorsión producida por las lentes en la imagen sin el uso de objetos de calibración y por lo tanto sin conocer ninguna estructura 3D. Estos métodos se basan en que dentro de una proyección perspectiva ideal, la cámara transforma líneas rectas en el espacio 3D en líneas rectas dentro del espacio 2D correspondiente a la imagen. Por lo tanto reforzando la linealidad de las partes de la imagen que se ven curvas debido a la distorsión de la lente de la cámara, es posible estimar la deformación que ésta produce. Hay métodos que utilizan restricciones epipolares y trilineales entre pares y tripletes de imágenes respectivamente para estimar la distorsión radial.

5.3.1.3 Método de calibración de Zhang

Zhang propone una técnica de calibración basada en la observación de una plantilla plana desde varias posiciones. Este método utiliza las coordenadas de los puntos situados una plantilla plana 2D tomando diferentes imágenes de la misma desde diferentes posiciones y orientaciones. De esta forma se combinan las ventajas de los métodos de calibración basados en las medidas de las coordenadas de la plantilla con las ventajas de la auto calibración en la cual no es necesaria utilizar plantilla

Este modo de calibración resulta muy flexible desde el punto de vista de que tanto la cámara como la plantilla puede ser movida libremente y además se pueden tomar tantas imágenes como se quieran sin tener que realizar medidas en la plantilla

La plantilla 2D no requiere tal especial diseño de la plantilla, ni tampoco una medición tan exacta de los puntos de la misma. Además, la sensibilidad del algoritmo de calibración frente a errores en las medidas, puede ser mejorada aumentando el

número de puntos en la plantilla, simplemente imprimiendo un tablero de ajedrez que con más esquinas.

Por las razones antes expuestas, y tomando en cuenta que en las librerías de tratamiento de imágenes OPENCV existen métodos para la calibración de cámaras basados en este método se decidió utilizar el método de calibración de cámaras de Zhang para este trabajo de grado

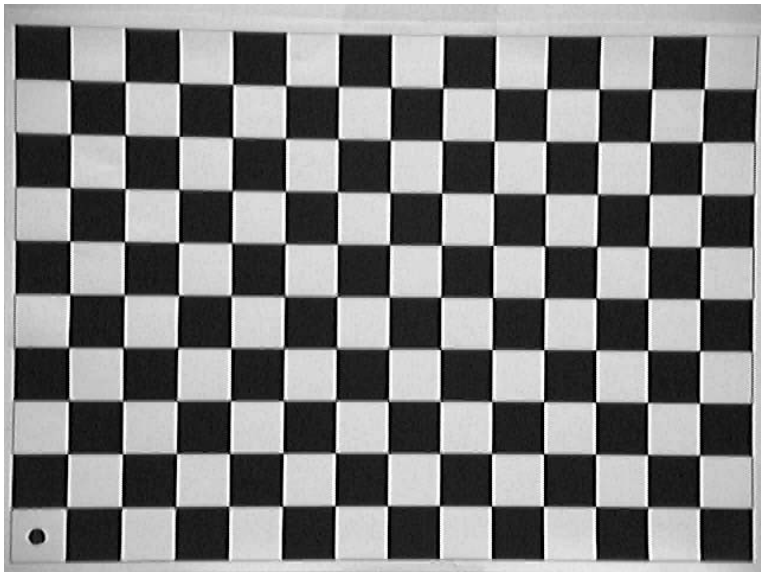


Figura 5.3: Plantilla de Calibración Bidimensional.

5.4 ESCOGENCIA DEL METODO DE RECONSTRUCCION 3D

Los métodos ópticos tradicionales de reconstrucción 3D han despertado especial interés en aplicaciones industriales y médicas, debido a sus características no invasivas, bajo costo de implementación y excelentes resoluciones. Se suelen clasificar en pasivos o activos. En lo métodos pasivos no es necesario controlar la fuente de iluminación, pero el tratamiento digital para obtener información de

profundidad es muy delicado y se requiere de un alto esfuerzo computacional. En los métodos activos el uso de un patrón de radiación simplifica el problema de la medida de la topografía.

Al pensar en aplicar cierta técnica de reconstrucción se debe tener en cuenta las características reales de los objetos a estudiar, tales como: grado de absorción luminosa y tamaño. También, existen objetos que tienen variaciones abruptas de altura (de casi 90^{o}) y variaciones fuertes de la pendiente de la superficie, estas características representan un problema en el proceso de reconstrucción para algunas técnicas ópticas, desde este punto de vista pueden llamarse objetos complicados o discontinuos. Para objetos conplicados discontinuos, es común utilizar los métodos convencionales de proyección de frajas y triangulación laser.

La figura 5.4 muestra el esquema clásico del método de proyección de franjas derivado de la técnica de triangulación láser.

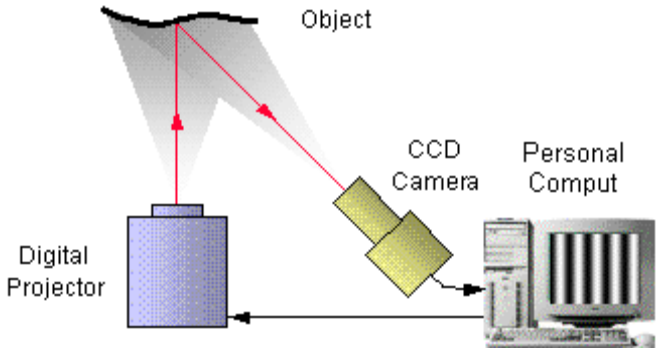


Figura 5.4: Montaje de un sistema de proyección de franjas.

Un sistema de proyección genera un sistema de franjas sobre la superficie del cuerpo. La cámara CCD, ubicada perpendicularmente al plano de referencia, adquiere la imagen del sistema de franjas deformado por la topografía del cuerpo. Así, la altura

para cada punto digitalizado de la superficie del cuerpo es codificada en deformación local del paso y orientación de las franjas proyectadas. Matemáticamente la imagen deformada de las franjas se puede escribir como:

$$I(x,y) = I_0(x,y) + M(x,y) * cos[2\pi\, f_0\, X + \phi(x,y)] \qquad (1)$$

Donde el término de interés es la fase φ o argumento de la función coseno que representa las variaciones del paso y orientación de las franjas. De esta manera, para cada punto de la imagen digitalizada existe un valor de la fase y una altura del cuerpo. Los parámetros del montaje y el fenómeno de codificación producen una relación inversa que permite obtener Z a partir de $\phi(x,y)$. Realizando el tratamiento digital necesario al sistema de franjas se puede calcular ϕ y obtener $Z(x,y)$. Para objetos discontinuos es imposible saber cuantas veces se corrieron las franjas debido a la altura del cuerpo, por esta razón no se puede determinar el valor de fase absoluto para este tipo de objetos con un sistema de franjas. La figura 2 muestra la imagen de un sistema de franjas proyectado sobre un objeto discontinuo, la marca amarilla señala una de las zonas en las que no se pude recuperar la fase absoluta y por consiguiente no se puede establecer el valor de altura.

Existen varias alternativas para extraer la fase de un sistema de franjas entre las que se destacan las técnicas de *Transformación de Fourier (**TF**) y Corrimiento de Fase*. En el primer caso, realizando la **TF** a la ecuación (1) y debido a la influencia de la portadora espacial f_0 se identifican tres picos separados, uno de estos picos posee información de $\phi(x,y)$. el cual se puede aislar y calcular. Sin embargo, este proceso

de demodulación de la fase de un punto se ve afectado por la información de los puntos vecinos debido a la influencia de la TF. Regiones donde no hay franjas afectan las regiones útiles. Este problema se presenta típicamente en objetos con sombras como se observa en la marca verde de la figura 5.5 en donde no hay franjas debido a la sombra producida por la topografía del cuerpo como consecuencia del tipo de iluminación. También se presentan inconvenientes con franjas de bajo contraste y de frecuencia alta, en las cuales no se cumple el teorema del muestreo.

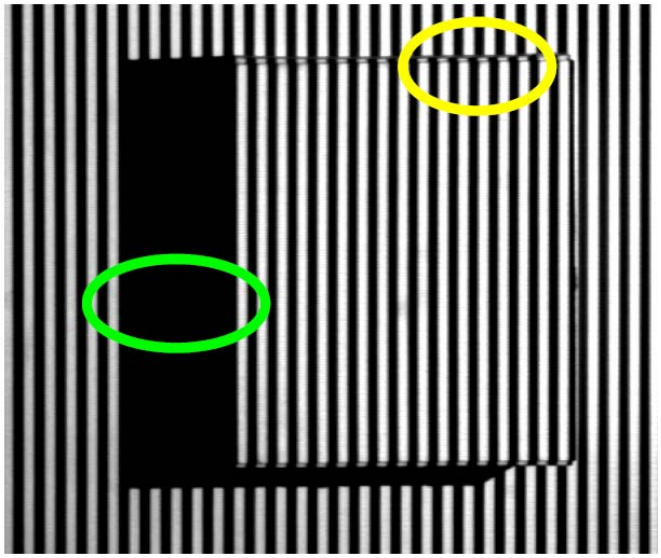


Figura 5.5: Sistema de franjas deformado por un cubo.

Por otro lado, en el método de corrimiento de fase, la fase se calcula utilizando varios sistemas de franjas deformadas por el cuerpo pero desplazadas en fase. Introduciendo un corrimiento constante al sistema de franjas deformado y utilizando las propiedades de las funciones senosoidales se calcula la fase para cada punto, independientemente de la influencia de los vecinos. Introducir un corrimiento de fase constante implica la utilización de un equipo especializado para desplazar las franjas homogéneamente con respecto al cuerpo. La figura 5.6 muestra los sistemas

de franjas deformados por un objeto con un corrimiento de fase de 0, $\frac{2\pi}{3}$, $\frac{4\pi}{3}$ respectivamente.

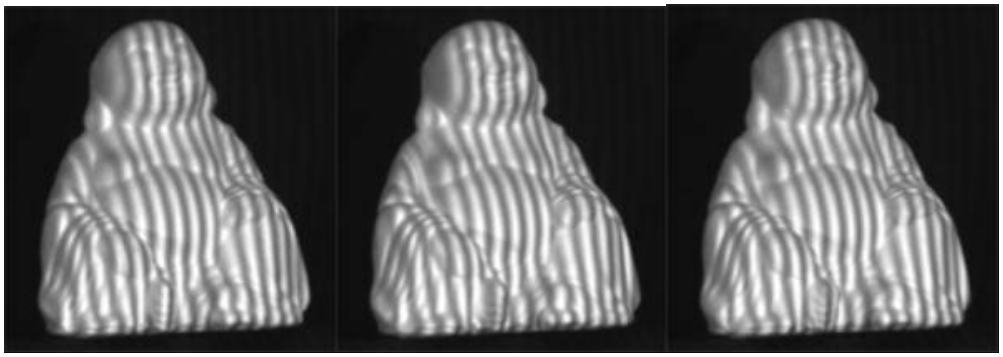


Figura 5.6: Corrimiento de fase en 3 pasos.

En el método de triangulación láser se proyecta un patrón láser (punto, línea, etc.) formando un ángulo conocido con respecto al objeto a iluminar y con respecto a la cámara CCD, como se observa en la figura 4. Midiendo la deformación geométrica del patrón láser con respecto al patrón proyectado sobre el plano de referencia donde se ubica el cuerpo, se puede obtener la altura del objeto. Para el desarrollo de este método es necesario hacer un barrido unidimensional o bidimensional al cuerpo según se proyecte una línea o un punto, respectivamente. En este caso no hay problema con objetos discontinuos, pero realizar el procedimiento de barrido requiere de equipo especializado y experimentalmente es un problema a la hora de trabajar con objetos a gran escala.

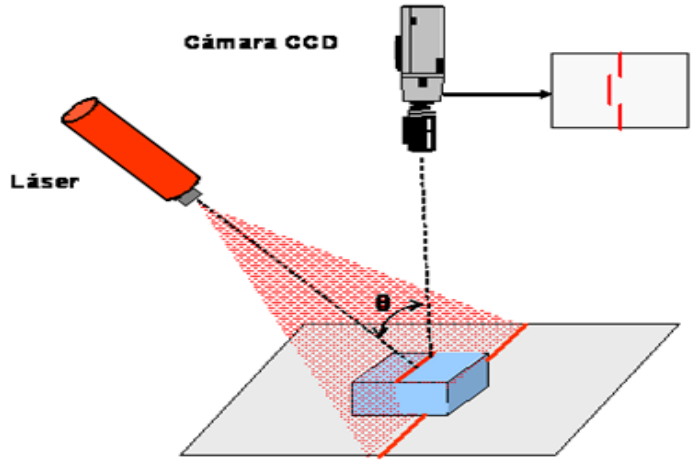


Figura 5.6: Montaje de un sistema de triangulación láser.

Sabiendo que en los sistemas de reconstrucción 3D utilizados en ingeniería y medicina se encuentran típicamente objetos macroscópicos discontinuos definidos anteriormente, es necesario adaptar o implementar técnicas de reconstrucción 3D para este tipo de objetos. Una alternativa de solución consiste en utilizar un método que agrupe las características importantes de la triangulación láser (resolver discontinuidades) y proyección de franjas (análisis bidimensional sin necesidad de barrido y de fácil implementación, el método de código de gris cumple con estas características.

El método de código de gris se basa en la proyección de n patrones de franjas de diferente frecuencia espacial, que al agruparlos permiten describir 2^n diferentes direcciones de proyección, donde cada dirección corresponde a una palabra en código binario. De esta manera, a cada punto en el espacio del plano de referencia le corresponde un código binario único que puede ser identificado en la imagen digitalizada por la CCD. Así, la topografía del objeto introduce un corrimiento lateral de las palabras con respecto al plano de referencia. Un procedimiento digital que permita identificar cada código antes y después de ubicar el objeto, permitirá calcular

el corrimiento lateral y determinar la altura correspondiente. De esta manera, al proyectar patrones 2D no se requiere de un barrido y al calcular corrimientos debido a la topografía del objeto resuelve discontinuidades

La técnica de código de gris es de fácil implementación y es flexible al tamaño del objeto. Sin embargo, presenta inconvenientes con imágenes de bajo contraste y está limitada para escenas estáticas, ya que es necesaria más de una toma para hacer la reconstrucción 3D.

5.5 RECONSTRUCCIÓN 3D UTILIZANDO MÉTODO DE CODIGO GRAY

5.5.1 Sistema Óptico

El montaje óptico que se requiere para el desarrollo de la técnica de código de gris consta de un sistema de proyección que genera una secuencia de patrones binarios sobre la superficie del objeto, y un sistema de adquisición para registrar las imágenes de los patrones proyectados sobre el plano de referencia antes y después de ubicar el objeto. Este montaje se ilustra en la figura 5.7, en donde los puntos P y C representan las pupilas de salida del proyector y de la cámara respectivamente, d es la distancia entre ellas, L es la distancia al plano de referencia, medida en dirección Z del sistema de coordenadas ortogonal (X,Y,Z),(ver figura 5.7). El eje óptico de la cámara es perpendicular al plano de referencia y el eje del proyector forma un ángulo con respecto a este, estos ejes interceptan al plano R en el punto O. FW es el ancho del campo de observación a lo largo de la coordenada X sobre el plano de referencia.

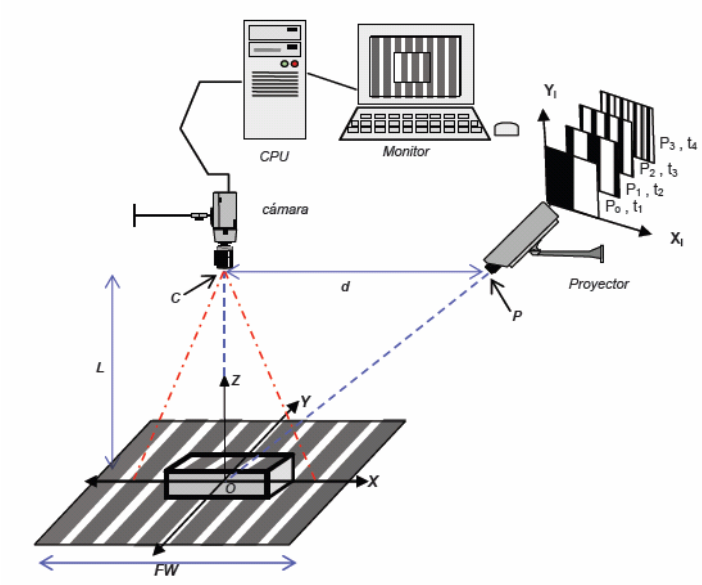


Figura 5.7: Montaje experimental del método de código de gris.

El método de código de gris ejecuta una codificación en el espacio de medida, basada en la proyección, en n tiempos diferentes, de n patrones binarios bidimensionales, formados por franjas blancas y negras de diferente ancho para cada patrón.

A fin de entender mejor la codificación del espacio, en la figura 5.8 se muestra un ejemplo para 4 patrones, ubicados en un sistema de coordenadas (X_i,Y_i, t) donde X_i, Y_i corresponden a las coordenadas del plano objeto del sistema óptico de proyección y la coordenada "t", es el tiempo en el que se proyecta cada patrón. A las franjas blancas se les asigna el valor lógico "1" y a las negras el valor lógico "0". De esta manera, al escoger el valor correspondiente a una misma posición (a, b) del plano X_i, Y_i para cada patrón, se forma un número binario en la dirección t, llamado también "*palabra binaria*", que se repite a lo largo de la coordenada Y. Al plano (Y_i, t) que tiene la misma palabra se le llama "*dirección de proyección*". De esta manera, cada posición X_i posee una dirección de proyección diferente. Así, para la

posición $\mathbf{X_i=a}$ en la figura 5.8, la dirección de proyección se puede identificar con el valor decimal (10) de la palabra binaria (1 0 1 0). En el ejemplo indicado en la figura se proyectan 4 patrones obteniéndose $2^4=16$ palabras binarias que producen 16 direcciones de proyección codificando todo el espacio X_i. Las palabras binarias se visualizan fácilmente en niveles de gris como se muestra en la parte superior de la figura.

En general, si se proyectan n patrones, al agruparlos en la dirección t se pueden describir 2^n diferentes direcciones de proyección, donde cada dirección está asociada a una palabra binaria. Los patrones se construyen utilizando la estructura de la lógica matemática donde, a partir de la proyección de n patrones "bits" se obtienen 2^n combinaciones diferentes de "palabras binarias de n bits" con valores lógicos 0 y 1.

En la tabla 5.1 se muestra la secuencia en código de gris utilizada para n=4.

Tabla 5.1: Secuencia de código de gris para n=4.

		C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
$R=2^3$	P_0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$R=2^2$	P_1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
$R=2^1$	P_2	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
$R=2^0$	P_3	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
Valor Decimal		0	1	3	2	6	7	5	4	12	13	15	14	10	11	9	8

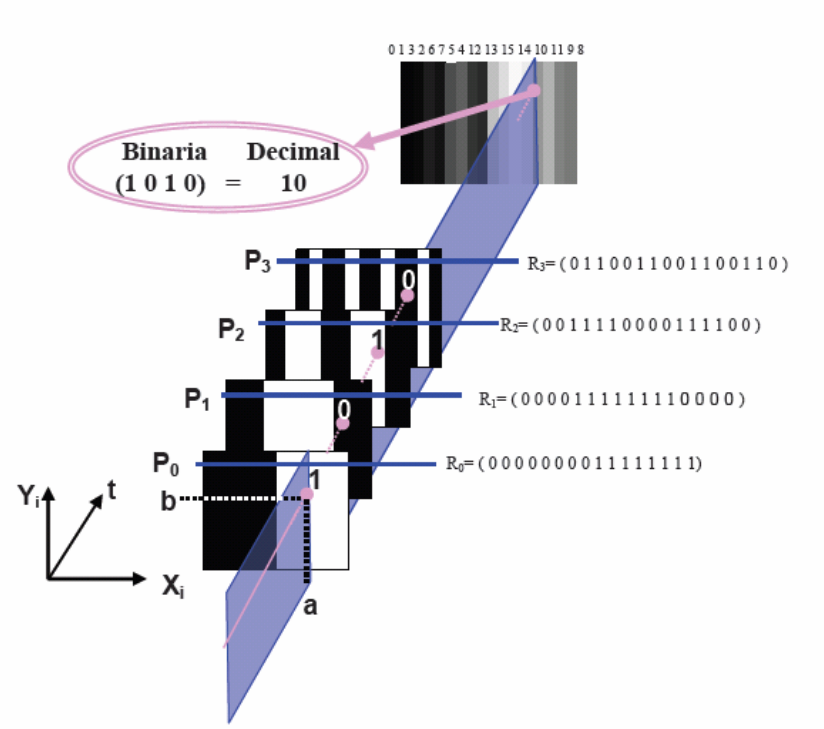


Figura 5.8: Combinación de los patrones.

R_0, R_1, R_2, R_3,de la tabla 2; codifican palabras de la más significativa a la menos significativa (2^3, 2^2, 2^1, 2^0,). Estos valores corresponden a la secuencia de valores en representación binaria de una fila de los patrones P_0, P_1, P_2, P_3 que se muestran en la figura 5.8.

Las columnas C_0, C_1, ...C15, que resultan de la combinación en el tiempo de P_0, P_1, P_2, y P_3, son códigos de palabras, cada una define una dirección de proyección diferente. En la última fila de la tabla se encuentra el valor decimal correspondiente a cada palabra en código binario.

En general, la lógica matemática permite definir la tabla 5.1 para un valor de n patrones obteniéndose las 2^n direcciones de proyección. Las filas permiten diseñar los

diferentes patrones de proyección en función del instante de proyección. De la misma manera, las columnas permiten crear las diferentes palabras binarias a identificar en el procedimiento de reconstrucción

5.5.2 Sistema de Proyección y Observación

La etapa anterior permite crear los n patrones bidimensionales que se deben proyectar para codificar 2^n direcciones de proyección. El sistema de proyección ubica las direcciones de proyección, codificadas en palabras binarias del espacio Xi, sobre el plano de referencia X.

La secuencia de números mostrada en la salida del proyector y en el plano de referencia de la figura 5.9 corresponde al valor decimal de las palabras binarias de cada dirección de proyección, para n=4.

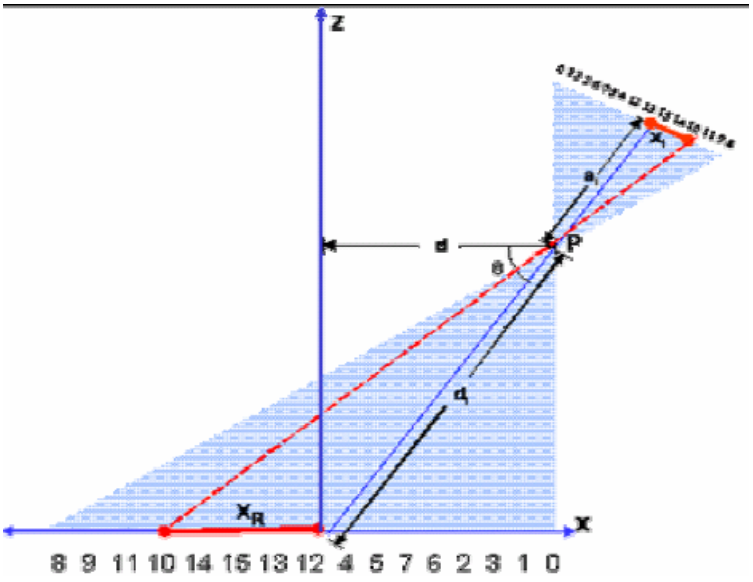


Figura 5.8: Esquema Óptico del Sistema de Proyección.

Así, tomando como ejemplo la palabra binaria (1010) que corresponde al valor decimal 10 ubicada en la posición X_i, esta es proyectada sobre el plano de referencia en la posición X_r, siguiendo una geometría de proyección no telecéntrica que matemáticamente corresponde a la siguiente ecuación:

$$X_R = \frac{d_i X_i}{a_i \sin\theta X_i \cos\theta} \tag{2}$$

donde a_i es aproximadamente la distancia entre el plano de proyector y la pupila de salida del mismo y d_i es la distancia entre la pupila de salida del proyector y el origen de coordenadas del plano de referencia.

Las imágenes en niveles de gris de todos los patrones proyectados son digitalizadas por el sistema de adquisición y almacenadas secuencialmente. En la figura 5.9 se muestra el esquema óptico del sistema de adquisición.

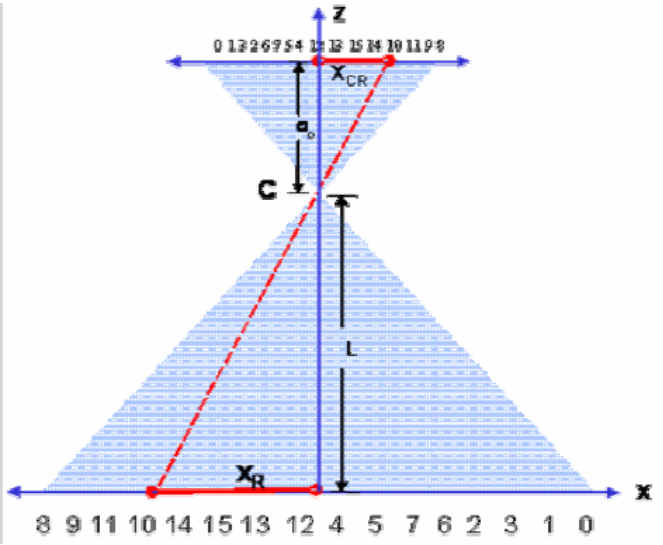


Figura 5.9: Esquema Óptico del Sistema de Adquisición.

A partir de la geometría no telecéntrica del sistema de observación se puede encontrar la proyección de la distancia X_R sobre el sensor de la CCD, utilizando la siguiente relación:

$$X_{CR} = \frac{a_o X_R}{L} \tag{3}$$

Donde a_o es aproximadamente la distancia focal del objetivo de la cámara.

5.5.3 Codificación de la Altura

En el plano de la cámara, la posición de cada píxel esta definida por los índices i, j. La altura del objeto Z(x,y) se evalúa para cada punto con respecto al plano de referencia. Al colocar un objeto sobre el plano referencia, debido a la triangulación, se presenta un corrimiento lateral de las palabras en código binario el cual codifica la altura para cada dirección de proyección.

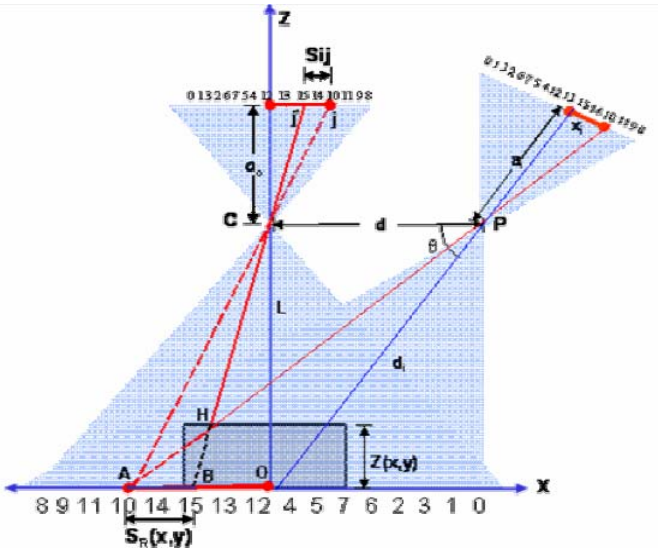


Figura 5.10: Geometría del sistema para la evaluación de la altura del objeto.

La codificación en altura por triangulación se esquematiza en la figura 5.10 tomando como ejemplo el rayo PA que pertenece a la dirección de proyección de la palabra 10 que corresponde a la posición Xi del plano de proyección. Sin objeto el rayo intercepta el plano de referencia en el punto A y es observado como la palabra 10 en el píxel de coordenadas (i,j) del plano de la CCD, dada en la ecuación (3). Al colocar un objeto sobre el plano de referencia, el rayo PA corta al objeto en el punto H y es ahora observado como la palabra 10, pero en la posición que antes pertenecía a la palabra 15 de coordenadas (i.j*) de la CCD. El corrimiento lateral de la posición de la palabra 10, del píxel j al píxel j*, es llamado Sij, se puede encontrar de la siguiente forma:

$$S_{ij} = j - j^* \tag{4}$$

Conociendo Sij se puede determinar el corrimiento $S_R(x,y)$ en el plano de referencia, que corresponde al segmento *AB* de la figura 5.10. Teniendo en cuenta la geometría no telecéntrica del sistema de observación se puede calcular matemáticamente por medio de la siguiente relación:

$$S_R(x,y) = S_{ij}\frac{FW}{N} \tag{5}$$

Donde, FW es el ancho del campo de visón de la cámara y N es el numero de palabras que se determina por el número de patrones proyectados, siendo N=2n, asumiendo que cada palabra ocupa un píxel de la imagen digitalizada. A partir del corrimiento de las palabras en el plano de referencia $S_R(x,y)$ y de la semejanza de los triángulos AHB y CHP se encuentra la siguiente relación en términos de la altura Z(x,y) del punto H:

$$\frac{\overline{AB}}{\overline{CP}} = \frac{Z(x,y)}{L - Z(x,y)} \tag{6}$$

en donde $\overline{CP}$ es la distancia **d** entre la pupila de salida del proyector **P** y la pupila de entrada de la cámara **C**.

Finalmente, reemplazando los segmentos $\overline{AB}$ y $\overline{CP}$ de la ecuación 6 por $S_R(x,y)$ y d respectivamente, y despejando Z(x,y) se obtiene la altura del punto H del objeto, dada en la siguiente ecuación:

$$Z(x,y) = \frac{LS_R(x,y)}{d + S_R(x,y)} \tag{7}$$

5.5.4 Proceso de Reconstrucción

La ecuación (7) permite calcular Z para cualquier punto digitalizado de la superficie del objeto, en función de los parámetros del sistema y de S_R para dicho punto. La deformación de S_{ij} depende del desplazamiento de la palabra binaria correspondiente. Como las imágenes de la superficie del cuerpo y del plano de referencia digitalizada por la CCD, están en niveles de gris, se debe binarizar para obtener los valores lógicos 0 y 1 utilizados en el plano objeto de proyección. Por lo tanto se requiere de un tratamiento de imágenes que permita binarizar y agrupar estos valores binarios para recuperar las palabras utilizadas en la proyección.

5.5.4.1 Binarización y Manejo de Sombras

La etapa de binarización permite convertir la imagen en niveles de gris a valores binarios que corresponden a planos significativos en función de la palabra binaria. Existen diferentes estrategias de binarización, la elección de la estrategia más apropiada depende solamente de las características de la imagen adquirida en

términos del nivel de iluminación, del contraste y de la presencia de discontinuidades en la superficie del objeto.
Tradicionalmente en la binarización se escoge un sólo valor de umbral, que es el mismo para todos los píxeles de la imagen. El valor más apropiado del umbral se fija, dependiendo del histograma de cada imagen. Este valor de umbral debe estar entre el nivel de gris más grande de la imagen y el nivel de gris más pequeño. Teniendo en cuenta la reflectividad de la superficie se hace difícil elegir este umbral cuando la densidad de franjas es muy alta ó cuando el nivel de gris más alto y el nivel de gris más bajo del rango del umbral están cercanos uno del otro.

En la figura 5.11 se muestra la binarización para cuatro diferentes valores de umbral de una imagen del patrón de franjas más denso deformado por un cilindro. También en esta figura, se puede apreciar el efecto de elegir un umbral único en función del contraste de las franjas y se observa que para valores de umbral pequeños se elimina información de las franjas mientras que para umbrales muy altos aparecen regiones negras perdiéndose también información. Esto demuestra que no se pueden binarizar correctamente imágenes de patrones de franjas con un valor de umbral global.

Por esta razón es necesario implementar un método de binarización que actué localmente. De esta manera se puede binarizar la imagen independiente de los cambios de las condiciones de iluminación ambiental, de la variación de las propiedades de reflectividad de la superficie y de la forma del objeto.

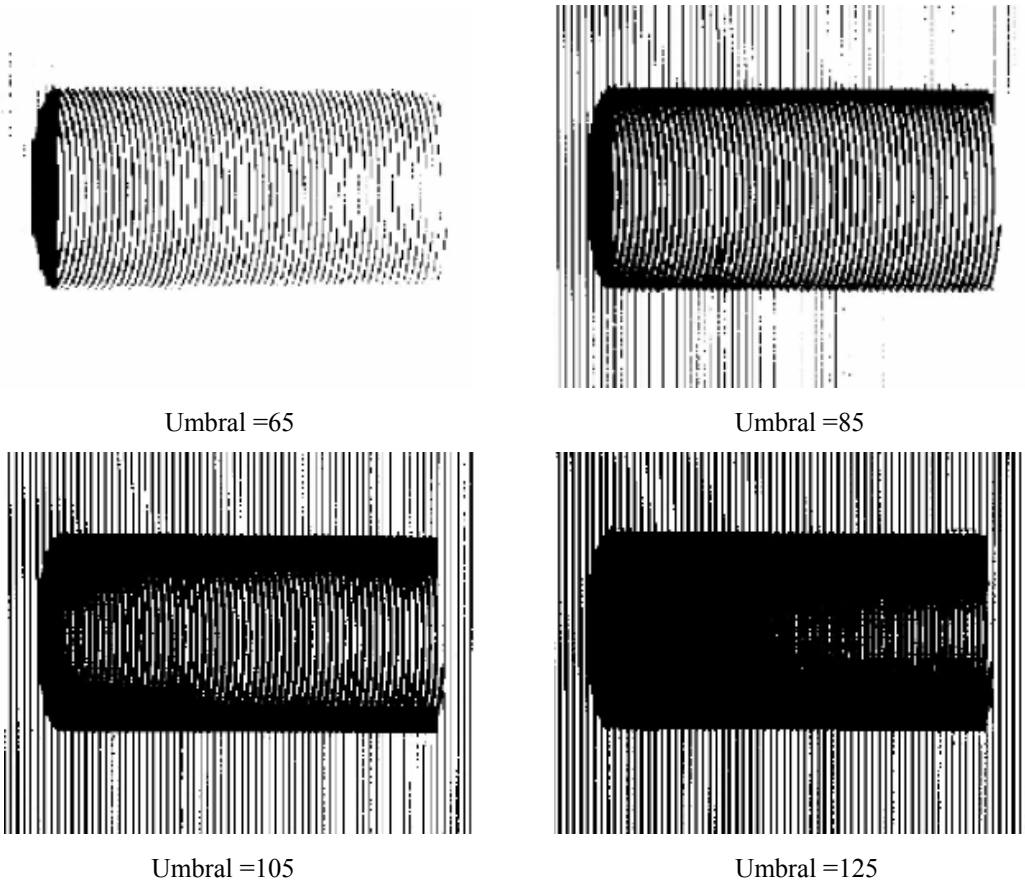
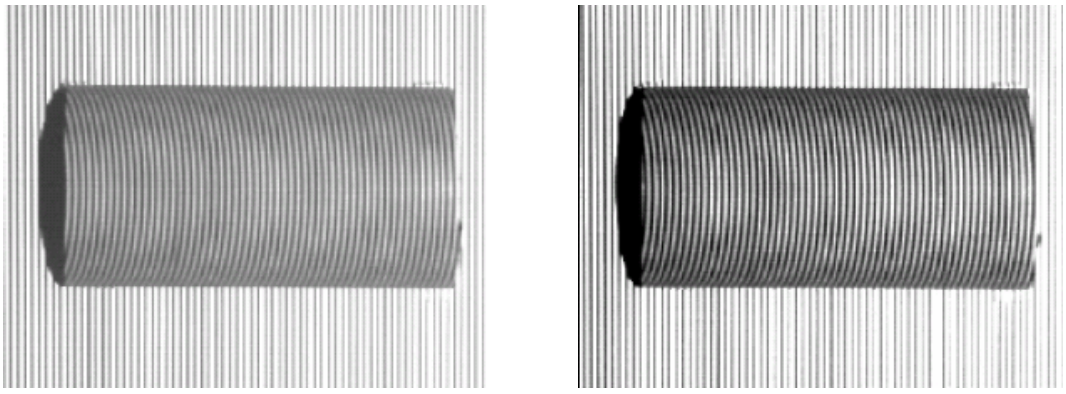


Figura 5.11: Binarización con un umbral global.

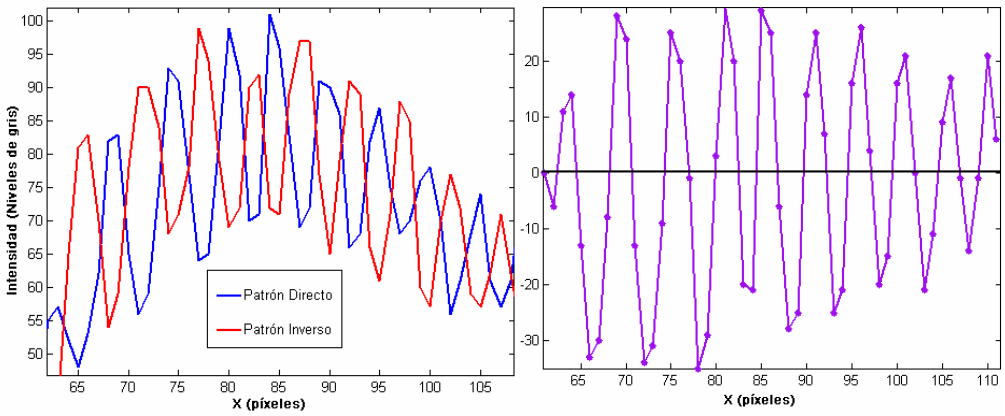
Aunque existen varias técnicas, el método de binarización escogido para el desarrollo de la técnica de código de gris es el sugerido por Marjàn Trobina, quien afirma que lo robusto de la técnica depende de la correcta localización de los bordes de las franjas. Es decir encontrar donde termina una franja negra y comienza una blanca o viceversa. Este método requiere de 2 imágenes por cada patrón, una imagen de la proyección patrón directo y otra de la proyección del patrón inverso para la localización de las franjas.



Patrón Directo Patrón Inverso

Figura 5.12: Imagen del patrón de franjas más denso proyectado sobre un cilindro.

En la figura 5.13 se muestra el perfil de las dos imágenes a lo largo de una fila para el patrón de franjas más fino proyectado sobre un objeto.



a b

Figura 5.13: a) Perfil del patrón de franjas directo e inverso más fino a lo largo de una fila b) Resta de los dos perfiles.

En esta figura se puede observar que los bordes están localizados en los puntos de intersección del perfil azul "patrón directo" y el perfil rojo "patrón inverso". La resta de estos dos perfiles da como resultado un vector que tiene valores

positivos, negativos y ceros como se observa en la figura 5.13(b). Un valor de cero en la diferencia corresponde a puntos de las imágenes con igual valor en nivel de gris. Al binarizar este vector con valor de umbral cero las franjas se localizan correctamente. Para binarizar toda la imagen este proceso se realiza de forma bidimensional. La figura 5.14 muestra la proyección de los patrones directo e inverso más fino proyectado sobre un cilindro y la imagen binarizada con el principio propuesto por Marján Trobina.

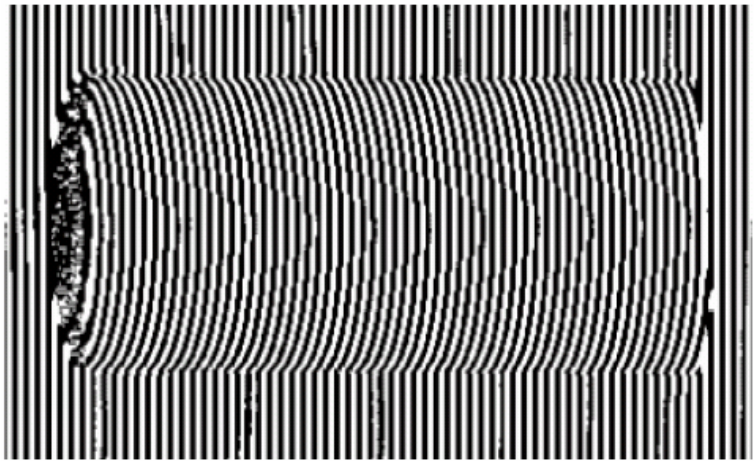


Figura 5.14: Imagen del patrón binarizado con el método de Trobina.

Aunque se elija un procedimiento de binarización adecuado, existen en el proceso de reconstrucción problemas en regiones donde no hay franjas, debido a la *sombra* producida por la topografía del objeto y al tipo de iluminación. En la figura 5.14 se observa que después de la binarización estas zonas contienen valores binarios aleatorios que no pertenecen a las franjas, por lo tanto no contienen información útil. A pesar de que estas zonas no afectan la información de sus vecinos es necesario definir una *máscara* que permita validar los píxeles útiles para obtener la reconstrucción del objeto. Para obtener la máscara es necesario adquirir una imagen en niveles de gris del objeto completamente iluminado por el proyector. Esta imagen se binariza de forma que las zonas de sombra tengan el valor lógico cero y las zonas

útiles el valor lógico uno. Finalmente, esta máscara se superpone a cada uno de los patrones binarizados.

5.5.4.2 Construcción de la Matriz de Corrimientos

Teniendo las imágenes de todos los patrones proyectados sobre el objeto y sobre el plano de referencia correctamente binarizadas, se obtienen los valores lógicos 0 ó 1 para cada posición, necesarios para la formación de las palabras binarias. Como se explicó en la sección 5.5.1 al combinar las imágenes binarizadas de n patrones proyectados en n tiempos diferentes se forman 2^n palabras binarias. De esta manera, combinando las imágenes binarizadas de los patrones proyectados sobre el plano de referencia, se halla la *matriz de palabras de la referencia (MPR).* Igualmente, combinando las imágenes binarizadas de los patrones deformados por la topografía del objeto se halla la *matriz de palabras del objeto (MPO).* A partir de estas dos matrices se puede calcular el corrimiento en píxeles de cada palabra. Así, para cada fila se busca una palabra en MPO y se extrae la posición de la columna **j*** en la que está ubicada. Luego, se busca la fila correspondiente en MPR y se extrae la posición de la columna **j** en la que está ubicada. Como la posición de la palabra j* esta desplazada con respecto a la posición j debido a la altura del cuerpo en ese punto, se calcula el corrimiento en píxeles de la palabra (Sij) por medio de la ecuación (4). Este corrimiento es almacenado en la posición j* de una nueva matriz. Repitiendo este procedimiento para todas las palabras y todas las filas se halla la *matriz de corrimientos de la imagen* S_{ij}, donde cada elemento de la matriz contiene un valor en píxeles del corrimiento de la palabra respectiva, que codifica la altura para cada punto del objeto.

La figura 5.15 se muestra *MPR* , *MPO* usando n=8. La imagen en color representa una ampliación de las zonas amarillas marcadas sobre *MPR* y *MPO*, donde cada color corresponde a una palabra binaria diferente. En esta última figura ampliada se puede ver que sobre la fila 230 la palabra 356 se corre del píxel 600 en MPR al píxel 620

en MPO, debido a la altura del cilindro en ese punto. Repitiendo este procedimiento en una fila para todas las palabras y para todas las filas se halla la matriz de corrimientos S_{ij} hallada a partir de la MPO y la MPR. La matriz S_{ij} se muestra en la figura 5.16.

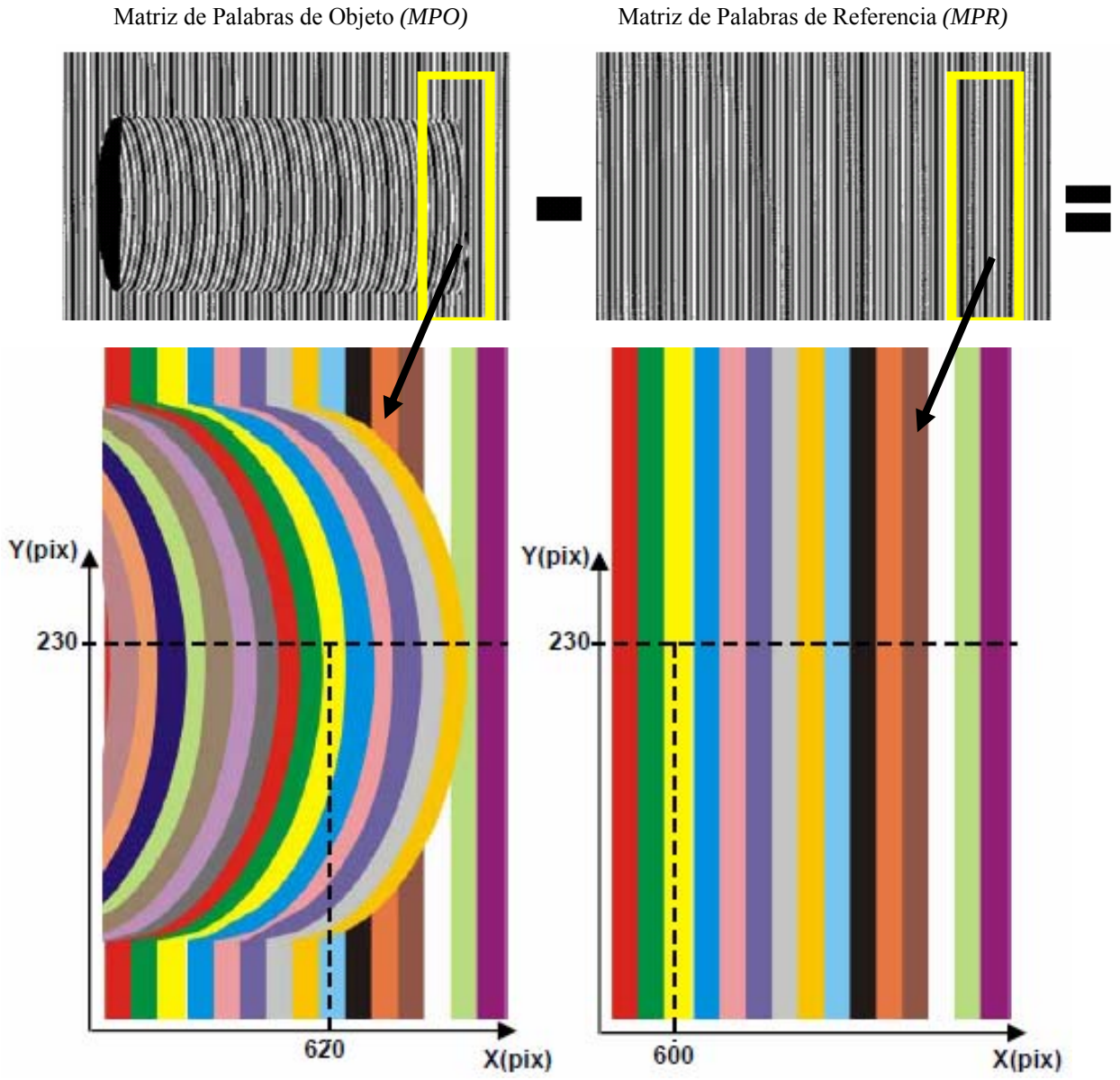


Figura 5.16: Proceso para la obtención de la matriz de corrimientos Sij.

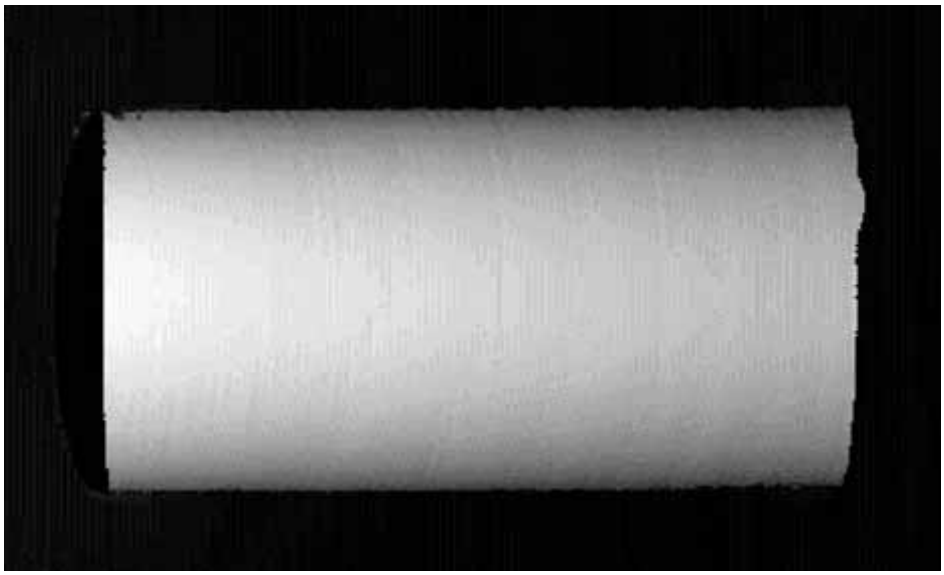


Figura 5.16: Matriz de corrimientos.

5.6 ESCOGENCIA DEL FORMATO DE ARCHIVO PARA EL MODELO 3D

Debido a la gran diversidad de formatos de archivos para modelos 3D, se hace necesaria la escogencia de un formato de archivo para poder almacenar el objeto reconstruido.

Algunos de los formatos utilizados por aplicaciones para el modelado 3D se encuentran: 3D2, 3DM, 3DS, DXF, FLT, MGF, MTV, NURBS, OBJ, POV, VRML, MD2, se escogió el formato OBJ por tratarse de un formato abierto y universalmente aceptado por otros fabricantes de software grafico 3D. El formato de archivo OBJ es un formato sencillo de datos que representa la geometría 3D del objeto, la posición de cada vértice, la posición de UV de cada coordenada de textura de vértices, normales, y las caras que hacen que cada polígono este definido como una lista de vértices y vértices de la textura.

5.7 ESCOGENCIA DEL LENGUAJE DE PROGRAMACIÓN

Para la codificación de los diferentes módulos del software, es utilizado el lenguaje de programación C++ por ofrecer velocidad de procesamiento y soportar librerías dedicadas al tratamiento de imagen.

Para el desarrollo de la interfaz de usuario se escogió el lenguaje de programación JAVA ya que ofrece una API muy completa para la creación de botones, etiquetas, cajas de texto, menús, ventanas, entre otros, como también API's para el enlace dinámico y estático de librerías desarrolladas en C/C++ y para el desarrollo de aplicaciones 3D, como lo son JNI, JAVA3D, respectivamente.

5.7.1 Aspectos fundamentales del lenguaje de programación C++

C++ es un lenguaje de programación, diseñado a mediados de los 80, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

El nombre C++ fue propuesto por Rick Masciatti en el año 1983, cuando el lenguaje fue utilizad por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "C Con Clase". El C++, introduce el operador de incremento (++) de C para indicar que C++ es un incremento de C que es una versión mejorada de C.

La definición "oficial" del lenguaje nos dice que C++ es un lenguaje imperativo orientado a objetos de propósito general basado en el lenguaje C, al que se han añadido nuevos tipos de datos, clases, plantilla, mecanismos de excepciones, sistemas de espacio de nombres, funciones inline, sobrecarga de operaciones, referencias, operadores para manejo de memoria persistente, y algunas utilidades

adicionales de librerías. Es un lenguaje hibrido que se puede compilar y resulta mas sencillo de aprender para los programadores que ya conoces C. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos.

Las principales características de C++ son: el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica (template). Por ende, se puede decir que C++ es un lenguaje que abarca tras paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

C++ ha experimentado un extraordinario éxito desde su creación. De hecho muchos sistemas operativos, compiladores he interpretes han sido escritos en C++ (Windows, y JAVA). Una de las razones de su éxito es ser un lenguaje de programación de propósito general que se adapta a múltiples situaciones.

5.7.1.1 Características de C++

C++ tiene varias características que otros lenguajes de programación no tienen. Las más destacadas son:

- Programación orientada a objetos: la posibilidad de orientar la programación a objetos permite al programador diseñar las aplicaciones desde un punto de vista más cercano a la vida real. Además, permite la reutilización de código de una manera más lógica y productiva.
- Portabilidad: un código escrito en C++ puede ser compilado en casi todo tipo de ordenadores y sistemas operativos sin hacer apenas cambios.

- Brevedad: el código escrito en C++ es muy corto en comparación con otros lenguajes, sobretodo porque en este lenguajes es preferible el uso de caracteres especiales que las "palabras claves".
- Programación modular: un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Además esta característica permite unir código en C++ con códigos producidos en otros lenguajes de programación como ensamblador o el propio C.
- Velocidad: el código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel.
- Potencia: C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel. Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel como:
    - La posibilidad de redefinir los operadores (sobrecarga de operadores).
    - La identificación de tipos en tiempo de ejecución (RTH).

5.7.2 Aspectos fundamentales del lenguaje de programación JAVA

JAVA es un lenguaje que tiene sus propias bibliotecas llamadas paquetes, que son independientes de la plataforma. Por ello no es necesario preocuparse si la aplicación se va a ejecutar en una plataforma x86, x64, POWERPC o una plataforma SPARG.

El compilador de JAVA no genera instrucciones nativas en su lugar genera los llamados "códigos de byte" (byte code) para la máquina virtual JAVA (JVM), que es una máquina que no existe físicamente.

Los creadores de JAVA (Sun Micro Systems) y otras empresas han desarrollado versiones de la JVM para una gran parte de las plataformas existentes en el mercado, es decir cada plataforma tiene su propia máquina virtual, y es esta que ejecuta los byte code.

JAVA ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de este.

JAVA reduce en un 50% los errores más comunes de programación con los lenguajes cono C y C++ al eliminar muchas de las características de este entre las que destacan:

- Aritmética de punteros.
- No existen referencias.
- Registros (Struct).
- Definiciones de tipos (tipedef)
- Macros (# define)
- Necesidad de liberar memoria (free).

5.7.2.1 Características de JAVA

- JAVA es orientado a objetos: JAVA trabaja con sus datos como objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.

- JAVA es distribuido: JAVA se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutina para acceder e interactuar

con protocolos como http y FTP. Esto permite a los programadores accder a la información a través de la red con tanta facilidad como los ficheros locales.

- JAVA es multiplataforma: el código fuente de JAVA se compila a un código de byte code de bajo nivel independiente de la maquina. Este código esta diseñado para ejecutarse en una maquina hipotética que es implementada por un sistema run-time, que si es independiente de la máquina.

- JAVA es robusto: java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en java ayuda a detectar errores, lo antes posible, en el siclo de desarrollo. JAVA obliga a la declaración explicita de métodos, reduciendo asi las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte de programador de la liberación o corrupción de memoria. También implementa los arreglos auténticos, en vez de listas enlazadas de punteros, con comprobación de límites, para evitar la posibilidad de sobrescribir o corromper memoria resultado de punteros que señalan a zonas equivocadas. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en JAVA.

- JAVA es seguro: las aplicaciones de JAVA resultan extremadamente seguras ya que no acceden a zonas delicadas de memoria o de sistema con la cual evitan la intensión de ciertos virus. Java no posee una semántica específica para modificar la pila del programa, la memoria libre o utilizar objetos y métodos de un programa sin los privilegios del núcleo del sistema operativo. Además para evitar modificación por parte de los crackers de la red, implementa un método seguro de autenticación por clave pública. El cargador

de clases puede verificar una firma digital antes de realizar una instancia de un objeto. Por tanto, ningún objeto se crea y almacena en memoria, sin que se validen los privilegios de acceso. Es decir, la seguridad se integra en el momento de compilación con el nivel de detalles y de privilegios que sea necesario.

- JAVA es interpretado: JAVA para conseguir ser un lenguaje independiente del sistema operativo y del procesador que incorpore la maquina utilizada es tanto interpretado como compilado. Y esto no es ningún contrasentido, el código fuente escrito con cualquier editor se compila generando el byte code. Este código intermedio es de muy bajo nivel, pero sin alcanzar las instrucciones maquinas propias de cada plataforma. El byte code corresponde al 80% de las instrucciones de la aplicación. Ese mismo código es el que se puede ejecutar sobre cualquier plataforma. Para ello hace falta el run-time, que si es completamente dependiente de la maquina y del sistema operativo que interpreta dinámicamente el byte code y añade el 20% de instrucciones que faltaban para su ejecución. Con este sistema es fácil crear aplicaciones multiplataforma, pero para ejecutarlas es necesario que exista es rum-time correspondiente al sistema operativo utilizado.

- JAVA es multithreaded (multi-hilo): JAVA permite muchas actividades simultaneas en un programa. Los threads, son básicamente pequeños procesos o piezas independientes de un gran proceso. Al estar los theads incluidos en el lenguaje son mas fáciles de usar y mas robustos que las implementaciones en C o C++. El beneficio de ser multi-hilo consiste en un mejor rendimiento interectivo y mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real esta limitado a las capacidades del sistema

operativo sobre el que corre, aun supera a los entornos de flujo único de programas tanto en facilidad de desarrollo como en rendimiento.

5.8 ENTORNOS DE DESARROLLO

5.8.1 Entorno de desarrollo DEV-C++

Dev-C++ es un entorno de desarrollo integrado (IDE por sus siglas en inglés) para programar en lenguaje C/C++. Usa MinGW que es una versión de GCC (GNU Compiler Collection) como su compilador. Dev-C++ puede además ser usado en combinación con Cygwin y cualquier compilador basado en GCC.

El Entorno está desarrollado en el lenguaje Delphi de Borland. Tiene una página de paquetes opcionales para instalar, con diferentes bibliotecas de código abierto.

Se trata de un entorno de trabajo dedicado al lenguaje de programación anteriormente mencionado en un sistema de ventanas múltiples fácil de interpretar y utilizar.

Entre las tareas a realizar con Dev-C++**,** mencionamos la depuración de código, reconocimiento automático de sintaxis, creación de ejecutables, corrección de errores y mucho mas, facilitando y completando el trabajo con cientos de librerías ya incluidas y otras que puedes añadir.

Dev-C++ es un muy buen programa, su nueva versión ha mejorado diversos errores menores y ha reformado significativamente su aspecto visual, logrando un entorno mas agradable y cómodo para el usuario. Además permite imprimir y posee facilidades para buscar y reemplazar.

Actualmente (enero de 2010) parece que el proyecto no está más en desarrollo activo. No ha habido noticias ni ninguna versión actualizada desde el último lanzamiento en 2005. Hay un equipo de desarrollo que ha tomado el IDE Dev-C++ y le ha agregado nuevas características tales como ayuda para los recopiladores múltiples y un diseñador del RAD para los usos de los wxWidgets. Este IDE se puede encontrar bajo el nombre de wxDev-C++. Está en un fuerte desarrollo.

5.8.1.1 Requisitos mínimos del sistema para la instalación de DEV-C++

- Microsoft Windows 95, 98, NT 4, 2000, XP, Vista
- 32 MB RAM
- 100 Mhz Intel compatible CPU
- 30 MB de espacio libre de disco

En la figura 5.17 se muestra el entorno de programación DEV-C++.

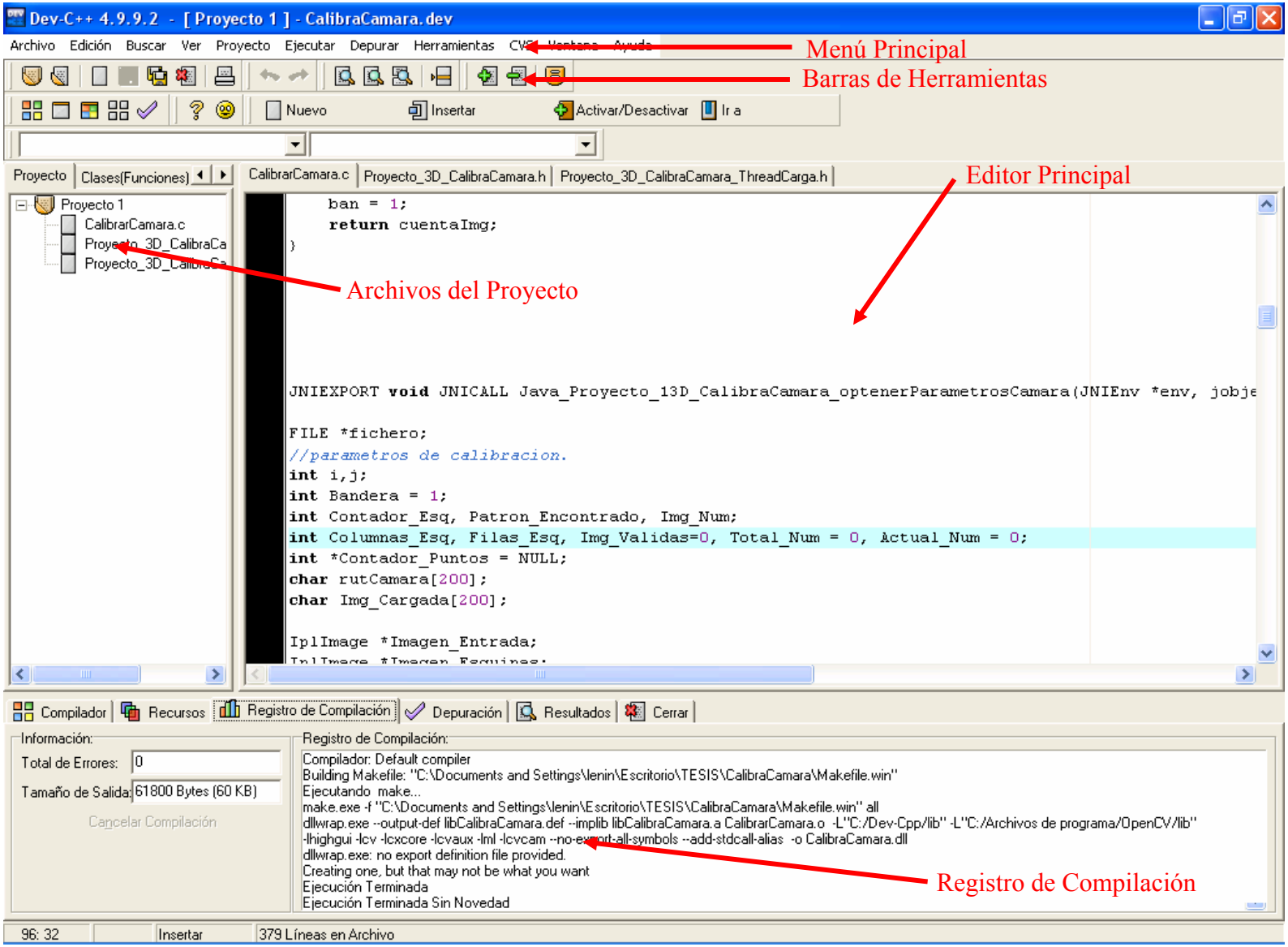


Figura 5.17: Entorno de programación DEV-C++.

5.8.2 Entorno de desarrollo NetBeans

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

NetBeans comenzó como un proyecto estudiantil en Republica Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad Carolina en Praga. La meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la de Delphi.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas

- Framework basado en asistentes (diálogos paso a paso)

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

5.8.2.1 Requisitos mínimos del sistema para la instalación de NetBeans

Microsoft Windows XP Professional SP3:

- Procesador: Intel Pentium III o equivalente a 800 MHz
- Memoria: 512 MB
- Espacio de disco: 750 MB de espacio libre en el disco

Linux

- Procesador: Intel Pentium III o equivalente a 800 MHz
- Memoria: 512 MB
- Espacio de disco: 650 MB de espacio libre en el disco

Requisitos de Software

NetBeans IDE funciona en el kit de desarrollo de Java SE (JDK), que consta del entorno de ejecución de Java y de herramientas para desarrolladores para la compilación, depuración y ejecución de aplicaciones escritas en lenguaje Java.

En la figura 5.18 se muestra el entorno de programación NetBeans.

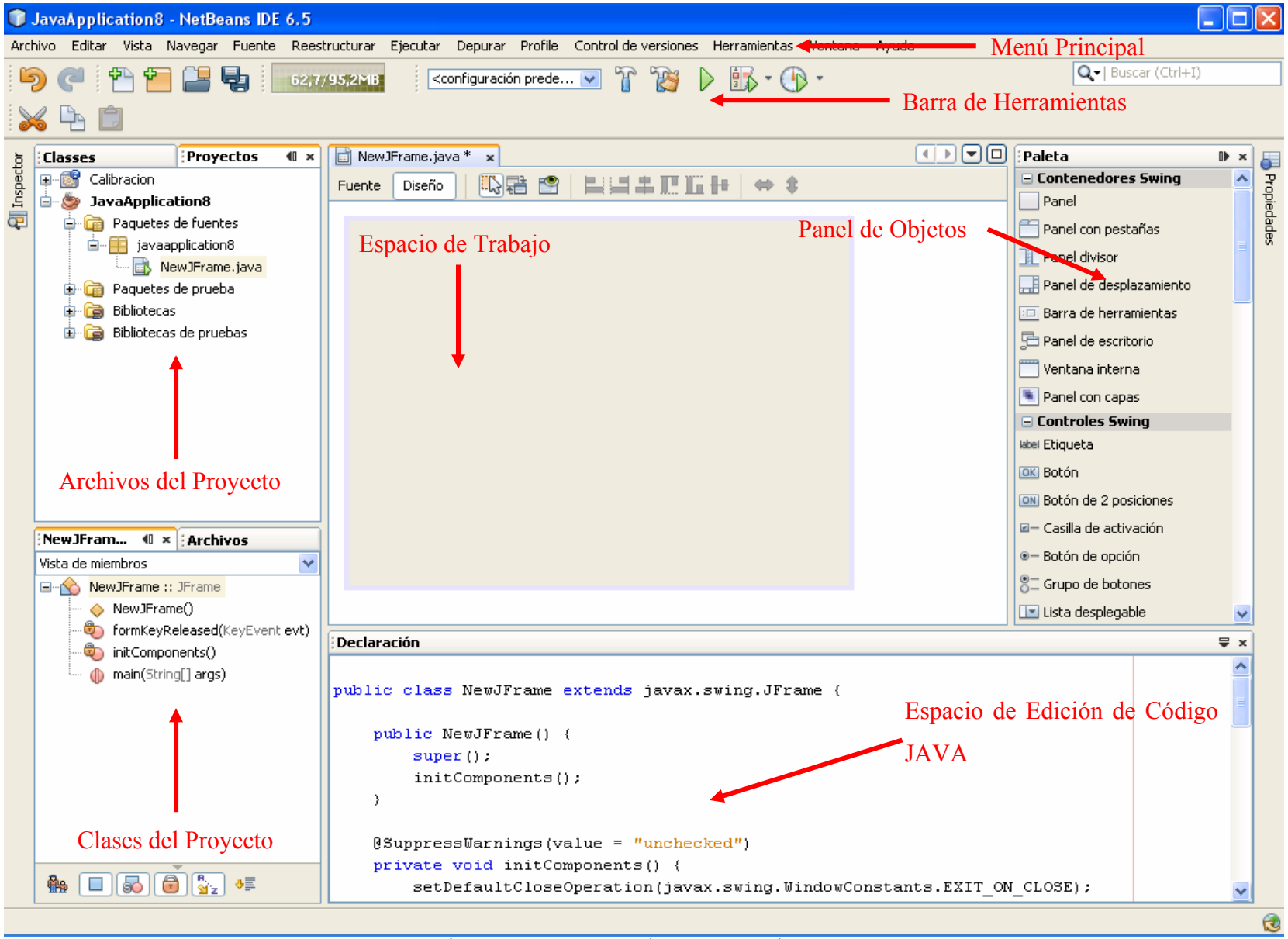


Figura 5.18 Entorno de programación NetBeans.

5.9 ELABORACIÓN DE LOS FLUJOS DE TRABAJO DISEÑO, IMPLEMENTACIÓN Y PRUEBAS

Los flujos de trabajo requisitos y análisis no son de interés en esta fase ya que se llevaron a cabo en la fase de elaboración, quedando así los flujos de trabajo de diseño, implementación y pruebas para esta fase.

5.9.1 Requisitos

No es de interés en esta fase.

5.9.2 Análisis

No es de interés en esta fase.

5.9.3 Diseño

En este flujo de trabajo nos enfocaremos en el diseño de la interfaz gráfica de usuario que se ha construido haciendo uso de los objetos que JAVA dispone, como botones, menús desplegables, entre otros que facilitan la interacción del usuario con el software.

Por medio de la interfaz gráfica de usuario se puede tener acceso a las diferentes funciones del software como por ejemplo:

- Abrir Archivo 3D.
- Guardar Archivo.
- Configurar Sistema.
    - Configuración del umbral binario para la máscara.
    - Calibración de la cámara.
    - Introducir parámetros externos del sistema.
    - Captura del plano de proyección de referencia.
- Visor 3D.
    - Panel de Control del Visor 3D.
- Reconstruir Objeto.

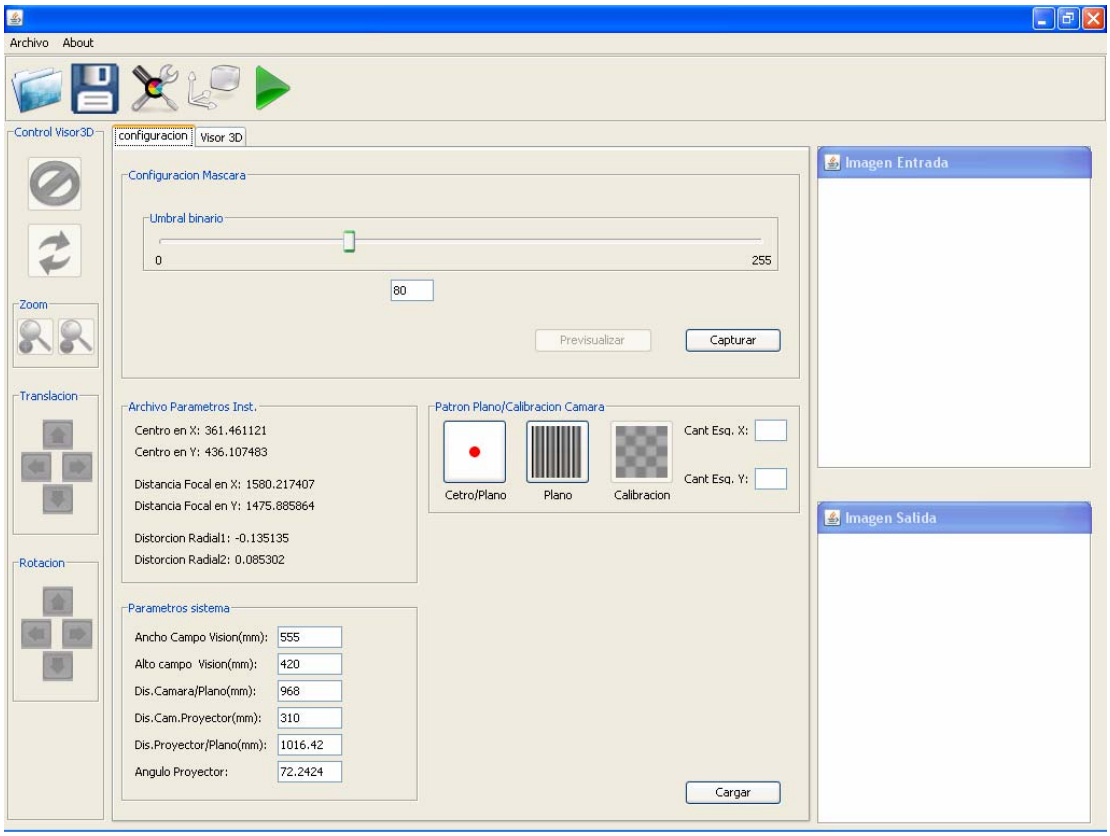


Figura 5.19: Pantalla de Configuración 3D del Sistema.

Fuente: Los Autores.

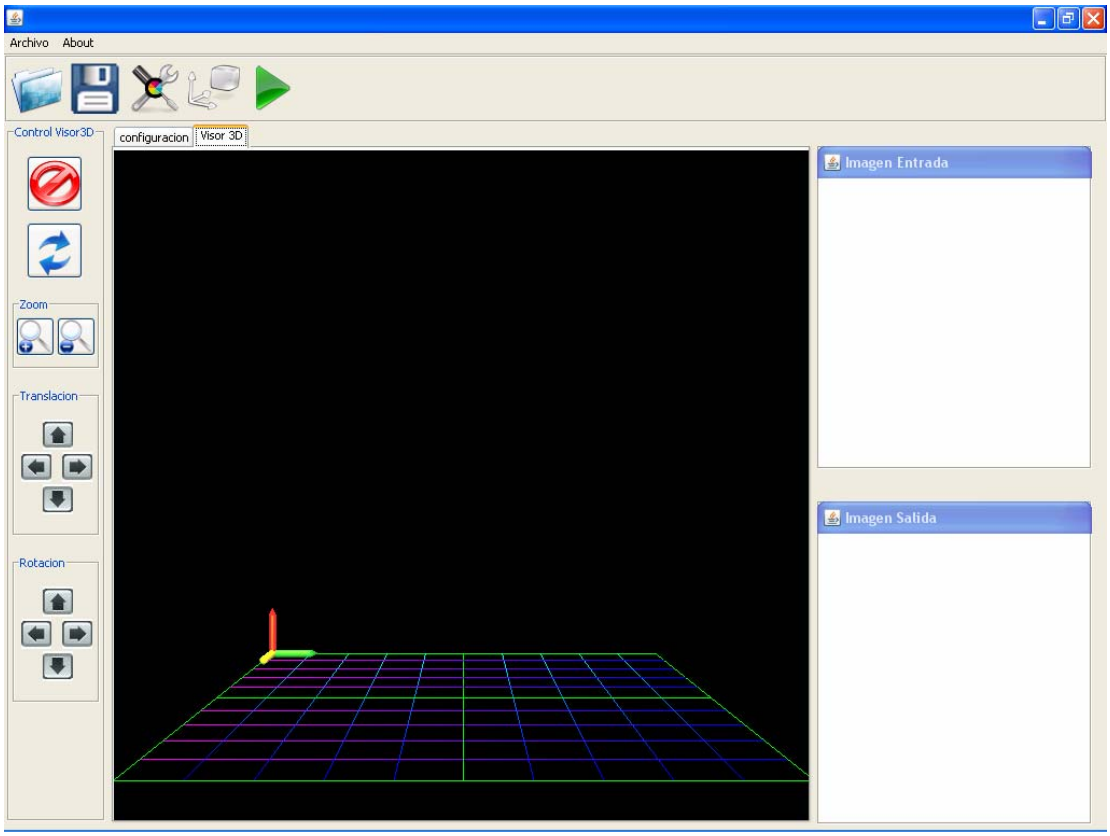


Figura 5.20: Pantalla de Visualización 3D del Sistema.

Fuente: Los Autores.

5.9.4 Implementación

Este flujo de trabajo implementa y lleva a cabo las pruebas de unidad de todos los componentes, trabajando principalmente a partir del modelo de diseño. Aquí se integran todos los componentes del sistema de software para obtener una versión ejecutable que cumpla con los requisitos que se establecieron en la fase de inicio. Los componentes de sistema se obtuvieron a partir del modelo de diseño, estos se expresan como componentes ejecutables del sistema, es decir, componentes ya codificados y en funcionamiento.

En la figura 5.21 se muestra el diagrama de componentes del sistema.

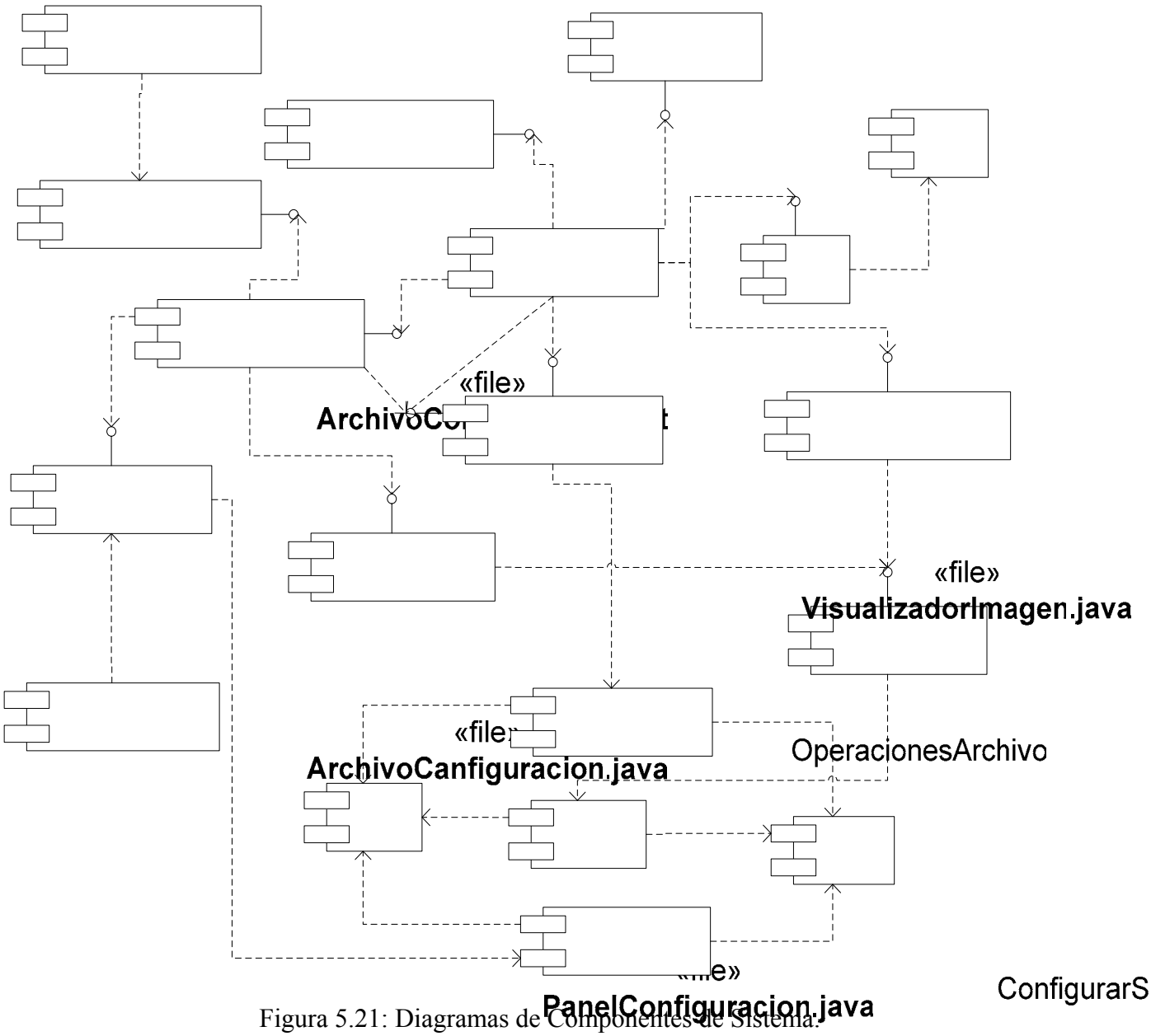


Figura 5.21: Diagramas de Componentes de Sistema.

Fuente: Los Autores.

5.9.5 Pruebas

El principal objetivo de este flujo de trabajo es realizar y evaluar las pruebas de los componentes obtenidos durante la implementación del sistema, con la finalidad de medir su funcionamiento y efectividad.

Para llevar a cabo este flujo de trabajo se aplicaron al sistema las pruebas de unidad y las pruebas de integración de los componentes más relevantes del sistema.

5.9.5.1 Pruebas de Unidad

Las pruebas de unidad se realizan para comprobar el funcionamiento individual de los componentes que no se derivan más.

A cada componente se le realizaron pruebas de caja negra, dichas pruebas verifican su comportamiento.

La prueba de la caja negra toma en cuenta la salida que el componente devolverá en función a una determinada entrada. Para la implementación de las mismas se determinaron las clases de equivalencia, las cuales representan un conjunto de estados válidos o inválidos para diferentes entradas.

5.9.5.1.1 Pruebas de unidad del componente PanelConfiguracion.java

Tabla 5.2: Clases de equivalencia del componente PanelConfiguracion.java.

Nº	Campo	Clase de Equivalencia	Valido	Invalido
1	Ancho Campo Visión	Es Número	**X**	
2	Ancho Campo Visión	No Numérico		**X**
3	Ancho Campo Visión	>1 y <3000 (Parte entera)	**X**	
4	Ancho Campo Visión	< 1 o >3001 (Parte entera)		**X**
5	Ancho Campo Visión	>1 y <99 (parte decimal)	**X**	
6	Ancho Campo Visión	<1 o > 99		**X**
7	Alto Campo Visión	Es Número	**X**	
8	Alto Campo Visión	No Numérico		**X**
9	Alto Campo Visión	>1 y <3000 (Parte entera)	**X**	
10	Alto Campo Visión	< 1 o >3001 (Parte entera)		**X**
11	Alto Campo Visión	>1 y <99 (parte decimal)	**X**	
12	Alto Campo Visión	<1 o > 99		**X**
13	Dis. Camara/Plano	Es Número	**X**	
14	Dis. Camara/Plano	No Numérico		**X**
15	Dis. Camara/Plano	>1 y <3000 (Parte entera)	**X**	
16	Dis. Camara/Plano	< 1 o >3001 (Parte entera)		**X**
17	Dis. Camara/Plano	>1 y <99 (parte decimal)	**X**	
18	Dis. Camara/Plano	<1 o > 99		**X**
19	Dis. Cam. Proyector	Es Número	**X**	
20	Dis. Cam. Proyector	No Numérico		**X**
21	Dis. Cam. Proyector	>1 y <3000 (Parte entera)	**X**	
22	Dis. Cam. Proyector	< 1 o >3001 (Parte entera)		**X**
23	Dis. Cam. Proyector	>1 y <99 (parte decimal)	**X**	
24	Dis. Cam. Proyector	<1 o > 99		**X**

25	Umbral Binario	Es Número Entero	**X**	
26	Umbral Binario	No Numérico		**X**
27	Umbral Binario	>1 y <255	**X**	
28	Umbral Binario	< 1 o >526		**X**
29	Umbral Binario	Es Decimal		**X**

Fuente: Los Autores

Tabla 5.3: Pruebas de caja negra del componente PanelConfiguracion.java.

Campo	Casos de Prueba	Salida	Clases Cubiertas
Ancho Campo Visión	150,45	**Valido**	**1,3,5**
Ancho Campo Visión	154g	**Invalido**	**2**
Ancho Campo Visión	4000,568	**Invalido**	**4,6**
Alto Campo Visión	405	**Valido**	**7,9**
Alto Campo Visión	ery	**Invalido**	**8**
Alto Campo Visión	2010,32	**Valido**	**7,9,11**
Dis. Camara/Plano	98,k	**Invalido**	**14**
Dis. Camara/Plano	“ ”	**Invalido**	**14**
Dis. Camara/Plano	1300	**valido**	**13,15**
Dis. Cam. Proyector	25	**Valido**	**19,21**
Dis. Cam. Proyector	*4,2	**Invalido**	**20**
Dis. Cam. Proyector	330.21	**Valido**	**19,21,23**
Umbral Binario	@54	**Invalido**	**26**
Umbral Binario	30	**Valido**	**25,27**
Umbral Binario	-105,30	**Invalido**	**28,29**

Fuente: Los Autores

5.9.5.1.2 Pruebas de unidad del componente CalibrarCamara.java

El componente *CalibrarCamara.java,* tiene como entrada una secuencia de imágenes de una plantilla de compuesta por cuadros negros y blancos en forma de tablero de ajedrez, este componente también recibe la cantidad de esquinas internas horizontales y verticales de la plantilla para generar el archivo de parámetros intrínsecos de la cámara. Hay que destacar que este componente debe recibir al menos diez (10) imágenes de la plantilla para poder obtener buenos parara metros de calibración. En la figura 5.22 se ejemplifica la prueba de unidad que se llevo a cabo.

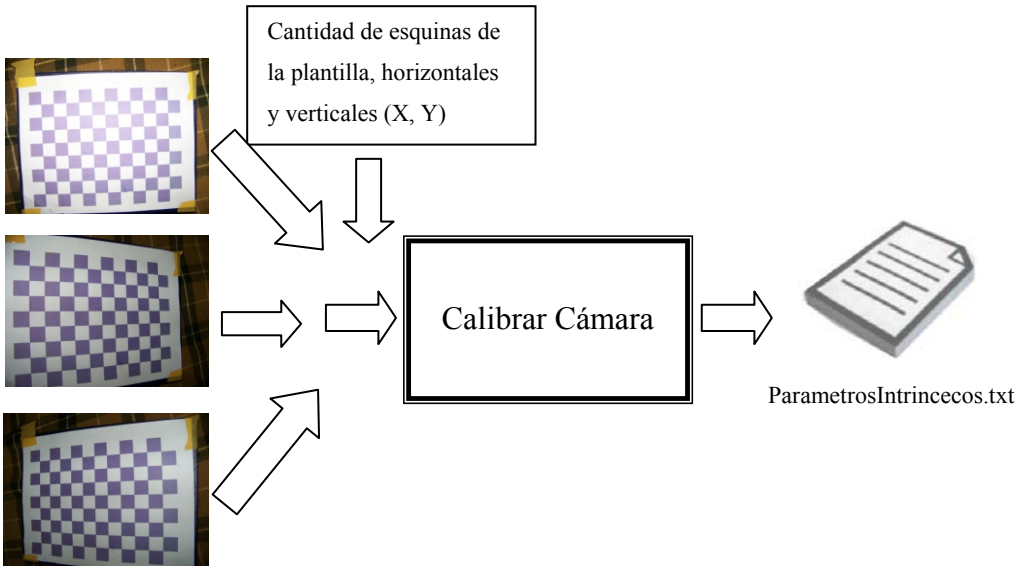


Figura 5.22: Prueba de unidad del componente CalibrarCamara.java.

Fuente: Los Autores.

5.9.5.1.3 Pruebas de unidad componente CapturaImagen.java

El componente *CapturaImagen.java,* recibe la orden de activación de los componentes *GeneradorDePatrones.java* y *PantallaMascara.java,* para iniciar la captura de imágenes del ambiente. En la figura 5.24 se ejemplifica la prueba de unidad que se llevo a cabo.

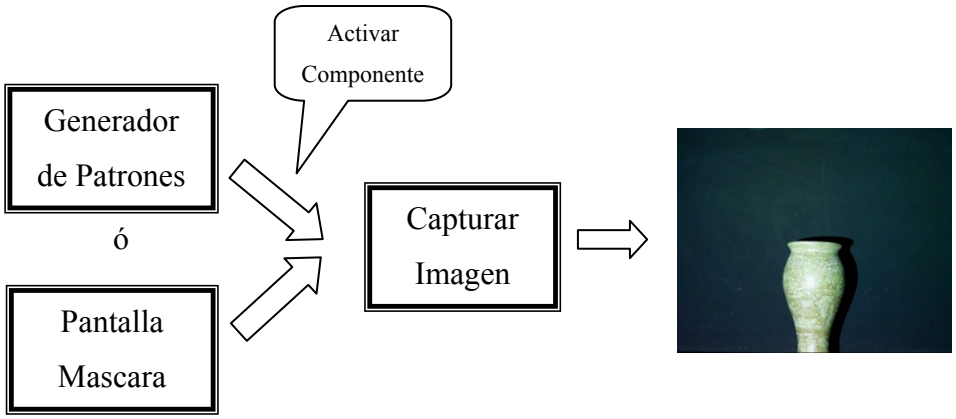


Figura 5.23: Pruebas de unidad del componente CapturarImgen.java.

Fuente: Los Autores

5.9.5.1.4 Pruebas de unidad del componente ReconstruirObjeto.java

El componente *ReconstruirObjeto.java,* realiza tres (3) procesos importantes *GenerarMatrizDeProyeccion, GenerarMatrizDeCorrimiento, ReconstruirObjeto,* esenciales para la obtención de la nube de puntos.

- Proceso *GenerarMatrizDeProyeccion*

  Este proceso tiene como entrada ocho imágenes de parones de líneas en código gray directo y ocho del invertido tanto del plano como del objeto

según sea el caso, estas imágenes se convierten a escala de grises luego se binarizan por el método de Marján Trobina, para luego sumarlas y generar la Matriz de Proyección del plano o del objeto. En la figura 5.24 se ejemplifica la prueba de unidad que se llevo a cabo, cabe destacar que los procesos del 1 al 5, se realiza ocho veces para luego generar la salida.

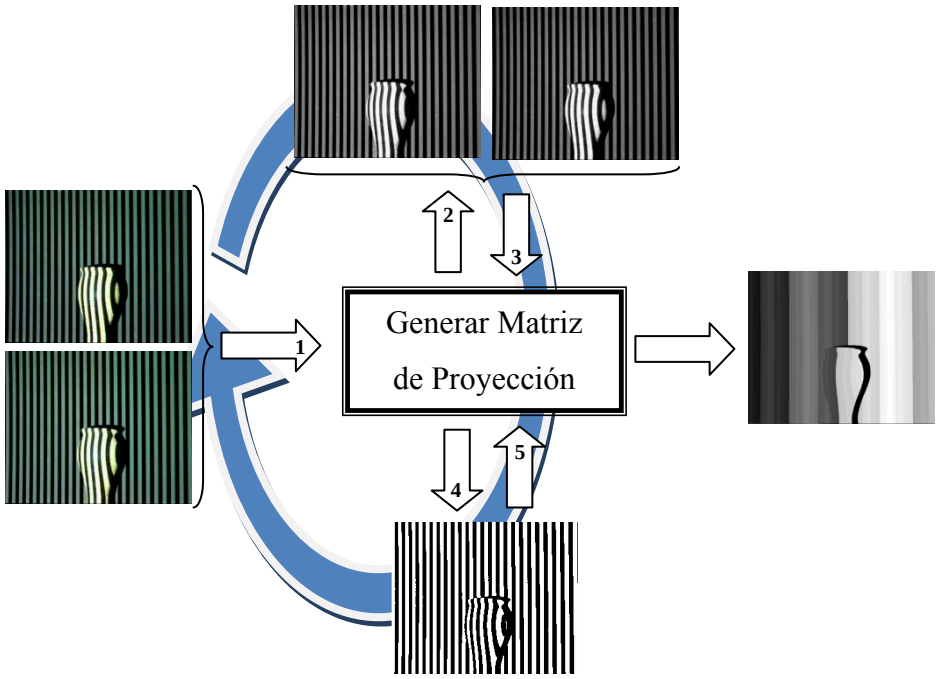


Figura 5.24: Pruebas de unidad del componente ReconstruirObjeto.java (proceso GenerarMatrizDeProyeccion).

Fuente: Los Autores.

- Proceso *GenerarMatrizDeCorrimiento*

Este proceso tiene como entrada la *MatrizDeProyeccionPlano* y la *MatrizDeProyeccionObjeto* y la *máscara* para generar la

MatrizDeCorrimiento. En la figura 5.25 se ejemplifica la prueba de unidad que se llevo a cabo.

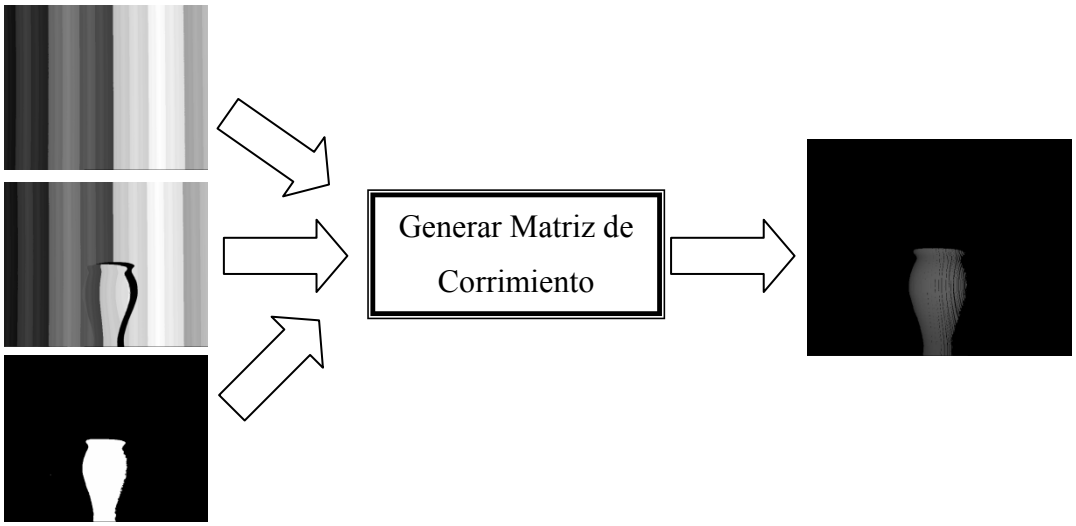


Figura 5.25: Pruebas de unidad del componente ReconstruirObjeto.java(proceso GenerarMatrizDeCorrimiento).

Fuente: Los Autores.

- Proceso *ReconstruirObjeto*

  Este proceso tiene como entrada la *MatrizDeCorrimiento,* y los parámetros externos del sistema para así generar la nube de puntos del objeto escaneado. En la figura 2.26 se ejemplifica la prueba de unidad que se llevo a cabo.

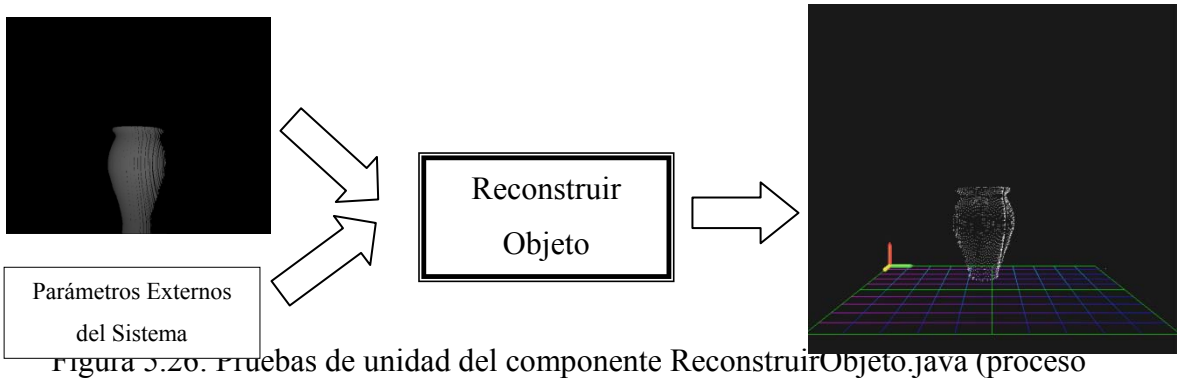


Figura 5.26. Pruebas de unidad del componente ReconstruirObjeto.java (proceso ReconstruirObjeto).

Fuente: Los Autores.

5.9.5.2 Pruebas de Integración

Las pruebas de integración consisten en evaluar la interacción entre los componentes una vez estos han sido integrados.

Luego de probados los componentes individualmente se realizaron estas pruebas para comprobar que estos funcionan correctamente como un todo.

Las pruebas de integración son necesarias ya que con estas se pueden detectar errores que no aparecieron en las pruebas de unidad los cuales posterior mente serán corregidos. Todo esto con la finalidad de lograr una versión del producto de software.

En el diagrama de la figura 5.27 se muestra la integración de cada uno de los componentes.

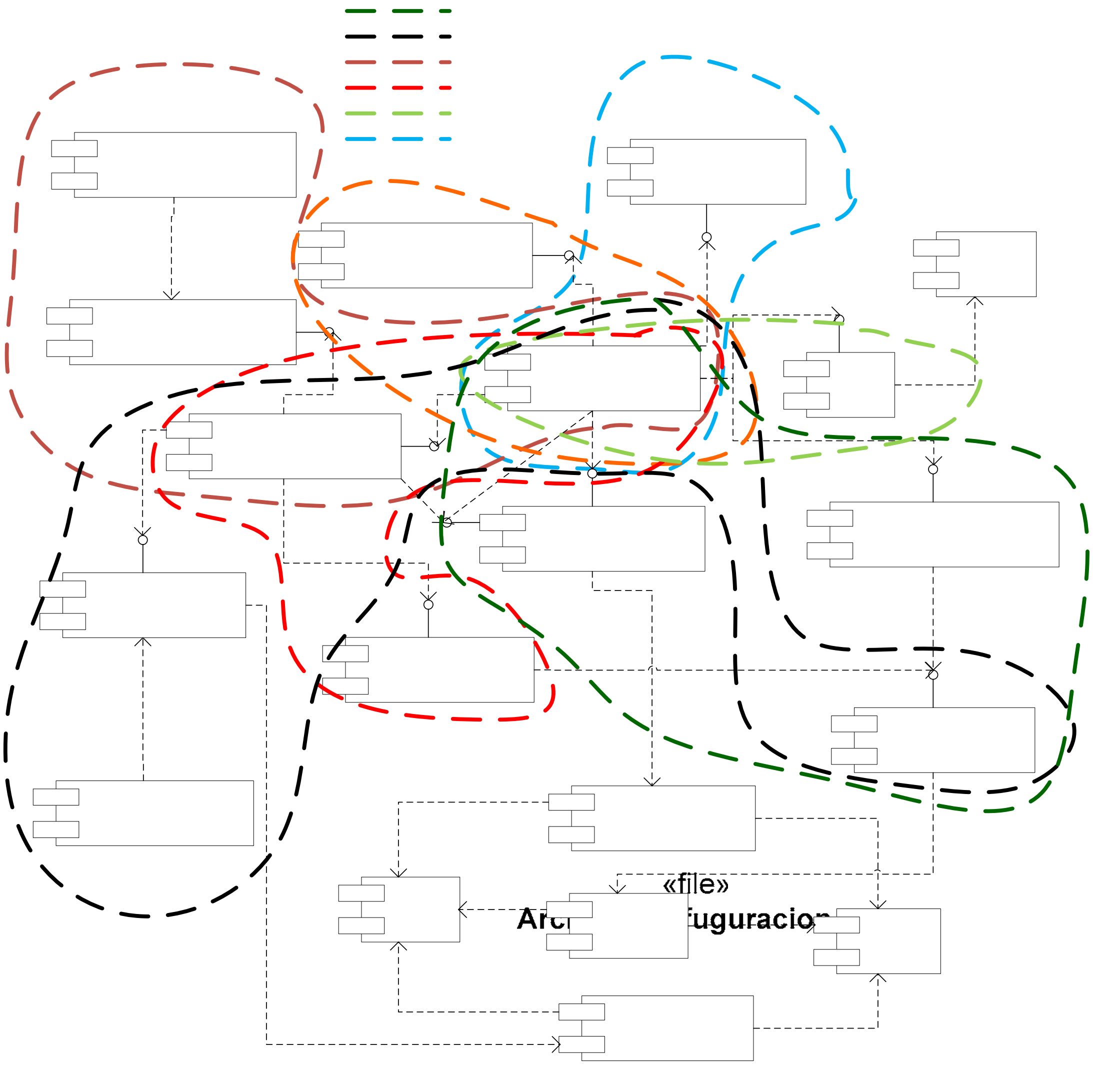


Figura 5.27: Diagrama de integración del sistema.

Fuente: Los Autores.

5.9.5.2.1 Pruebas integración de la fase 2

Los componentes involucrados en esta prueba fueron: *VentanaPrincipal.java, PanelConfiguracion.java, CalibrarCamara.java, CapturarImagen.Java, ParametrosCamara.txt.*

Se observo la interacción entre los componentes involucrados en esta fase, se verifico su correcto funcionamiento y se compararon los resultados arrojados en esta prueba, con los resultados obtenidos en las pruebas de unidad.

Surgieron problemas al seleccionar valores incorrectos con respecto al número de esquinas internas en la plantilla de calibración en los campos "cant de esq en X", "cant de esq en Y", ya que la cantidad de esquinas encontradas por el algoritmo no coincidía con la cantidad de esquinas introducidas por el usuario. Los problemas fueron resueltos exitosamente.

En el diagrama de la figura 5.28 se muestran las pruebas de integración de estos componentes.

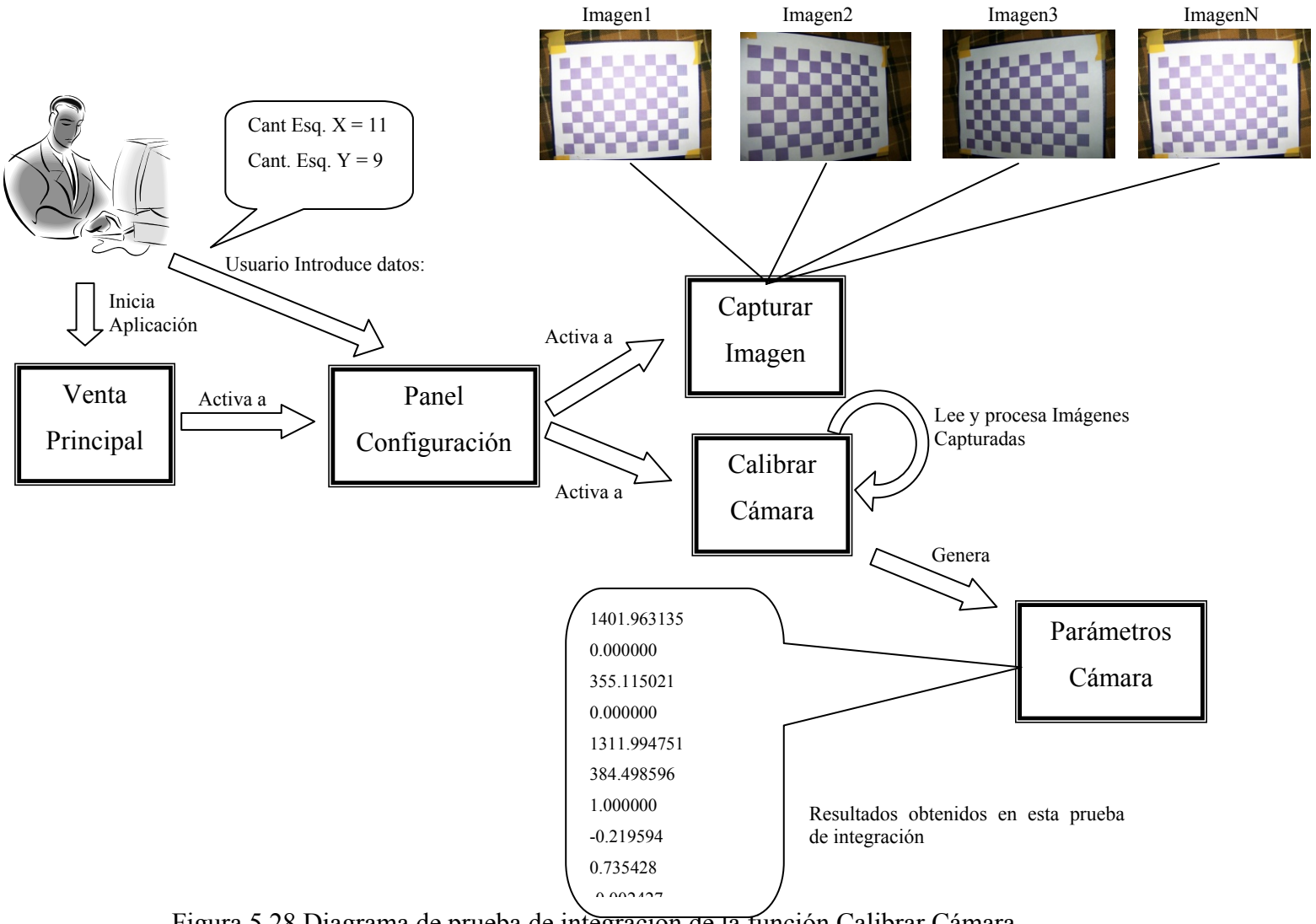


Figura 5.28 Diagrama de prueba de integración de la función Calibrar Cámara.

Fuente: Los Autores.

5.9.5.2.2 Pruebas de integración de la fase 1

Los componentes involucrados en esta prueba fueron: *VentanaPrincipal.java, ReconstruirObjeto.java, GeneradorDePatrones.java, CapturarImagen.java, ArchivoConfuguracion.txt, GUI3D.java, Obj3D.obj.*

Hubo un problema con la integración de estos componentes en el momento en que se levantaba la ventana donde se generan los patrones de código grey simultanea mente se iniciaba el proceso de reconstrucción si esperar que todas la imágenes fuesen tomadas, trayendo como consecuencia inestabilidad en la aplicación y su cierre inmediato.

Los problemas que surgieron se solucionaron, y con esto logrando una ejecución eficiente del proceso.

En el diagrama de la figura 5.29 se muestran las pruebas las pruebas de integración de estos componentes

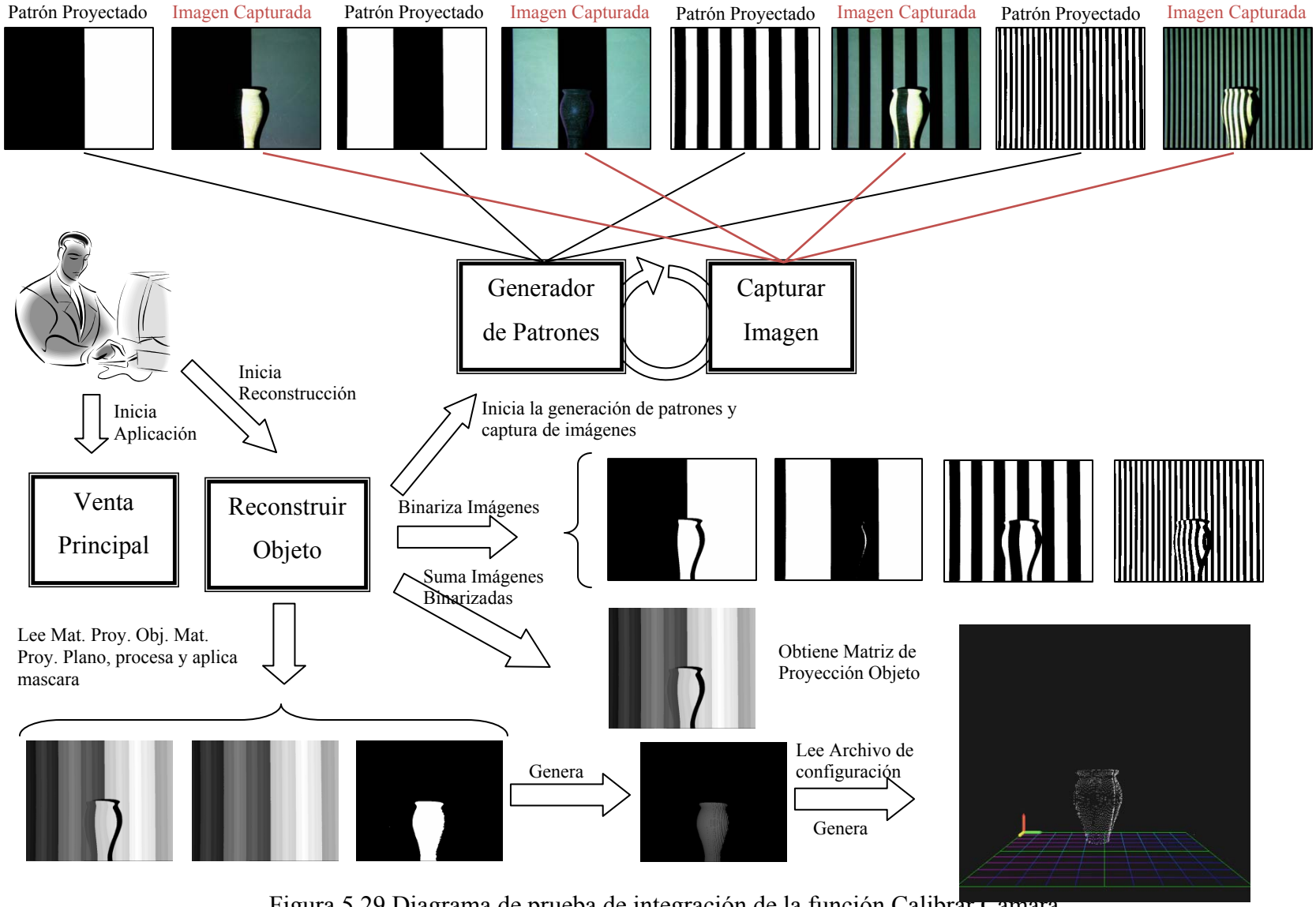


Figura 5.29 Diagrama de prueba de integración de la función Calibrar Cámara.

Fuente: Los Autores

5.10 CONCLUSIONES

A lo largo de esta fase se logró determinar los aspectos más importantes para la construcción del software. Estos aspectos fueron la escogencia del tipo de dato y el desarrollo de los diversos algoritmos que permitieron que el software funcione a cabalidad.

En el desarrollo de esta fase se llevaron a cabo las pruebas de unidad e integración. Estas pruebas permitieron evaluar el funcionamiento de los módulos desarrollados por separado y el funcionamiento de los módulos una vez integrados.

En esta se presentaron problemas al momento de integrar los módulos de software, estos problemas fueron solventados realizando una evaluación precisa de cada algoritmo.

Solo los problemas mencionados anteriormente fueron los observados durante esta fase, se solucionaron y se cumplieron a todos los objetivos propuestos.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

1. Con la fase de inicio se obtuvo el ámbito del sistema propuesto a través del modelo de dominio, se estableció una arquitectura candidata e identifico parte de los riesgos críticos del sistema CICLOPE.

2. En la fase de elaboración se recopilaron los requisitos del sistema que no aparecieron en la fase de inicio, se logro establecer una arquitectura solida para el sistema CICLOPE.

3. En la fase elaboración se explicaron en detalle los diferentes modelos de análisis (diagramas de clases de diseño, diagramas de clases de colaboración), obteniendo el diagrama de clases de diseño de esta fase.

4. A partir del diagrama de clases de diseño se obtuvieron los diagramas de secuencia del sistema que representan un funcionamiento ideal de este.

5. En la fase de elaboración se obtuvo una primera implementación del sist perteneciente al diagrama de componentes de esta fase.

6. La fase de construcción fue una de las más complejas ya que se presentaron problemas de diferente índole.

7. En la fase de construcción se obtuvo el diagrama de componentes definitivo del sistema, basado en el diagrama de clases de diseño y los diagramas de clases de análisis, obtenido en la fase de elaboración.

8. Los objetos brillantes o superficies reflectantes afectan la precisión en la reconstrucción de los objetos.

9. Con respecto a la superficie de proyección se escogió de color negro para lograr un buen contraste entre el plano y los objetos, y así de esta manera obtener una máscara de buena calidad.

10. Fue necesaria la utilización de una cámara fotográfica de buena calidad ya que las cámaras web probadas eran muy sensibles por la luz proyectada sobre el objeto.

11. Fue necesario disminuir todo el brillo en el proyector ya que el ccd (siglas en inglés de charge-coupled device: dispositivo de cargas interconectadas) de la cámara respondía de forma no deseada.

12. La excesiva luz ambiental contribuyo a generación de ruido en el modelo por lo que se tuvo que usar poca luz ambiental a la hora de escanear un objeto.

13. El sistema CICLOPE cumplió a cabalidad con todos los objetivos planteados inicialmente.

6.2 RECOMENDACIONES

1. Utilizar equipos que proporcionen gran velocidad de procesamiento, ya que el tratamiento de imágenes se lleva a cabo a través de una gran cantidad de algoritmos matemáticos.

2. Utilizar cámaras fotográficas que tengan una buena resolución de video.

3. Utilizar proyectores con una buena resolución de video y buen contraste.

4. Utilizar tarjetas capturadoras de video que proporcionen una resolución no menor a los 800x600 mp.

5. Se recomienda un ambiente con poca luz para así obtener reconstrucciones de buena calidad.

6. Para la calibración de la cámara, se recomienda que sean impresas en una impresora de buena calidad, y las esquinas de los cuadros de la plantilla estén bien definidas. Estas deben ser impresas en papel blanco y cuadros en color negro.

7. Se recomienda que la cantidad de imágenes tomadas de la plantilla de calibración de cámaras no debe ser menos a diez (10) imágenes, de esta forma garantizamos mejores resultados.

8. Se recomienda la lectura de manual de usuario antes de usar el sistema para así tener una mejor compresión del funcionamiento del mismo

9. Incentivar el desarrollo de proyectos de este tipo por parte de los profesores y estudiantes del departamento de Computación y Sistemas. Parte de este incentivo debe provenir de las autoridades del núcleo dotando de buenos laboratorios y estructura física para investigación tanto para profesores como para estudiantes del departamento de computación y sistemas.

10. Este proyecto de investigación puede ser continuado en el área de computación geométrica para la elaboración del mallado del objeto, en el área de computación grafica para el texturizado y color del objeto y algunas otras áreas de interés.

BIBLIOGRAFIA

[1.]Llaraza W. y Bravo J. (2000) **"Desarrollo de un software para obtener un ambiente virtual partiendo de las primitivas geométricas extraída de fotos estáticas utilizando técnicas de procesamiento de imágenes".**

[2.]Romero L., Meneses J., Rodríguez A., Gonzáles O. (2006) **"Sistema de reconstrucción tridimensional para el análisis dinámico de un cuerpo: estudio cuantitativo del vulcanismo de lodo".**

[3.]Patiño A., Miranda D., Meneses J. (2003) **"Escáner 3D de objetos a 360° de observación".**

[4.]Blanes F., Jiménez M., Puerto R., Ñeco P., Reinoso O. (2007) **"Reconstrucción tridimensional de escenas con un par estereoscópico de cámaras"**

[5.]Cerca M., Barrientos B., García J., Hernández C. (2007) **"Obtención del relieve digital mediante proyección de luz estructurada en modelos analógicos de extensión"**

[6.]Loncán Salazar Pierre Paul (2000) **"Estructura de la Información"** http://cursa.ihmc.us/rid=1201385077671_1590506156_7659/InteligenciaArtificial.pdf.

[7.] Wikipedia la enciclopedia libre **"Visión Artificial"** http://es.wikipedia.org/wiki/Computer_Vision

[8.] Reconstrucción 3D **"Introducción a la Reconstrucción 3D"** http://www.elai.upm.es/spain/Investiga/GCII/personal/lrodriguez/web3D/reconstruccion_3d.htm

[9.] Wikipedia la enciclopedia libre **"Escáner".** http://es.wikipedia.org/wiki/Escaner

[10.] Wikipedia la enciclopedia libre **"Escáner 3D".** http://es.wikipedia.org/wiki/Escaner_3D

[11.] Wikipedia la enciclopedia libre **"Pixel".** http://es.wikipedia.org/wiki/Pixel

[12.] Platero C. (2008) **"Introducción a la Visión Artificial"** http://www.elai.upm.es/spain/Asignaturas/Robotica/ApuntesVA/cap1IntroVA.pdf

[13.] Luz Estructurada **"Introducción a la luz estructurada"** http://www.elai.upm.es/spain/Investiga/GCII/areas/luz estructurada.htm.

[14.] Joyanes L., (1998), **"Programación orientada a objetos. 2.ª ed."** Editorial Editorial McGraw-Hill.

[15.] Wikipedia la enciclopedia libre **"Vision Artificial o por Computadora".** http://es.wikipedia.org/wiki/Visi%C3%B3n_artificial.

[16.] Aula Click **"Imagenes"** http://www.aulaclic.es/html/t_5_1.htm#a1

[17.] Wikipedia la enciclopedia libre **"Compresion de Archivos"** http://es.wikipedia.org/wiki/Compresi%C3%B3n_de_datos

[18.] Beltrán D., Basañez L.,(2008) **"Técnicas y algoritmos para la adquisición, transmisión y visualización de escenas 3D"** Universidad Politécnica de Catalunya, España.

[19.] Ricolfe C., (2006). **"CARACTERIZACIÓN Y OPTIMIZACIÓN DEL PROCESO DE CALIBRADO DE CÁMARAS BASADO EN PLANTILLA BIDIMENSIONAL"** INSTITUTO DE AUTOMÁTICA E INFORMÁTICA INDUSTRIAL.

[20.] Amaya D., Valderrama Z., (2003). **"RECONSTRUCCIÓN 3D DE OBJETOS A PARTIR DEL MÉTODO DE CÓDIGO DE GRIS"** Universidad Industrial De Santander

ANEXO A “MANUAL DE USUARIO”

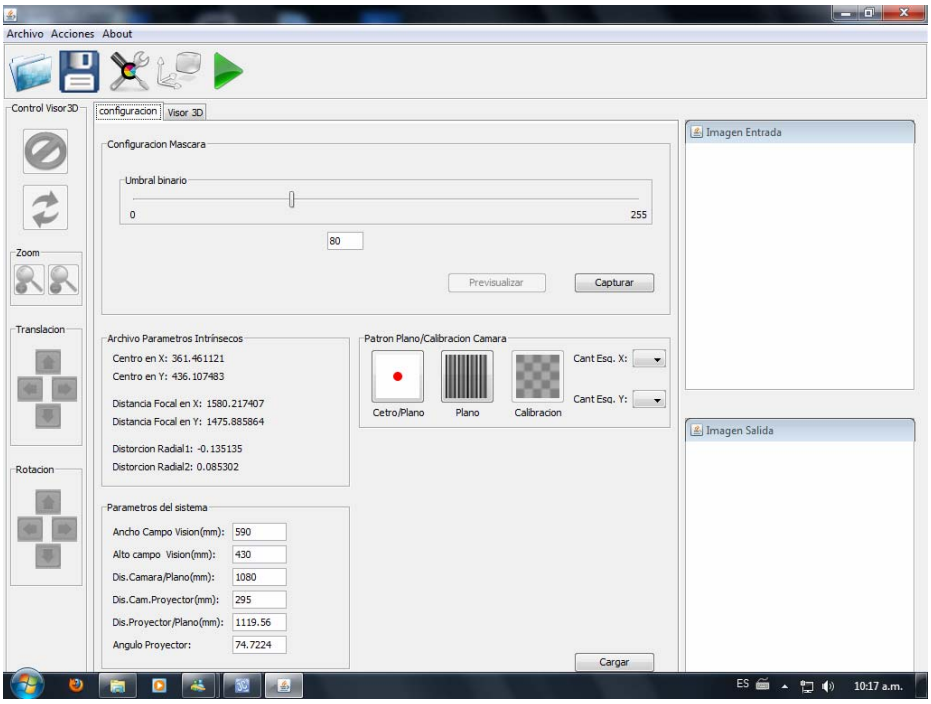
A.1 DESCRIPCIÓN DEL SOFTWARE CICLOPE

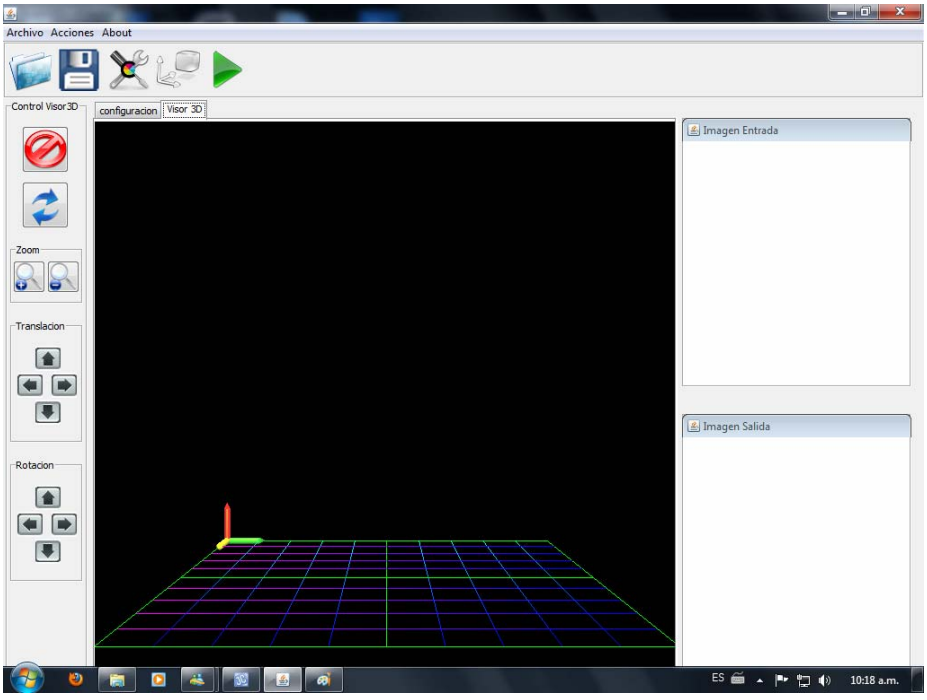
Ciclope es un software capaz de reconstruir objetos 3D según su forma dentro de un ambiente definido, aplicando técnicas de visión artificial. El software se encarga de generar patrones a proyectar por un video beam sobre el objeto a reconstruir y a su vez recibe información a través de una cámara en forma de imágenes del objeto con los patrones proyectados sobre él, esto con la finalidad de calcular y determinar su forma, la cual podrá ser visualizada por el usuario en forma de una nube de puntos a través de un visor 3D proporcionado por software, además de permitir almacenar la nube de puntos en formato .OBJ.

A.2 PANTALLA PRINCIPAL

El software Ciclope, le permite al usuario manipular fácilmente todo el proceso a través de su interfaz basada en menús, barras de herramientas y botones, los cuales ofrecen todas las opciones del sistema.

Su interface principal está formada por un menú principal que contiene todas las alternativas disponibles para el usuario, una barra de tareas que ofrece acceso rápido al usuario a las opciones más importante del sistema, un panel de configuración el cual contiene botones, cajas de textos, que nos permiten una fácil configuración del sistema previo a la reconstrucción, dos ventanas internas que nos permite visualizar el objeto a reconstruir, una interfaz 3D que nos permite visualizar la nube de puntos del objeto reconstruido, y un panel con botones que permiten la interacción del usuario con la interfaz de visualización 3D de la nube de puntos.


Archivo Acciones About
Control Visor3D
configuracion
Visor 3D
Configuracion Mascara
Umbral binario
0
255
80
Previsualizar
Capturar
Zoom
Translacion
Rotacion
Archivo Parametros Intrinsecos
Centro en X: 361.461121
Centro en Y: 436.107483
Distancia Focal en X: 1580.217407
Distancia Focal en Y: 1475.885864
Distorcion Radial1: -0.135135
Distorcion Radial2: 0.085302
Patron Plano/Calibracion Camara
Cetro/Plano
Plano
Calibracion
Cant Esq. X:
Cant Esq. Y:
Parametros del sistema
Ancho Campo Vision(mm): 590
Alto campo Vision(mm): 430
Dis.Camara/Plano(mm): 1080
Dis.Cam.Proyector(mm): 295
Dis.Proyector/Plano(mm): 1119.56
Angulo Proyector: 74.7224
Cargar
Imagen Entrada
Imagen Salida
ES
10:17 a.m.


Archivo Acciones About
Control Visor3D
configuracion
Visor 3D
Zoom
Translacion
Rotacion
Imagen Entrada
Imagen Salida
ES
10:18 a.m.

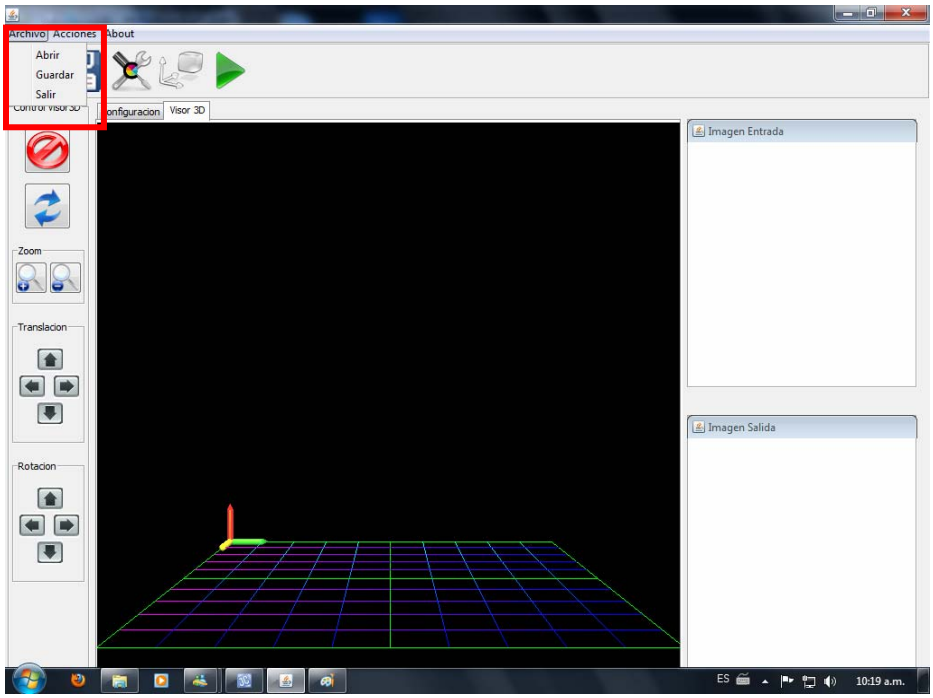
A.2.1 Menú

El menú principal consta de tres opciones que agrupan las tareas que pueden llevar a cabo el software de acuerdo a sus funciones, en este menú se encuentra:

- Menú Archivo.
- Menú Acciones.
- Menú acerca de.

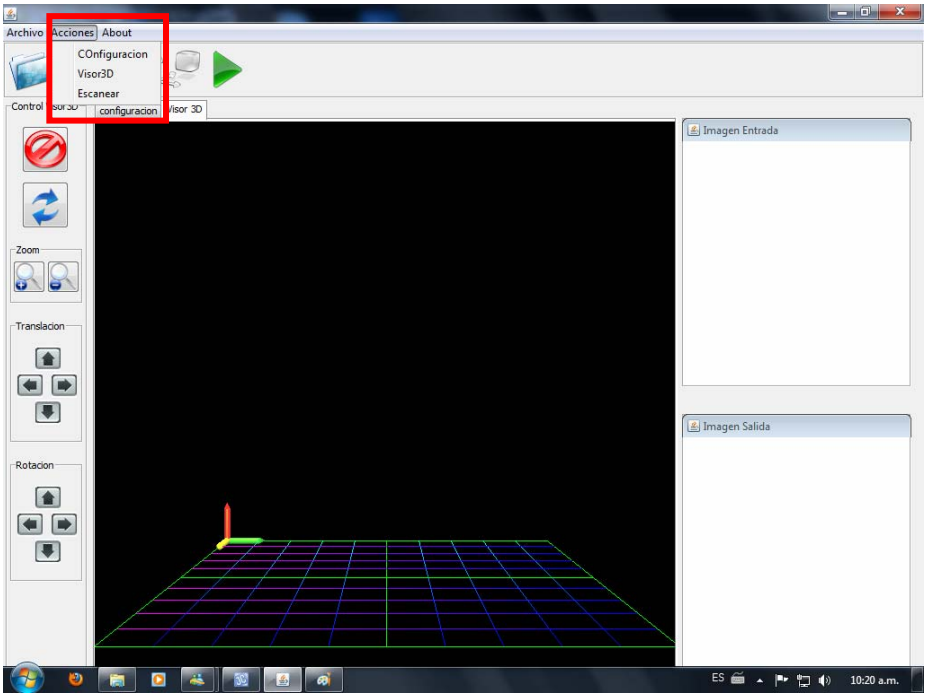
A.2.2 Menú Archivo

El menú archivo permite acceder a eventos básicos del software como lo son: Abrir la cual nos permite abrir un archivo .obj previamente almacenado, Guardar este evento nos permite almacenar el archivo de reconstrucción de un objeto en formato .obj, y el evento Salir que nos permite salir del software.



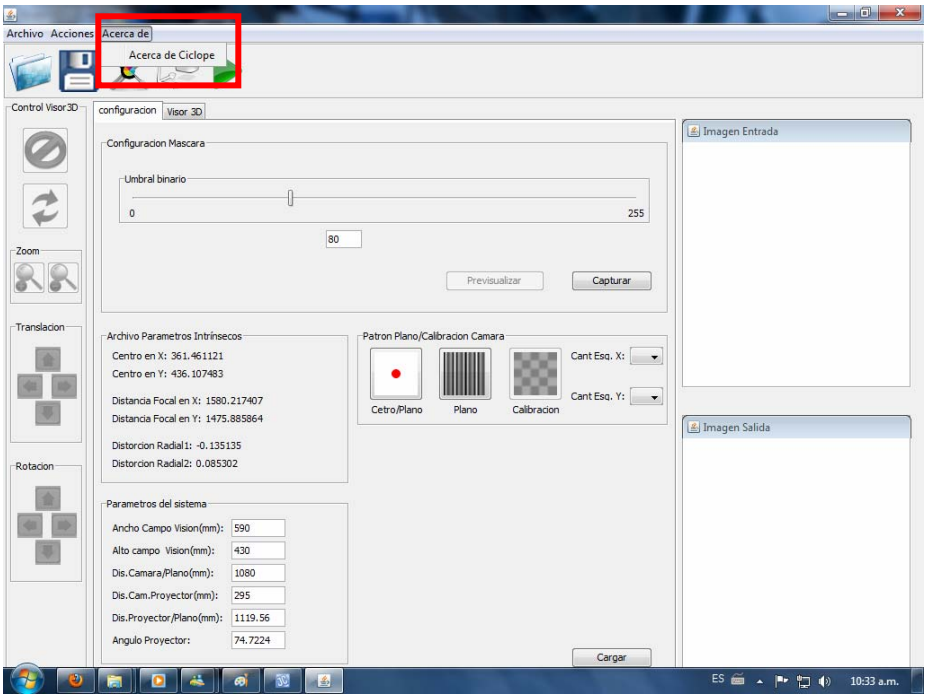
A.2.3 Menú Acciones

El menú permite al usuario acceder a las acciones principales del software, a través de este puede acceder al panel de configuración o al visualizador 3D, además puede ejecutar la acción de calibración de la cámara y la de ejecutar la reconstrucción de un objeto.



A.2.4 Menú Acerca de

Muestra información referente al software.

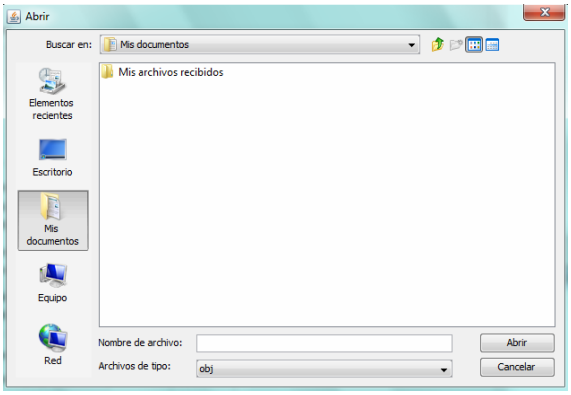


A.2.5 Barra de herramientas

Esta nos permite acceder a acciones del software de forma rápida a través de botones, estas acciones son: Abrir, Guardar, Configuración, Visor3D, Escanear.

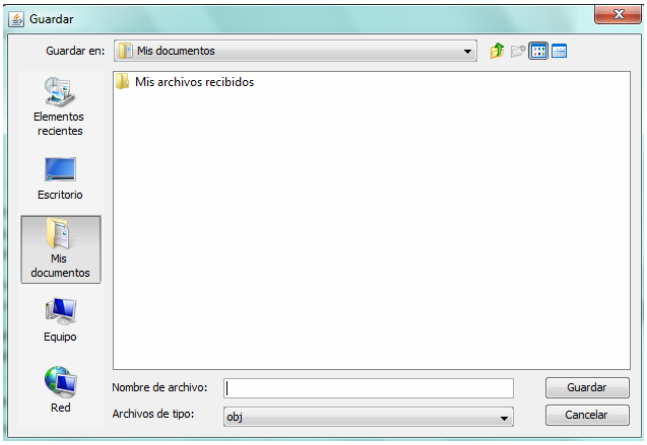
A.2.6 Botón Abrir

Este nos da acceso a una ventana de búsqueda de archivos .obj, para ser visualizados en Visor3D



A.2.7 Botón Guardar

Este da acceso a una venta de búsqueda de carpeta donde almacenar el archivo .obj de la reconstrucción 3d de un objeto.



A.2.8 Botón configuración

Este botón da acceso rápido a el panel de configuración de la ventana principal en caso de que este en segundo plano.

A.2.9 Botón Visor3D

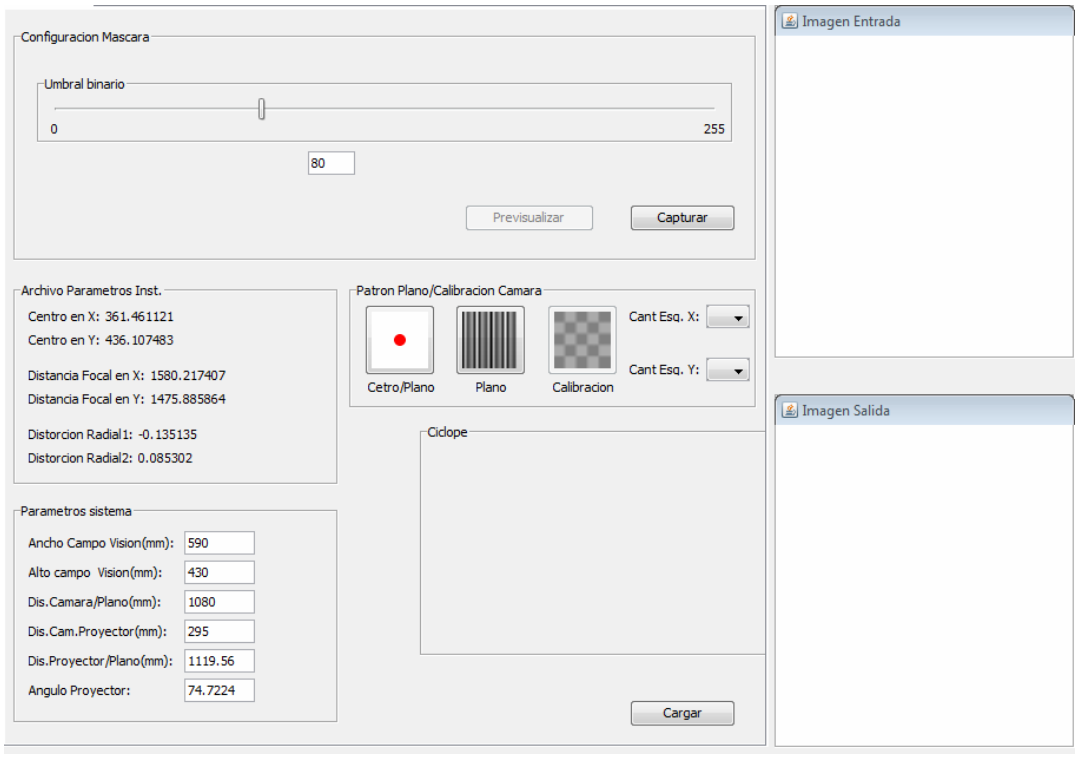
Este da acceso a la interfaz de visualización de la nube de punto 3D en caso de que se encuentre en segundo plano.

A.2.10 Botón Escanear

Este da inicio al proceso de reconstrucción 3d de un objeto.

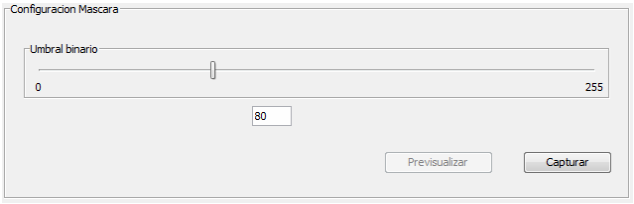
A.3 PANEL CONFIGURACIÓN

El panel de configuración le da acceso al usuario a todas las acciones previas y necesarias para la reconstrucción 3D de un objeto, esto lo hace a través de paneles compuestos de botones, caja de texto, barras deslizantes y caja de selección



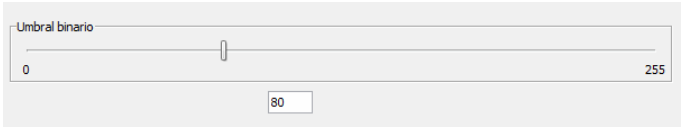
A.3.1 Panel Configuración Macara

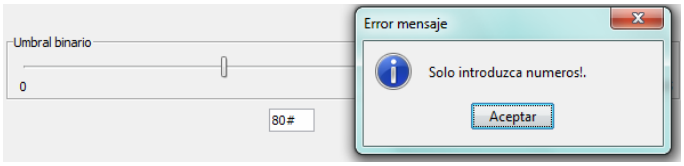
Es un panel perteneciente al panel de configuración, el cual se encuentra compuesto por barra deslizante, caja de texto, botones, y su función es la de brindar al usuario la opción de configurar el valor de umbral binario de la máscara, los cambios hecho por este será visualizados a través de las ventanas internas.



A.3.1.1 Barra Deslizante

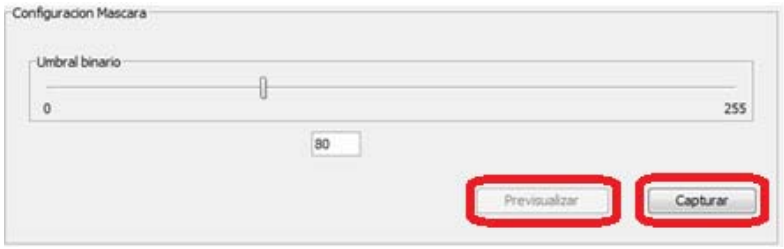
Con esta el usuario puede asignar un valor de umbral binario a la configuración de la máscara o introducir el valor directo en la caja de texto, el valor a introducir por el usuario tiene un rango de 0 a 255, en caso de que el valor introducido en la caja de texto sea mayor a 255 este se ajustara automáticamente a el valor máximo de la barra deslizante (255), en caso de que en la caja de texto sean introducidos letras o caracteres especiales la entrada no será procesada, y se advertirá al usuario que de ingresar solo números a través de una ventana de dialogo.





A.3.1.2 Botones capturar y Previsualizar

A través del botón capturar, se le indica al software que debe generar un patrón blanco a ser proyectado por el video beam que será capturado por la cámara, esto producirá una imagen del objeto a reconstruir iluminada necesaria para producir una máscara, la imagen capturada será visualizada a través de la ventana interna "Imagen entrada". Una vez obtenida la imagen de entrada para la máscara podremos observa en la ventana interna "Imagen salida" los cambios de umbral binario introducidos por medio de la barra deslizante presionando el botón previsualizar.

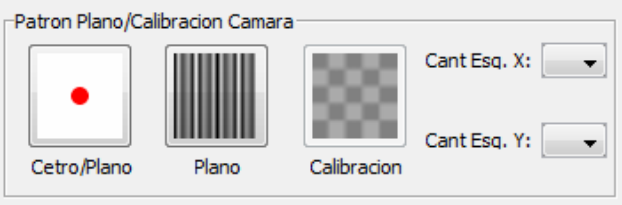


A.3.2 Panel Archivo de parámetros Intrínsecos

Es un panel perteneciente al panel de configuración, en este se visualiza los parámetros intrínsecos de la cámara luego de haber sido calibrada la cámara.

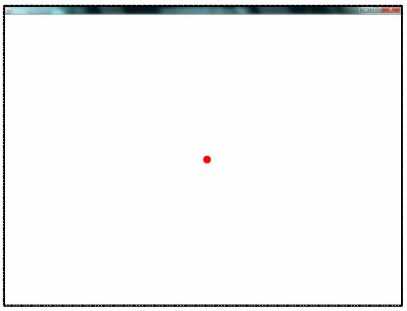
A.3.3 Panel Patrón Plano/Calibración cámara

Es un panel perteneciente al panel de configuración, en el se encuentran botones y cajas de selección, que le permiten al usuario, ejecutar acciones que forman parte de la configuración previa a la reconstrucción del objeto 3d, estas acciones son calibrar cámara, generar patrones plano, centro plano.



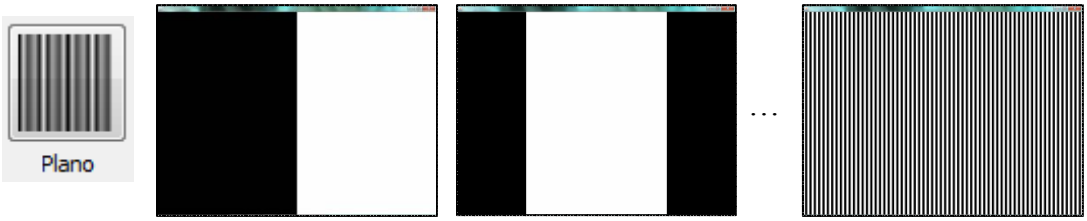
A.3.3.1 Botón Centro/Plano

Este botón perteneciente al panel Patrón Plano/Calibración cámara, ejecuta una interfaz de un patrón blanco con un círculo rojo en el centro, necesario para la tomas de parámetros externos del sistema.



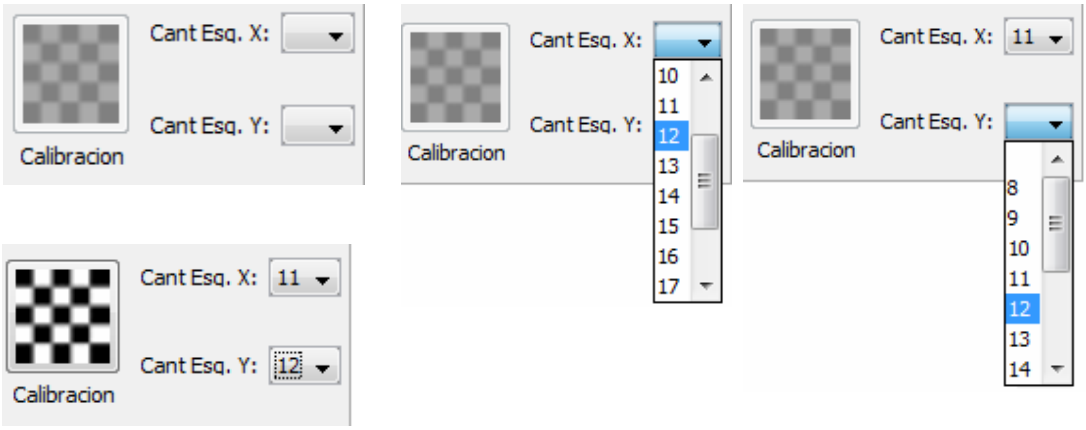
A.3.3.2 Botón Plano

Este botón perteneciente al panel Patrón Plano/Calibración cámara, ejecuta un interfaz donde se muestran en forma secuencial de patrones de líneas en código gray, para el plano de referencia, necesarios para generar la matriz de proyección del plano.



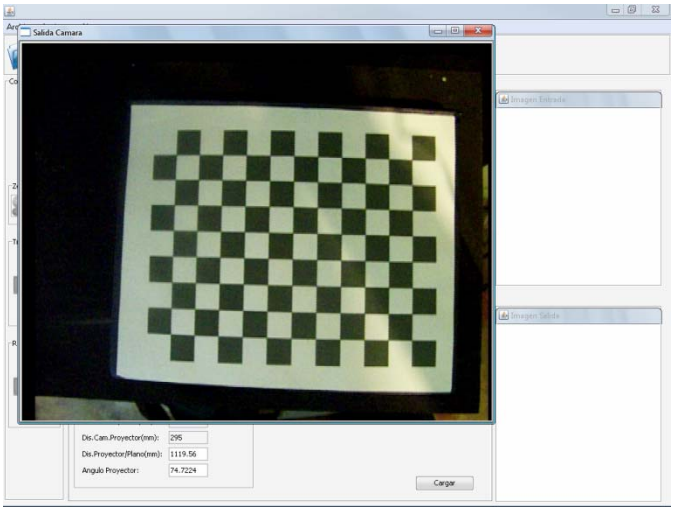
A.3.3.3 Botón Calibración

Este botón perteneciente al panel Patrón Plano/Calibración cámara, da acceso a la interface de captura de patrones de calibración para la cámara, cabe acotar que este botón se encuentra deshabilitado, en tanto no se escoja un valor relacionado con las cantidad de esquinas en las coordenadas "X" y "Y" de la plantilla de calibración, a través de las cajas de selección cantidad esquinas X y cantidad esquinas Y.



A.3.3.4 Interfaz Calibración Cámara

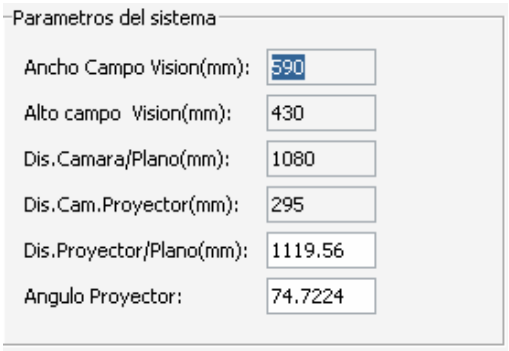
Esta es activada a través de el botón calibración perteneciente al panel "Patrón Plano/Calibración cámara" y el ítem "Calibrar cámara" en el menú "Acciones", a través de esta interfaz se visualiza la plantilla de calibración, para realizar la captura de imágenes de la plantilla de calibración, se utiliza el botón izquierdo del ratón y para cerrar la interfaz se utiliza el botón derecho del ratón, además esta genera unos mensajes en caso de los siguientes errores, cerrar la venta sin haber capturado ninguna imagen, o en caso de que el número de esquinas en "X" y "Y" del patrón de calibración no coincidan con el numero introducido por el usuario en las caja de selección del "Botón Calibración".



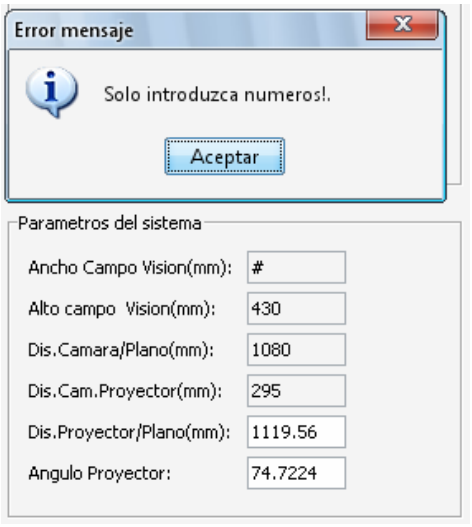
A.3.4 Panel Parámetros del sistema

Es un panel perteneciente al panel de configuración, en este se visualiza y se introducen, parámetros externos del sistema como ancho y alto del campo de visión,

distancia de la cámara al plano de proyección, distancia de la cámara al proyector, necesarios para la reconstrucción 3D. Este panel es compuesto por etiquetas y cajas de texto donde se introducen valores numéricos, encaso de introducir valores no

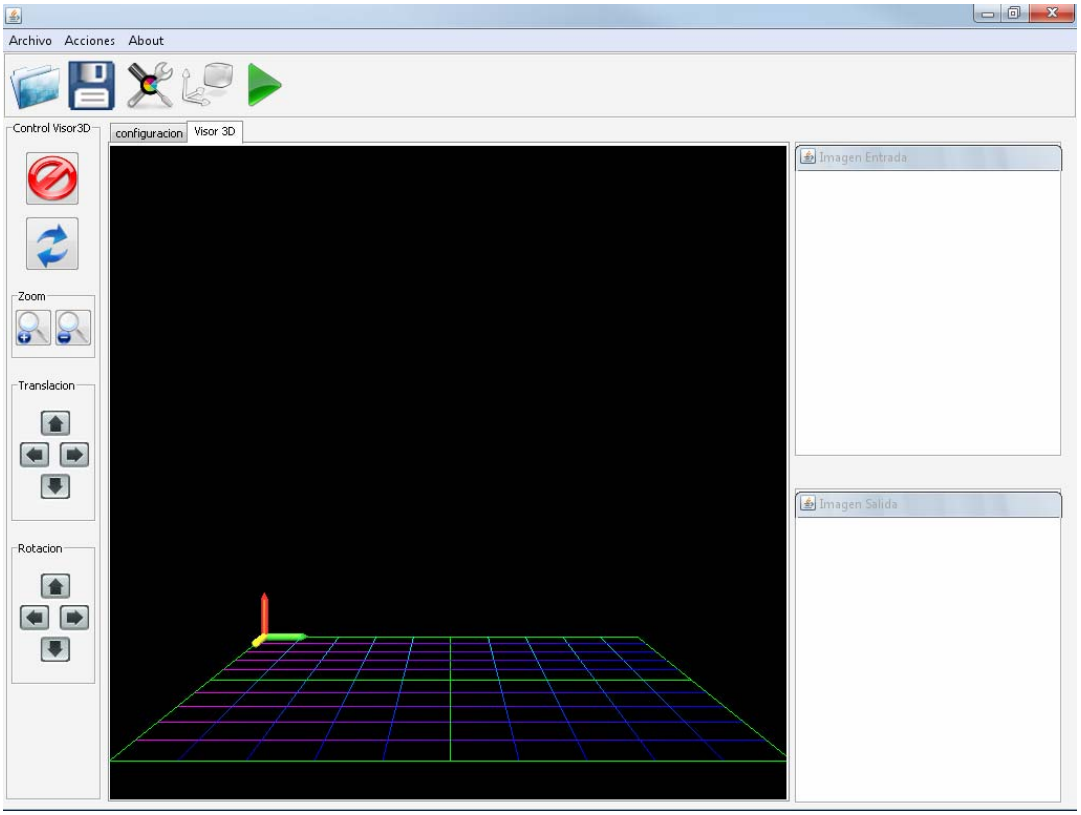


numéricos. Informa la usuario a través de una ventana de dialogo que debe introducir sol o valores numéricos.



A.4 VISOR3D Y CONTROL VISOR3D

El visor3D es donde se visualizan los objetos reconstruidos, este está provisto del panel de Control Visor3D, que se encarga de tareas como: restaurar, eliminar, rotar, Zoom, trasladar.



METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	**DESARROLLO DE UN SOFTWARE PARA RECONSTRUCCIÓN TRIDIMENSIONAL DE OBJETOS A PARTIR DE IMÁGENES, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL**
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
López Loroima Lenin Armando	**CVLAC: 13.914.069** **E MAIL:** leninll_1979@hotmail.com
Sandoval Rojas José Miguel	**CVLAC: 15.875.204** **E MAIL:** jmsandovalr@gmail.com
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALÁBRAS O FRASES CLAVES:

Visión Artificial.

Reconstrucción 3D

Luz Estructurada.

Procesamiento de Imágenes.

Calibración de Cámaras.

Desarrollo de Software.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÀREA	SUBÀREA
COMPUTACIÓN	VISION ARTIFICIAL
	RECONSTRUCCION 3D
	DESARROLLO DE SOFTWARE

RESUMEN (ABSTRACT):

El objetivo del trabajo de investigación titulado **"DESARROLLO DE UN SOFTWARE PARA RECONSTRUCCIÓN TRIDIMENSIONAL DE OBJETOS A PARTIR DE IMÁGENES, USANDO TÉCNICAS DE VISIÓN ARTIFICIAL",** es la adquisición de imágenes y modelado tridimensional de objetos utilizando técnicas de visión artificial mediante la luz estructurada. Para ello se desarrollo una aplicación que consiste básicamente en proyectar en un plano una secuencia de patrones de luz sobre un objeto determinado, estos son captados por una cámara, posteriormente dichas imágenes son enviadas al computador quien las procesara, Utilizando técnicas de triangulación se obtiene la profundidad del objeto a reconstruir esto permitirá la generación del modelo tridimensional. Para la calibración de la cámara se utilizo el método de calibración propuesto por Zhang. Para la implementación de la aplicación, se uso el lenguaje de programación JAVA para la construcción de las interfaces de usuario, y el lenguaje de programación C++ para los motores de reconstrucción 3D y calibración de cámara, junto con las librerías JNI para la comunicación entre JAVA y C++ y OPENCV para el tratamiento de imágenes.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Stefano Larese	ROL	CA	AS (X)	TU	JU
	CVLAC:				
	E_MAIL				
	E_MAIL				
Bastardo José Luis.	ROL	CA	AS	TU	JU (X)
	CVLAC:				
	E_MAIL				
	E_MAIL				
Claudio Cortines.	ROL	CA	AS	TU	JU (X)
	CVLAC:				
	E_MAIL				
	E_MAIL				
.	ROL	CA	AS	TU	JU
	CVLAC:				
	E_MAIL				
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

AÑO	MES	DÍA
2010	08	12

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
Tesis Reconstrucción 3D	application/msword

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: _____
(OPCIONAL)

TEMPORAL: _____
(OPCIONAL)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:
Pregrado
_____

NIVEL ASOCIADO CON EL TRABAJO:
Ingeniero
_____

ÁREA DE ESTUDIO:
Computación y Sistemas
_____

INSTITUCIÓN:

Universidad De Oriente Núcleo de Anzoátegui

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

De acuerdo al artículo 41 del reglamento del trabajo de Grado:

"Los trabajos de grado son de propiedad exclusiva de la Universidad de Oriente y sólo podrán ser utilizados a otros fines con el consentimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario".

López L. Lenin Armando	**Sandoval R. José Miguel**
AUTOR	**AUTOR**

Larese Stefano	**Bastardo José Luis**	**Cortines Claudio**
TUTOR	**JURADO**	**JURADO**

POR LA SUBCOMISION DE TESIS