BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Reconstrucción parcial 3D dinámica aplicada a robótica a través de algoritmos genéticos"

Tesis profesional presentada por

Mónica Pérez Meza

como requisito parcial

para obtener el título de maestro en ciencias

de la Computación

Puebla, Pue. Primavera de 2007

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Reconstrucción parcial 3D dinámica aplicada a robótica a través de algoritmos genéticos"

Tesis profesional presentada por

Mónica Pérez Meza

como requisito parcial

para obtener el título de maestro en ciencias de la Computación

Jurado Calificador

Dr. Manuel Isidro Martín Ortíz

Presidente

Dr. Rodrigo Montúfar Chaveznava

Vocal y Director

Dr. Ivo Humberto Pineda Torres

Secretario

Puebla, Pue. Marzo de 2007

Resumen

Particularmente, la navegación en robótica móvil es una actividad donde el robot necesita conocer en cada momento la estructura de su entorno para moverse con certidumbre en él, lo cual hace necesaria la reconstrucción de la escena. Uno de los métodos que resuelven la reconstrucción de una escena es la visión estéreo, que es un área específica de estudio dentro de la visión computacional.

El término estéreo en visión se utiliza cuando existe más de una vista de una escena. A través de varias imágenes de una escena, tomadas desde distintas localidades o posiciones, se puede tener una interpretación de las características tridimensionales de la escena en estudio.

En el desarrollo de este método se utiliza el análisis bifocal, el cual consiste de un sistema de visión con dos cámaras, las cuales capturan dos imágenes del objeto de estudio en posiciones diferentes.

En éste trabajo hemos propuesto realizar visión estéreo utilizando solo una cámara en lugar de dos, aplicando el algoritmo evolutivo conocido como algoritmo de las moscas modificado de acuerdo con el problema que atacamos, con la finalidad de efectuar una reconstrucción *3D* parcial y proponer su aplicación a la tarea de navegación en robótica móvil. Como método, desplazamos una única cámara a distintas posiciones conocidas y controladas, para captar imágenes en cada una de las posiciones y obtener la escena tridimensional parcialmente reconstruida a partir de éstas imágenes.

Índice general

Índice de Tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Descripción del problema

Una serie de algoritmos que pueden dar la solución al problema de la visión estéreo son los algoritmos evolutivos y genéticos. Por medio de la visión estéreo es posible obtener la profundidad de los objetos en una escena, y con este propósito se utilizan dos cámaras que capturan dos imágenes, a partir de las cuales es posible determinar la ubicación de los objetos en relación con un sistema de referencia, dando como resultado la obtención de la tercera dimensión de los objetos.

En este trabajo se emplea solo una cámara en lugar de dos, aplicando geometría proyectiva, conociendo los desplazamientos de dicha cámara y modificando el algoritmo de las moscas se logra la reconstrucción *3D* de una escena. La cámara se desplaza a diferentes posiciones conocidas, captando las imágenes en cada una de ellas y obteniendo una escena parcial en *3D*, pero suficiente para tener una interpretación del entorno.

El resultado obtenido es utilizable, entre otras aplicaciones, a la operación de navegación en robótica móvil, donde el robot necesita conocer en cada momento la estructura de la escena para navegar de manera segura.

La navegación es un problema en la robótica móvil, el cual consiste en construir un mapa del entorno desconocido para posteriormente localizar al robot dentro de dicho entorno como en [11].

1.2. Antecedentes del proyecto

La visión es el mecanismo sensorial de percepción más importante en el ser humano. Hace unas décadas, los investigadores de visión por computadora iniciaron el desarrollo de métodos que obtenían la profundidad mediante imágenes tomadas por cámaras o sensores, localizados en diferentes posiciones físicas.

La visión estéreo pretende conseguir una estructura tridimensional partiendo de dos o más imágenes bidimensionales. Según el número de imágenes que se emplee, hablaremos de visión bifocal -dos imágenes-, trifocal -tres imágenes-, y así sucesivamente. Un ejemplo de visión bifocal es aquel con el que contamos los humanos, pues para obtener la profundidad de los objetos partimos de dos lentes (la mirada de cada ojo) que convergen en el objeto.

La palabra estéreo, en el ámbito computacional, está ligada directamente con la recuperación de una escena tridimensional mediante el uso de múltiples imágenes. La obtención de las imágenes de la escena ligeramente desplazadas se puede alcanzar por alguno de los procedimientos siguientes:

- **La captura estática:** Alineando dos o más cámaras a cierta distancia entre sí una de la otra (como cada ojo), donde cada cámara se encarga de la captura de una imagen.

- **La captura dinámica:** Desplazando una sola cámara a lo largo de una trayectoria y capturando las imágenes en cada una de las diferentes posiciones del desplazamiento.

En el desarrollo de un sistema estéreo se tienen que considerar los siguientes aspectos:

- La geometría presente en el montaje de las cámaras, ya que éstas deben mantener sus ejes ópticos paralelos (en el caso de una sola cámara) o convergentes (en el caso de dos o más cámaras).
- La calidad de la imagen proporcionada.
- La iluminación presente en la escena.

La computación evolutiva usa modelos computacionales de procesos evolutivos como elementos claves en el diseño e implementación de sistemas para resolver problemas complejos. Los algoritmos evolutivos mantienen una población de estructuras que se ven alteradas por métodos de selección y otros operadores, como la cruza y la mutación. Cada individuo en la población recibe una calificación de aptitud en el ambiente; la selección se usa para mantener a los individuos con más alta aptitud en la siguiente generación, la cruza y la mutación modifican a los individuos, además proveen a heurísticas generales para la exploración.

Por lo anterior, se comenzaron a utilizar estos métodos no tradicionales para dar una solución alternativa a la reconstrucción tridimensional. A finales de los años noventa, Jean Louchet [14] utilizó una estrategia evolutiva individual combinada con técnicas de geometría proyectiva, para obtener la profundidad de los objetos presentes en una escena en un corto tiempo. Dentro de las estrategias evolutivas individuales, los individuos de la población forman parte de la solución. A éste algoritmo se le denomina “algoritmo de las moscas”.

El algoritmo de las moscas es una técnica para la exploración de valores en el espacio de los parámetros, enfocada a aplicaciones de reconocimiento de patrones. En la aplicación se describe la manera a la cual una población evoluciona; para nuestro caso, esta población que evoluciona viene a constituir la representación de una escena de tres dimensiones a partir de la utilización de las imágenes estéreo. Cada individuo representa un punto tridimensional en la escena, que puede ser ajustado con parámetros opcionales. La evolución es controlada por una función de ajuste que contiene todos los cálculos relacionados con la proyección de las moscas y los niveles de intensidad o color de los píxeles, empleando operadores clásicos evolutivos (como la cruza, la mutación y la inmigración).

1.3. Objetivos

1.3.1. General

Realizar una reconstrucción parcial *3D* de un ambiente dinámico empleando la teoría que se encuentra detrás de la visión estéreo en conjunto con las estrategias de convergencia de los algoritmos genéticos particularmente empleando un sistema de visión monocular o cíclope.

1.3.2. Específicos

- Reconstruir en *3D* un ambiente dinámico empleando algoritmos genéticos.
- Adecuar el algoritmo de las moscas acorde a nuestras condiciones, tareas u objetivos.
- Emplear visión monocular en vez de visión estéreo.

- Proponer los resultados para su aplicación en robótica móvil, en particular, para la tarea de navegación.
- Analizar los resultados para proponer mejoras al sistema y así poder extender su aplicación a otras áreas relacionadas con visión.

Capítulo 2

Marco teórico y de referencia

2.1. Modelo de proyección

El modelo geométrico de proyección *3D* a *2D* más utilizado es el modelo pin-hole. Los algoritmos que reconstruyen la estructura *3D* de una escena o calculan la posición de objetos en el espacio necesitan las ecuaciones que unen los puntos *3D* con sus correspondientes proyecciones *2D*, y aunque dichas ecuaciones vienen proporcionadas por la ecuación de proyección, es normal suponer que los puntos *3D* pueden venir dados con relación a un sistema de coordenadas distinta del proporcionado por el sistema de referencia de la cámara, siendo necesario además relacionar las coordenadas de un punto en la imagen con las coordenadas correspondientes en el sistema de referencia proporcionado por la cámara. Tenemos, por lo tanto, que conocer los parámetros extrínsecos (relacionan dos sistemas de referencia *3D*) los cuales son:

- Rotación en X, Y y Z.
- Traslación de X, Y y Z.

Y los intrínsecos (de la cámara a la imagen) los cuales son:

- El tamaño del píxel $(s_x \ y \ s_y)$.

- Posición del centro del plano imagen (u_0, v_0).
- Distancia focal *(f)*.
- Distorsión en x y distorsión en y.
- Tamaño de la imagen.

Esto es lo que se conoce como calibración de la cámara. Una descripción sencilla de problemas en los que la calibración es importante puede encontrarse en [12].

2.1.1. Modelo *pin-hole*

El modelo de cámara con el cual trabajamos es el modelo *pin-hole*. En él se asume que un punto tridimensional *P(X,Y,Z)* se proyecta en el plano de imagen pasando a través de un único punto llamado Centro óptico. La recta que une el punto P y el centro óptico se llama línea de proyección e intersecta al plano imagen justo en el píxel *p(x,y)* que es la proyección de *P(X,Y,Z)*. El centro óptico está situado a la distancia focal del plano imagen. Este modelo lo completan el eje óptico, que es una línea perpendicular al plano de imagen que atraviesa el centro óptico, y el plano focal, que es el plano perpendicular al eje óptico como podemos apreciar en la figura 2.1.

A partir de este modelo, empleando triángulos semejantes, obtenemos las ecuaciones que relacionan un punto en el espacio con su proyección en la imagen. Estas ecuaciones son:

$$x = f\frac{X}{Z} \tag{2.1}$$

$$y = f\frac{Y}{Z} \tag{2.2}$$

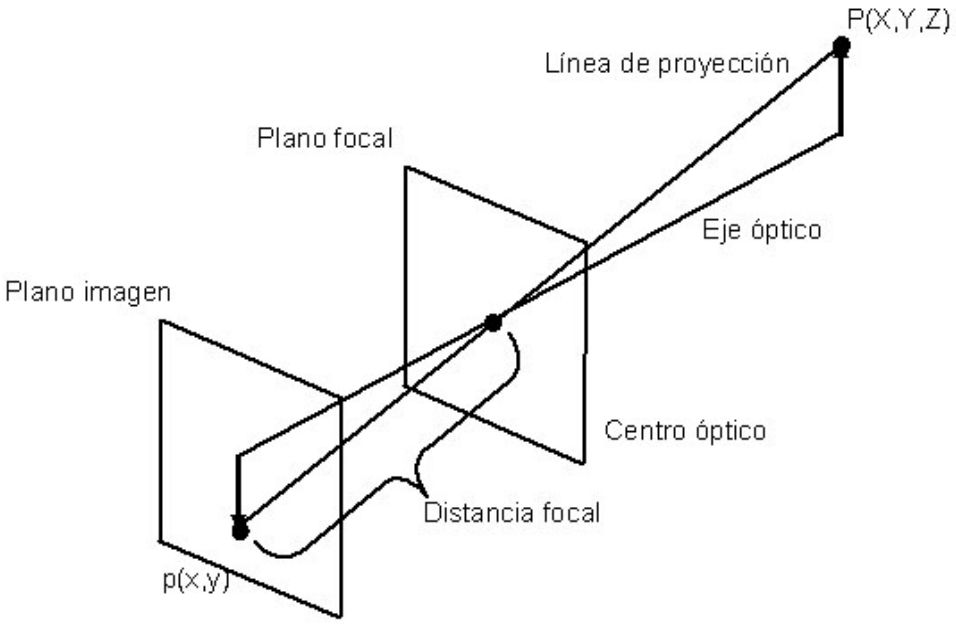

Figura 2.1: Modelo pin-hole.

2.2. Visión estéreo

A través de varias imágenes de una escena, tomadas desde distintas posiciones, se puede obtener las características tridimensionales de la escena en estudio. Gracias a esto podemos ver los objetos como sólidos en tres dimensiones espaciales con altura y profundidad [10][21].

La Visión estéreo o estereoscópica es el proceso en el que se emplea más de una imagen para obtener la tridimensionalidad. Por ejemplo, dadas dos vistas ligeramente desplazadas una de otra, éstas tendrán muchas cosas en común, pero cada una contendrá cierta información visual que la otra no tiene. A la diferencia entre ambas imágenes se le denomina disparidad.

Además, a la forma de percibir la sensación de profundidad, lejanía o cercanía de los objetos que nos rodean, a este proceso de fusión se denomina estereopsis.

2.2.1. Mecanismos de la estereopsis

Si observamos objetos muy lejanos, los ejes ópticos de nuestros ojos son paralelos.

Si observamos un objeto cercano, nuestros ojos giran para que los ejes ópticos queden alineados sobre él, es decir, convergen. A su vez se produce una acomodación o enfoque para poder ver nítidamente el objeto. A este proceso conjunto se le llama fusión.

2.2.2. Análisis bifocal

En el análisis bifocal se tiene un sistema de visión con dos cámaras, o bien una sola cámara (que es nuestro caso), que toma dos imágenes del objeto de estudio en dos tiempos distintos, suponiendo que en ese tiempo la cámara o el objeto se ha movido.

La geometría de dos vistas es conocida como la *Geometría Epipolar* y se muestra en la figura 2.2. Donde un punto *M (3D)* es proyectado en la imagen 1 como m_1 por medio de un centro óptico en este caso C_1 y en la imagen 2 como m_2 por medio del centro óptico C_2.

A partir de m_1, no se puede saber la ubicación exacta de M, ya que en el proceso de proyección se ha perdido la información de profundidad.

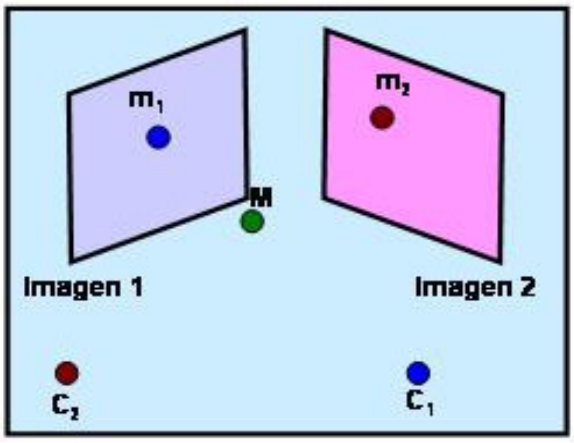

Figura 2.2: Geometría de dos vistas.

Sin embargo, en la figura 2.3 se muestra que M se encuentra en la línea que nace del centro óptico C_1, para proyectarse en m_1, es decir, que M pertenece a la recta $\overline{m_1, C_1}$, donde varios puntos (M incluido) pertenecientes a esta recta, pueden ser los que forman el punto m_1 en la imagen 1.

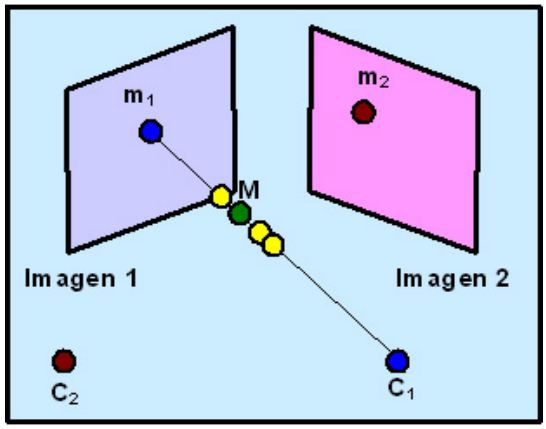

Figura 2.3: M pertenece a la recta $\overline{m_1, C_1}$.

Si a partir de m_1 se desea conocer la ubicación de m_2, es necesario proyectar en la imagen 2 los posibles puntos que puedan formar a m_2 (ver figura 2.4). Se observa que m_2 es uno de esos puntos proyectados, por lo tanto se puede afirmar que m_2 pertenece a la proyección de la recta $\overline{m_1, C_1}$ en la imagen 2, realizada por el centro óptico C_2. La proyección de esta recta, se denomina *línea epipolar* y se puede apreciar en la figura 2.5.

Cuando se trabaja con visión estéreo los principales problemas a resolver son: la calibración, la reconstrucción y la correspondencia.

La calibración es el proceso para determinar la geometría externa (posición relativa y orientación de cada cámara) e interna (longitud focal, centros ópticos, y distorsiones de las lentes) de las cámaras. La estimación precisa de esta geometría es necesaria para hacer el traslado del sistema de coordenadas del mundo a las coordenadas de las

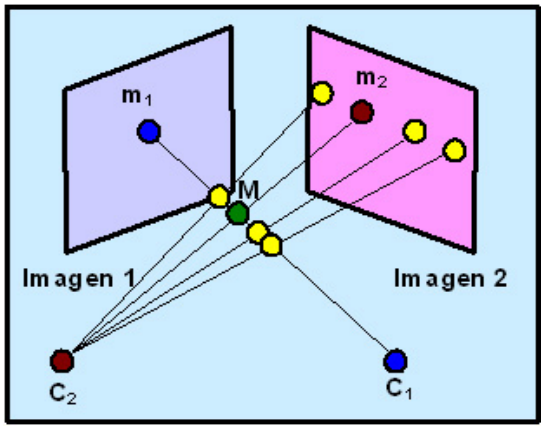

Figura 2.4: Localizando la ubicación de m_2.

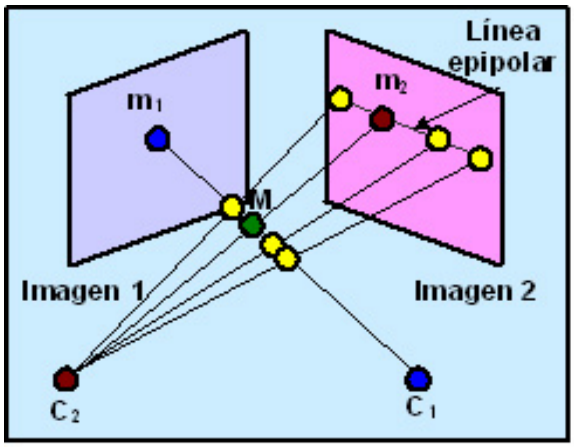

Figura 2.5: Geometría epipolar.

cámaras. Podemos clasificar dos tipos de metodologías que realizan esta tarea: Calibración fotogramétrica [23] y Auto-calibración [23].

El problema de la reconstrucción se encarga de determinar la estructura tridimensional de un mapa de disparidades, basado en la geometría de la cámara. A éste proceso se le conoce como triangulación.

El problema de correspondencia consiste en determinar la localización en cada imagen de un punto dado en el espacio. Los métodos basados en correspondencia buscan relacionar los píxeles en una imagen con sus contrapartes en la otra imagen. Existen dos enfoques: los métodos locales y los métodos globales. Los primeros usan un pequeño vecindario de píxeles para encontrar la correspondencia entre imágenes. Los métodos globales utilizan como parámetros de correspondencia regiones grandes o imágenes enteras.

2.3. Algoritmos evolutivos

Los algoritmos evolutivos manipulan individuos evaluados por medio de una función de ajuste, de una manera similar a la evolución biológica. El diagrama general de estos algoritmos se presenta en la figura 2.6.

Donde:

- La población es un grupo de individuos.
- Un individuo es definido por sus genes $X = (x_1, x_2, \ldots, x_n)^T$ usualmente en el espacio buscado.
- La evaluación es el valor obtenido en la función de ajuste aplicada a cada individuo.

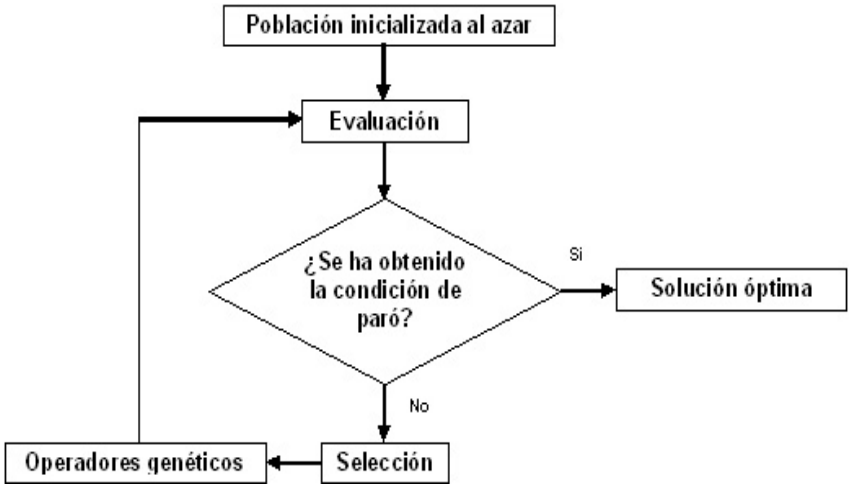

Figura 2.6: Esquema general de los algoritmos evolutivos.

- La selección elimina parte de la población, manteniendo preferentemente a los mejores individuos.
- La evolución aplica operadores genéticos (cruza, mutación,..) a los nuevos individuos de la población.

Algunos tipos de algoritmos evolutivos son:

- Algoritmos genéticos, son una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas.
- Estrategias evolutivas. Son reglas que definen la manera de actuar del individuo ante cierta situación.
- La programación genética, son una serie de instrucciones en un lenguaje de programación específico.

En este trabajo se emplearon los algoritmos genéticos. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las

que actúan en la evolución biológica, así como también a una selección de acuerdo con algún criterio dado, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Un ejemplo lo encontramos en [16] donde se emplean algoritmos genéticos para la calibración de cámaras, aquí se tiene en la representación de los individuos los parámetros de la calibración, los cuales van evolucionando hasta conseguirse los mejores parámetros.

Además existen trabajos que utilizan la computación evolutiva en el campo de visión, en los cuales se realiza la extracción de primitivas: contornos, curvas y círculos de una imagen. Todo esto a través de algoritmos genéticos como se presenta en [18]. La extracción de primitivas se realiza por medio de la codificación de las funciones de filtros que son la representación del individuo, eligiendo el mejor de ellos, correspondiendo al filtro más apto.

En los algoritmos genéticos se establece una codificación apropiada de las soluciones del espacio de búsqueda y una forma de evaluar la función objetivo para cada una de estas codificaciones. Las soluciones que se identifican son individuos que pueden formar parte de la población de búsqueda. La codificación de una solución se interpreta como el cromosoma del individuo compuesto de un cierto número de genes a los que les corresponden ciertos *alelos*.[1] Se consideran dos operaciones básicas: la mutación y el cruce. La mutación de un individuo consiste en modificar un gen, cambiando al azar, el alelo correspondiente. El cruce de dos individuos (llamados padres) producen un individuo llamado hijo tomando un número k (elegido al azar) de genes de uno de los padres y los t - k del otro. La población evoluciona de acuerdo a las estrategias de selección de individuos, tanto para las operaciones como para la supervivencia. La

[1] Cada una de las formas alternativas de un gen dado, concernientes al mismo carácter. En una célula diploide, los miembros de un par alélico ocupan posiciones correspondientes sobre un par de cromosomas homólogos. Si estos alelos son genéticamente idénticos, la célula o el organismo son homocigóticos; si son diferentes, se trata de heterocigóticos. Tres o más alelos de un gen dado constituyen una serie alelomórfica.

selección se puede hacer simulando una lucha entre los individuos de la población con un procedimiento que, dados dos individuos, selecciona uno de ellos teniendo en cuenta su valoración (la función objetivo) y la adaptación al ambiente y a la población (criterios de diversidad, representatividad). La lucha por la supervivencia tiene por objeto mantener controlado el tamaño de la población. La selección de los luchadores se puede hacer de diferentes maneras: dos individuos seleccionados al azar, cada nuevo individuo con otro seleccionado al azar o con el peor de los existentes, etc.

No solo se han usado los algoritmos genéticos en el campo de visión, también existen trabajos que usan programación genética (PG), como en [7], donde se describe el uso de PG para el análisis de imágenes conllevando a la segmentación. La PG muestra que se pueden obtener filtros óptimos que resuelven dicho problema, similarmente a lo que realiza la programación dinámica, para dar solución al problema de la correspondencia, y esto es dividiendo el problema en subproblemas.

Las estrategias evolutivas son métodos que se emplean generalmente en la optimización [12], sin embargo el trabajo presentado por J. Louchet [14] utiliza una estrategia evolutiva individual, para realizar la reconstrucción tridimensional de una escena mediante dos imágenes.

2.4. Algoritmo de las moscas

El algoritmo de las moscas es un algoritmo que evoluciona rápidamente y por esta razón, una de las aplicaciones para el cual fue diseñado, es para la detección de obstáculos en tiempo real usando pares de imágenes estéreo. El algoritmo va a producir un conjunto de puntos *3D* los cuales van a deducir las superficies.

Una mosca va a ser definida como un punto *3D* con coordenadas *(x,y,z)*. Las coordenadas de las proyecciones de las moscas son (x_R,y_R) en la imagen derecha y (x_L,y_L)

en la imagen izquierda, en el sistema de visión estéreo.

Si la mosca esta sobre la superficie de un objeto opaco, entonces el píxel correspondiente al par de imágenes tendrá el mismo nivel de gris. Por lo tanto, si la mosca no está sobre la superficie de un objeto, los niveles de gris de sus proyecciones y sus vecinos inmediatos no serán probablemente idénticos. Esto se puede expresar como una función de ajuste usada para controlar la evolución de la población de las moscas para una posición inicial aleatoria que converge sobre las superficies de los objetos visibles.

La función de ajuste evalúa el grado de similaridad de los vecindarios del píxel de las proyecciones de la mosca en cada imagen. Esto asegura un valor de ajuste alto para las moscas que se encuentran sobre la superficie de un objeto.

De manera genérica, la función de ajuste es:

$$ajuste(indv) = \frac{G}{(\sum_{colores} \sum_{(i,j)\in N} (L(x_L + i, y_L + j) - R(x_R + i, y_R + j))^2)} \tag{2.3}$$

La población de moscas es inicializada aleatoriamente en la intersección de la vista de dos cámaras. Las moscas evolucionan siguiendo los pasos de los algoritmos evolutivos.

Las moscas se concentran sobre obstáculos y sobre regiones donde el valor de la gradiente de gris es alto. El numerador G de la función de ajuste evita que las moscas se fijen o sean atrapadas dentro de regiones uniformes (como el cielo, superficie de carreteras, etc.).

2.4.1. Evaluación

La función de ajuste es usada para evaluar una comparación de la mosca en sus proyecciones de las imágenes izquierda y derecha dadas por la cámara. Si la mosca está sobre la superficie del objeto, las proyecciones tendrán similar vecindario sobre ambas imágenes y por lo tanto esta mosca será atribuida a un alto ajuste.

La figura 2.7 y 2.8 ilustran este principio. La figura 2.8 muestra los vecindarios de dos moscas en las imágenes izquierda y derecha. En este ejemplo, la mosca 1, que se encuentra sobre la superficie de un objeto, tendrá un mejor ajuste que la mosca 2.

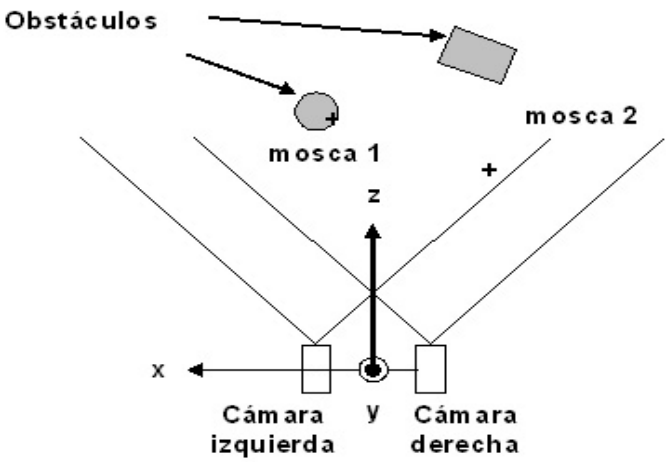

Figura 2.7: Ejemplo donde se utiliza el algoritmo de las moscas, se muestran dos moscas de la población.

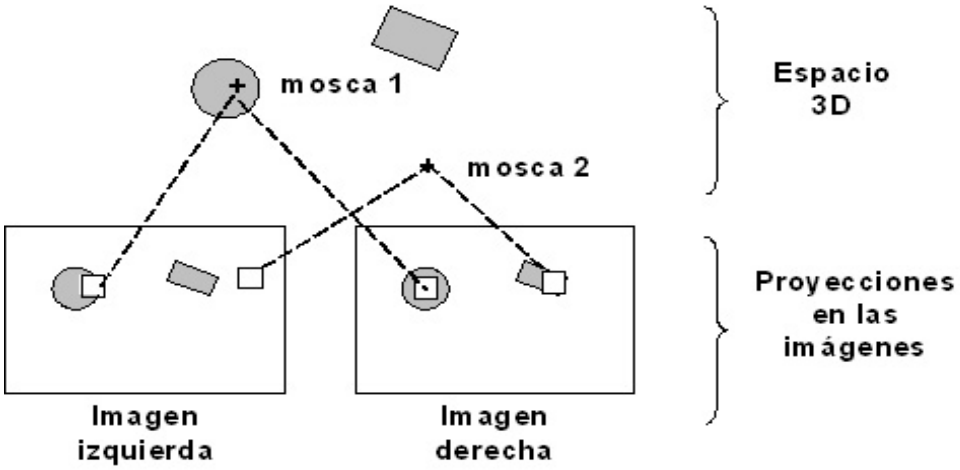

Figura 2.8: Proyecciones de dos moscas en las imágenes izquierda y derecha.

Una posible expresión matemática de la función de ajuste es:

$$F = \frac{\mid \nabla(M_L) \mid \cdot \mid \nabla(M_R) \mid}{(\sum_{colores} \sum_{(i,j)\in N}(L(x_L + i, y_L + j) - R(x_R + i, y_R + j))^2)} \tag{2.4}$$

$$G = \mid \nabla(M_L) \mid \cdot \mid \nabla(M_R) \mid \tag{2.5}$$

Donde:

- *(x_L,y_L)* y *(x_R,y_R)* son las coordenadas de las proyecciones izquierda y derecha del individuo actual.
- *L(x_L+i,y_L+j)* es el valor gris de la imagen izquierda en el píxel *(x_L+i,y_L+j)* y *R(x_R+i,y_R+j)* es el valor gris de la imagen derecha en el píxel *(x_R+i,y_R+j).*
- N es una vecindad introducida para obtener una alta discriminación de comparación de las proyecciones de las moscas.
- $|\nabla(M_L)|$ y $|\nabla(M_R)|$ son normas del gradiente de Sobel sobre las proyecciones izquierda y derecha de la mosca. Este valor sirve para penalizar a las moscas que se proyecten en regiones uniformes, es decir, a las moscas menos significativas.
- En las imágenes a color, la diferencia de cuadrados son calculados sobre cada canal de color.

2.4.2. Detección de bordes

Un borde en una imagen, es un cambio significante local en la intensidad de la imagen, usualmente asociada con una discontinuidad en cualquier intensidad de la imagen. Ya que al delimitar los objetos se definen los límites entre ellos y el fondo o entre los

objetos mismos. Las técnicas usadas en la detección de bordes tienen por objeto la localización de los puntos en los que se produce una variación de intensidad, empleándose para ello operadores que pueden ser aplicados a una parte de la imagen o a toda la imagen. La solución básica de muchos algoritmos de detección de bordes es el cómputo de operadores de derivadas locales (primera o segunda). Los operadores basados en la primera derivada que utilizamos son: Sobel, Prewitt y Canny.

En la figura 2.9 se muestra el vecindario usado para explicar los operadores Sobel y Prewitt.

a0	a1	a2
a7	[i,j]	a3
a6	a5	a4

Figura 2.9: Vecindario de píxeles usado para explicar los operadores Sobel y Prewitt.

Algunos detectores de bordes son:

Operador de Sobel: El operador de Sobel es la magnitud del gradiente calculado por:

$$M = \sqrt{s_x^2 + s_y^2} \tag{2.6}$$

donde las derivadas parciales son calculadas por:

$$s_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \tag{2.7}$$

$$s_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \tag{2.8}$$

con la constante $c = 2$.

s_x y s_y pueden ser implementados usando las máscaras de convolución que se muestra en la figura 2.10.

Sx =

-1	0	1
-2	0	2
-1	0	1

Sy =

1	2	1
0	0	0
-1	-2	-1

Figura 2.10: Vecindario s_x y s_y para Sobel.

Este operador localiza un énfasis sobre los píxeles que son ajustados para el centro de la máscara.

Operador de Prewitt: El operador Prewitt usa las mismas ecuaciones del operador de Sobel, excepto que la constante $c = 1$. En la figura 2.11 se muestra sus máscaras de convolución.

Sx =

-1	0	1
-1	0	1
-1	0	1

Sy =

1	1	1
0	0	0
-1	-1	-1

Figura 2.11: Vecindario s_x y s_y para Prewitt.

A diferencia del operador de Sobel, este operador no emplea algún énfasis en los píxeles que son ajustados para el centro de las máscaras.

Detector de bordes Canny: El detector de bordes Canny extrae bordes y cierra los contornos. Y se obtiene realizando lo siguiente:

- Se convoluciona f con filtros gaussianos unidimensionales. De esta forma la imagen se suaviza (eliminación de ruidos).

- A continuación se calcula el gradiente de la imagen suavizada usando una aproximación del gradiente de la función gaussiana (para determinar los píxeles donde se produce máxima variación).

- La matriz M correspondiente al módulo del gradiente de la función gaussiana tendrá valores grandes donde la variación de la intensidad sea grande. Se realiza, por tanto, un umbral, eliminando aquellos píxeles que no tienen una magnitud (módulo del gradiente) alta.

- Posteriormente se realiza un proceso de eliminación de falsos bordes y realzando bordes poco definidos. Este proceso se realiza eliminando aquellos píxeles que no son máximos locales.

En resumen, la detección de bordes usando operadores de aproximación del gradiente tiende a funcionar bien en los casos en que se involucran imágenes con transiciones de intensidad claramente definidas y ruidos relativamente bajos. Los pasos por cero (segunda derivada) ofrecen una alternativa en los casos en que los bordes están difusos o cuando está presente un alto contenido de ruido. El paso por cero ofrece fiabilidad en las localizaciones de bordes y la propiedad de suavizado de la convolución gaussiana reducen los efectos del ruido. El precio a pagar por estas ventajas es el incremento de complejidad de cálculo y tiempo.

El detector de bordes con el cual se obtiene mejores resultados es el operador de Sobel, así los valores con alta aptitud son obtenidos por moscas cuyas proyecciones tienen similitud y significancia (no-uniformidad) alrededor del píxel.

2.4.3. Selección

La selección es elitista y determinista, esta acomoda a las moscas acuerdo a su valor de ajuste y retiene los mejores individuos.

Una parte del operador [1][8] reduce el ajuste de una parte de la población de moscas y las fuerza a explorar otras áreas del espacio a examinar.

2.4.4. Operadores genéticos

Los siguientes operadores genéticos son aplicados a los individuos seleccionados.

- **Cruce.** Construye un descendiente aleatorio localizado en la línea segmentada entre sus padres; el descendiente de dos moscas $F_1 = (x_1, y_1, z_1)$ y $F_2 = (x_2, y_2, z_2)$ es la mosca $F_3 = (x_3, y_3, z_3)$ definida por:

$$F_3 \;=\; \lambda F_1 + (1 - \lambda) F_2 \tag{2.9}$$

  Con λ elegida aleatoriamente en el intervalo de [0,1].

- **Mutación Gaussiana.** Agrega un ruido Gaussiano a cada una de las tres coordenadas de la mosca mutada.

- **Inmigración.** Usado para mejorar la exploración del espacio de búsqueda, creando nuevos individuos aleatoriamente. Esto asegura una exploración constante en el espacio de búsqueda.

2.5. Operaciones morfológicas

En visión artificial es frecuente utilizar la morfología para el tratamiento de regiones en el sentido de determinar cómo se pueden cambiar, contar o evaluar.

Algunas de las aplicaciones en donde puede utilizarse la morfología son:

- Suavizado de los bordes de una región.
- Separación de determinadas regiones que el proceso de segmentación presenta unidas.
- Unión de regiones que han sido separadas durante la segmentación.
- Como consecuencia de los dos puntos anteriores, se facilita el cómputo de regiones en una imagen.

Algunas de las operaciones morfológicas que empleamos son: dilatación, erosión, apertura y cierre.

2.5.1. Dilatación

La transformación morfológica de la dilatación $\oplus$ combina dos conjuntos utilizando la adición de vectores (o adición de Minkowski). La dilatación $X \oplus B$ es el conjunto de puntos de todas las posibles adiciones vectoriales de pares de elementos, uno de cada conjunto X y B.

$$X \oplus B = \{d \in E^2 : d = x + b \ para \ cada \ x \in X \ y \ b \in B\} \tag{2.10}$$

La figura 2.12 presenta un ejemplo de dilatación,

$X = \{(0,1),(1,2),(2,0),(2,1),(3,0),(3,1)\}$

$B = \{(0,0),(0,1)\}$

$X \oplus B = \{(0,1),(1,2),(2,0),(2,1),(3,0),(3,1),(0,2),(1,3),(2,2),(3,2)\}$

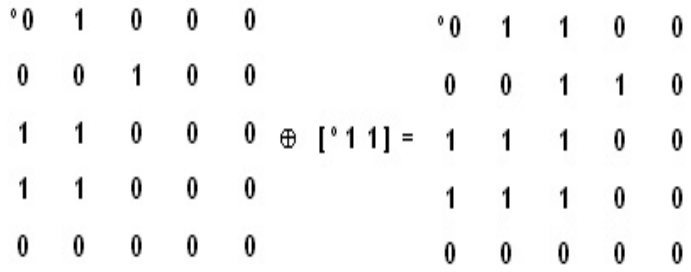

Figura 2.12: Dilatación.

2.5.2. Erosión

La transformación morfológica de la erosión $\otimes$ combina dos conjuntos utilizando la substracción de vectores. Es dual de la dilatación. Ni la erosión ni la dilatación son transformaciones invertibles.

$$X \otimes B = \{d \in E^2 : d = x - b \ para \ cada \ x \in X \ y \ b \in B\} \tag{2.11}$$

Esta expresión muestra que cada punto d del conjunto X, que para nosotros será la imagen, es probado; el resultado de la erosión está dado por los puntos d para los cuales todos los posibles $d + b$ están en X. La figura 2.13 muestra un ejemplo del conjunto de puntos X erosionados por el elemento estructural B.

$X = \{(0,2),(1,2),(2,0),(2,1),(2,2),(2,3),(3,2),(4,2)\}$

$B = \{(0,0),(0,1)\}$

$X \otimes B = \{(2,0),(2,1),(2,2)\}$

2.5.3. Apertura y cierre

Las operaciones morfológicas que se utilizan en este sistema son **apertura** y **cierre**.

La operación de **apertura** es creada con las operaciones de erosión seguida de una dilatación, la cual generalmente suaviza el contorno de una imagen, rompe istmos

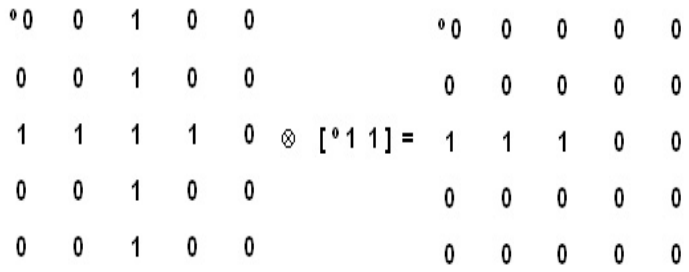

Figura 2.13: Erosión.

estrechos y elimina protuberancias delgadas.

La operación de **cierre** es obtenida con la dilatación seguida de una erosión, la cual también tiende a suavizar secciones de contornos pero, en oposición a la apertura, generalmente fusiona separaciones estrechas y entrantes delgados y profundos, elimina pequeños huecos y rellena agujeros del contorno.

Capítulo 3

Análisis del modelo

Para el análisis de éste modelo se utiliza una cámara, la cual desplazamos a distintas posiciones conocidas, para capturar las imágenes en cada una de las posiciones y obtener la escena tridimensional parcialmente reconstruida a partir de éstas imágenes. Las imágenes van a estar separadas una distancia considerable donde esa distancia va a ser conocida.

3.1. Análisis general

En la figura 3.1 se muestra el algoritmo que contiene los pasos para la obtención del número de regiones en la cual será dividida la imagen izquierda y así ser analizada por regiones. Se calcula y se genera el número de moscas que corresponden por región.

La generación de moscas es aleatoria y comenzamos con la generación de una población de moscas en la intersección del espacio de las dos imágenes. A mayor número de moscas se va a tener una mejor visibilidad de la escena pero tendremos también un mayor desgaste en cómputo. En base a las pruebas realizadas es suficiente considerar 3000 moscas como un número adecuado para el manejo de la reconstrucción del objeto.

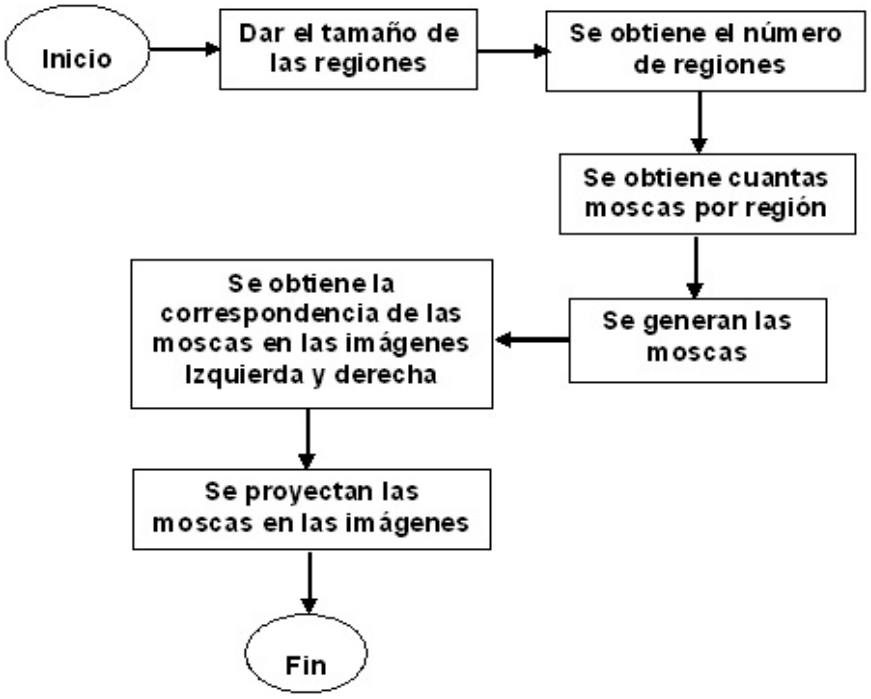

Figura 3.1: Algoritmo que muestra la obtención de las regiones.

Los pasos a seguir para la captura de las imágenes se presenta en la figura 3.2, donde se captura la primera imagen en un tiempo t_1 la cual se almacena para posteriormente procesarla, después se desplaza la cámara una corta distancia sobre el plano horizontal y se captura la segunda imagen en un tiempo t_2 para posteriormente procesarla. Ya teniendo las dos imágenes se aplica el algoritmo para iniciar con la reconstrucción de la escena en tres dimensiones. Para continuar con la reconstrucción de la escena, la segunda imagen toma el lugar de la primera, se realiza el desplazamiento horizontal y se captura la segunda imagen.

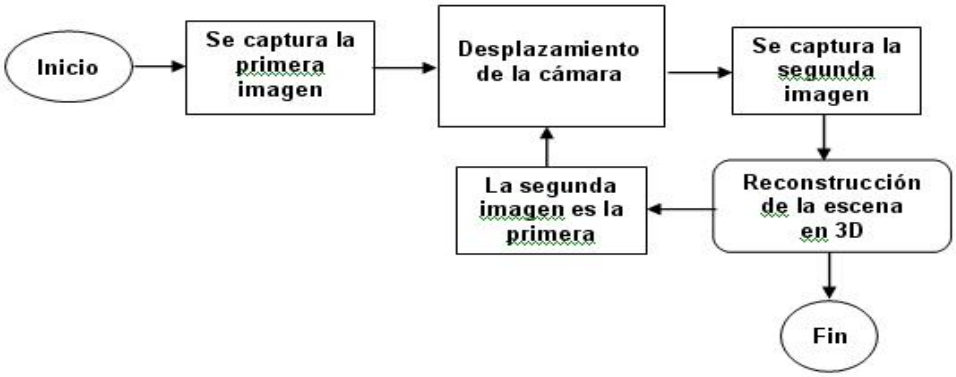

Figura 3.2: Análisis general del modelo.

3.2. Reconstrucción de la escena

La figura 3.3 muestra el algoritmo de las tareas que se llevan a cabo para la reconstrucción de la escena en tres dimensiones.

Primero se obtiene la desviación estándar de los colores de los píxeles en cada región, cuando la desviación estándar sea pequeña no se tomará en cuenta esa región, ya que implica que hay un mismo color presente en casi toda la región, mientras que en una desviación estándar grande implica que hay muchos colores en ella y por tanto contendrá objetos. La desviación estándar se va a aplicar cuando exista un desplazamiento de la cámara, esto quiere decir cuando se encuentre una nueva imagen.

Después se calcula la función de ajuste a cada una de las moscas las cuales se localizarán y serán agrupadas por regiones. Posteriormente se ordenará de acuerdo a los resultados obtenidos de la función de ajuste, de las mejores a las peores para que finalmente a partir de estos datos se conserve un porcentaje de las cuales serán las moscas más aptas y se desechen la moscas menos aptas.

Posteriormente se generan moscas por cruce, mutación y nuevas, para así calcular su correspondencia y ser proyectadas en las imágenes.

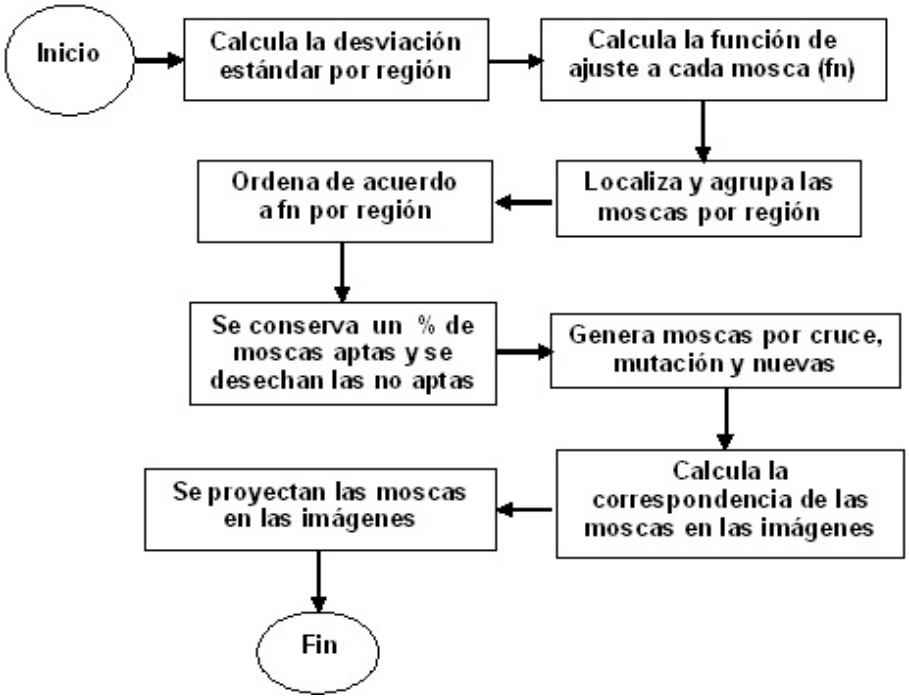

Figura 3.3: Algoritmo que muestra la función de reconstrucción.

En la figura 3.4 se muestra el algoritmo del cálculo de la función de ajuste.

La figura 3.5 muestra el algoritmo para obtener la correspondencia de cada mosca. Utilizando las ecuaciones 2.1 y 2.2 se obtiene la proyección en la imagen izquierda y para la imagen derecha se realiza una rotación y traslación a la coordenada tridimensional de cada mosca, posteriormente se aplican las ecuaciones 2.1 y 2.2.

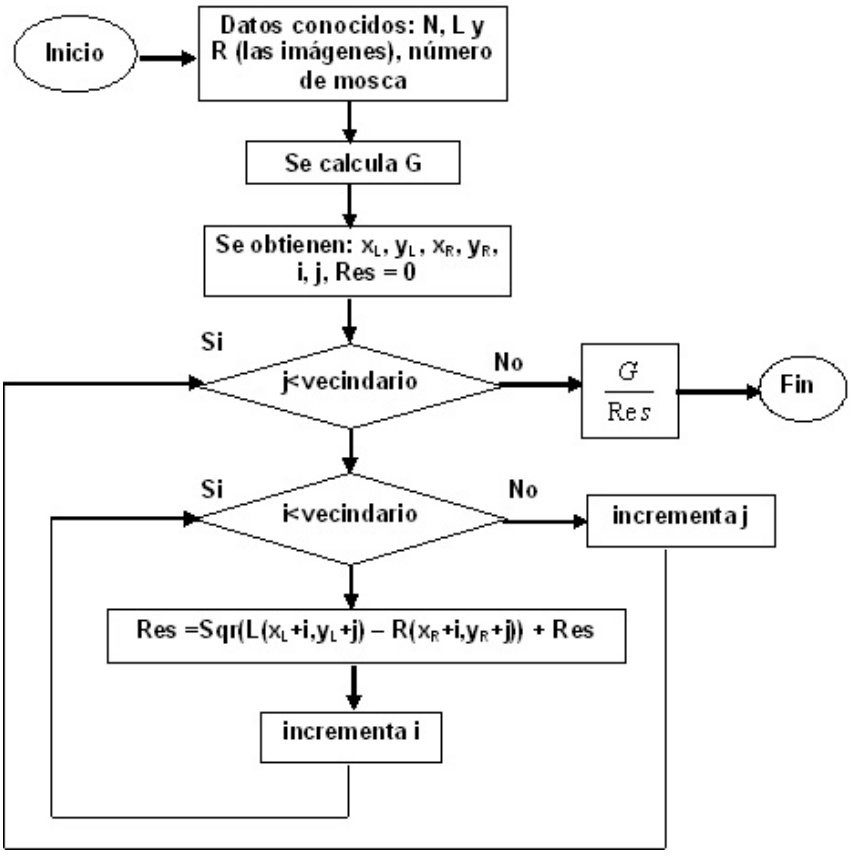

Figura 3.4: Algoritmo que muestra el cálculo de la función de ajuste.

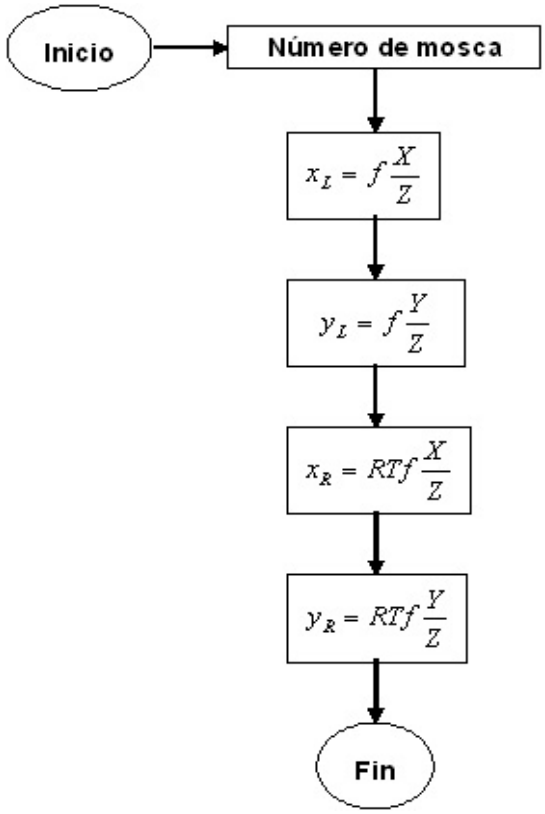

Figura 3.5: Algoritmo que muestra la obtención de la correspondencia.

3.3. Generación de moscas

Ya obtenido el número de regiones y obtenido el número de moscas que corresponden a cada región, estas son generadas y proyectadas. En la figura 3.6 se muestra la generación de las coordenadas de las moscas aleatoriamente de un intervalo. El intervalo de X y Y va a depender de Z.

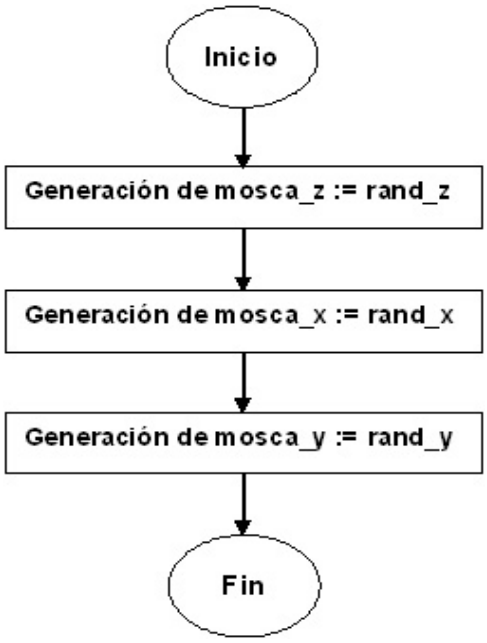

Figura 3.6: Algoritmo que muestra la generación de moscas.

3.4. Operadores genéticos cruce y mutación

Posteriormente de que las moscas han sido generadas y proyectadas se aplica la función de ajuste a cada una de ellas y así eliminar las moscas menos aptas, para proseguir con la regeneración de nuevas moscas a partir de las moscas aptas, aplicando los operadores de cruce, mutación e inmigración (moscas nuevas).

En la figura 3.7 se presenta el algoritmo de la regeneración de nuevas moscas por el operador de cruce el cual consiste, que dados dos padres F_1 y F_2 se genera el hijo F_3 como en la ecuación 2.9.

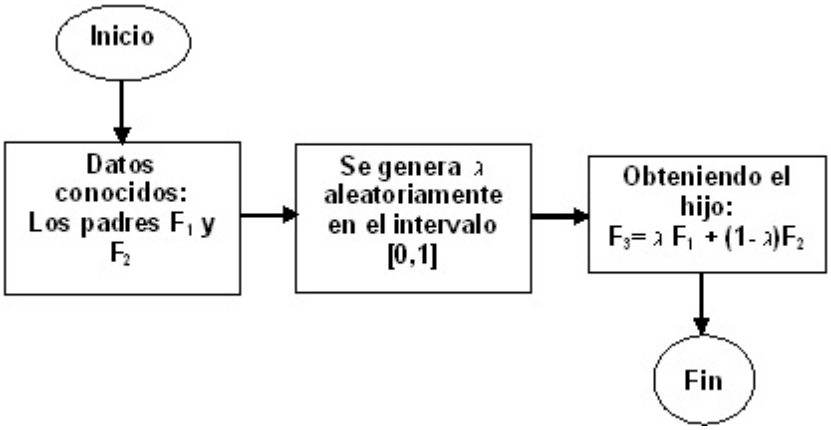

Figura 3.7: Algoritmo para la regeneración de moscas por cruce.

En la figura 3.8 se muestra el algoritmo de la regeneración de las nuevas moscas por el operador de mutación, la mutación gaussiana anexa un ruido gaussiano a cada una de las tres coordenadas de la mosca a mutar.

Figura 3.8: Algoritmo para la regeneración de moscas por mutación.

Capítulo 4

Diseño del modelo

4.1. Arquitectura del modelo

Los módulos o funciones con los que cuenta el modelo son: Abrir, Guardar, Salir, Calibración, Tamaño de las regiones, Opciones, Vistas y Reconstruir. El módulo Reconstruir contiene a la Función de ajuste, Correspondencia, Cruce, Mutación y Nueva.

A continuación en la figura 4.1 se muestra la arquitectura del modelo, todos los módulos se encuentran en el menú principal.

Los módulos mencionados anteriormente se describen a continuación:

- **Módulo Abrir.** Este módulo permite la apertura de archivos de imágenes.
- **Módulo Guardar.** Este módulo permite salvar la imagen que se tiene en pantalla.
- **Módulo Salir.** Con este módulo se sale del sistema.
- **Módulo Calibración.** Este módulo permite determinar y visualizar la geometría externa e interna de la cámara.
- **Módulo Tamaño de las regiones.** Divide a la imagen en regiones para la distribución de moscas.

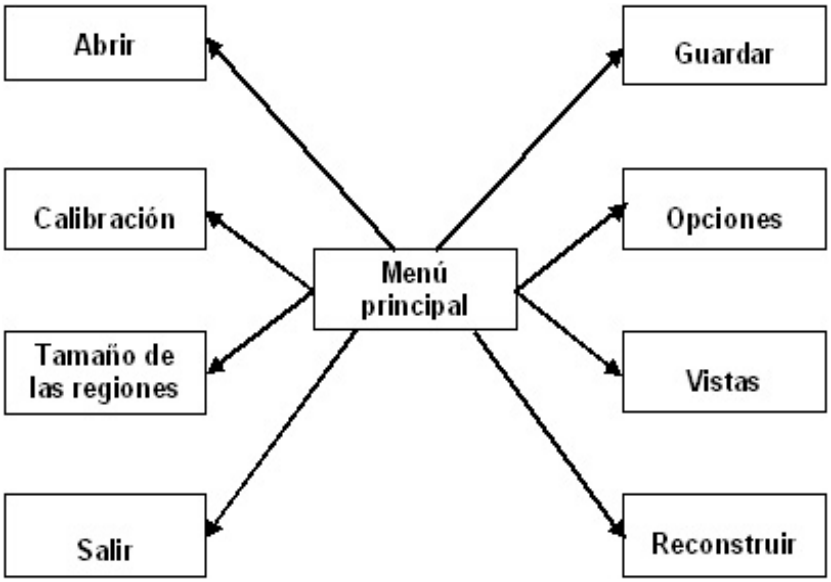

Figura 4.1: Arquitectura del modelo.

- **Módulo Opciones.** Este módulo permite dar el número de moscas con el cual se desea trabajar y el color de ellas.

- **Módulo Vistas.** Este módulo muestra la posición de las moscas en los planos *XY*, *XZ*, *YZ* y *XYZ*.

- **Módulo Reconstruir.** Este módulo realiza la reconstrucción *3D*, utilizando dos imágenes *2D* y los siguientes módulos:

- **Módulo Función de Ajuste.** En la función de ajuste se aplica la ecuación 2.4, se busca que las diferencias de color entre la mosca y su vecindad sea mínima, de preferencia cero, lo que da un valor en la función de ajuste muy grande.

- **Módulo Correspondencia.** Con este módulo se obtiene la correspondencia de las moscas en las imágenes izquierda y derecha.

- **Módulo Cruce.** Con este módulo se obtiene una mosca (hijo) a partir de dos

padres.

- **Módulo Mutación.** Con este módulo se obtiene una mosca (hijo), con solo agregarle ruido gaussiano a una mosca (padre).

- **Módulo Nueva.** Con este módulo se obtienen nuevas moscas.

En la figura 4.2 se muestra la arquitectura del módulo Reconstruir, donde se señala que para realizar una reconstrucción siempre se debe aplicar la función de Ajuste, para que podamos elegir que moscas son las moscas aptas y desechar las no aptas, después se aplica un porcentaje de mutación, un porcentaje de cruce y un porcentaje de nueva para que finalmente sean proyectadas en las imágenes izquierda y derecha.

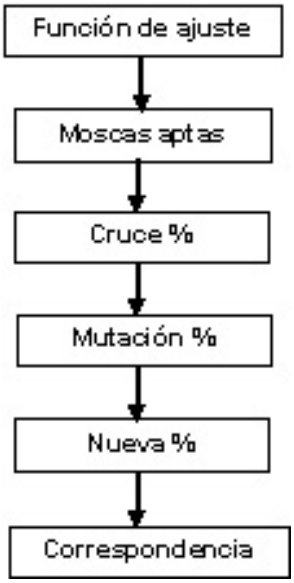

Figura 4.2: Arquitectura del módulo Reconstruir.

4.2. Descripción de la arquitectura del modelo

A continuación se describe cada una de las funciones con las que cuenta el modelo, se mencionan los datos de entrada y los resultados al aplicar cada uno de los algoritmos diseñados para cada una de las funciones. También se menciona los prototipos de estas funciones.

- **Abrir.**

  Función. Abre un archivo imagen.

  Datos de entrada. Nombre del archivo el cual se desea abrir.

  Datos de salida. Los datos del archivo cargados en memoria.

  Prototipo. procedure TfrmMoscaster.BAbrirIma1Click(Sender: TObject);

- **Salir.**

  Función. Sale del programa.

  Datos de entrada. Se llama a la función de Delphi close la cual va a cerrar la pantalla principal.

  Datos de salida. Liberación de memoria.

  Prototipo. procedure TfrmMoscaster.BSalirClick(Sender: TObject);

- **Guardar.**

  Función. Guarda imagen.

  Datos de entrada. Datos de la imagen que se tenga en pantalla.

  Datos de salida. Archivo de la imagen almacenada.

  Prototipo. procedure TfrmMoscaster.GuardarIma1Click(Sender: TObject);

- **Calibración.**

  Función. Mostrar los parámetros intrínsecos y extrínsecos de la cámara.

  Datos de entrada.

Datos de salida. Datos de los parámetros intrínsecos y extrínsecos de la cámara calibrada.

Prototipo. procedure TfrmMoscaster.CalibrarClick(Sender: TObject);

- **Tamaño de las regiones.**

  Función. Divide la imagen en regiones.

  Datos de entrada. Tamaño de x y y.

  Datos de salida. Imagen dividida en regiones.

  Prototipo. procedure TfrmMoscaster.NoCuadros1Click(Sender: TObject);

- **Opciones.**

  Función. Agrega el número de moscas con el cual se desea trabajar y el color.

  Datos de entrada. Número de moscas y color.

  Datos de salida. Número de moscas y moscas al color elegido.

  Prototipo. procedure TfrmMoscaster.Opciones1Click(Sender:TObject);

- **Vistas.**

  Función. Muestra los planos XY, XZ, YZ y XYZ.

  Datos de entrada. Coordenadas de las moscas.

  Datos de salida. Vistas de los planos XY, XZ, YZ y XYZ.

  Prototipo. procedure TfrmMoscaster.VistasClick(Sender:TObject);

- **Reconstruir.**

  Función. Reconstruye la escena *3D*.

  Datos de entrada. Dos imágenes *2D*.

  Datos de salida. Reconstrucción de la escena *3D*.

  Prototipo. procedure TfrmMoscaster.Opciones1Click(Sender:TObject);

Las funciones las cuales utiliza la función Reconstruir son:

- **Función de ajuste.**

  Función. Selecciona las moscas aptas.

  Datos de entrada. Arreglo donde se encuentra la numeración de cada mosca y sus coordenadas bidimensionales para ser localizadas en la imagen.

  Datos de salida. Arreglo de los resultados de aplicar la función de ajuste a cada mosca.

  Prototipo. procedure TfrmMoscaster.CAjuste();

- **Cruce.**

  Función. Obtiene un hijo a partir de dos padres.

  Datos de entrada. Coordenadas de la mosca 1 *(x_1, y_1, z_1)* y mosca 2 *(x_2, y_2, z_2)* en el espacio.

  Datos de salida. Coordenada del nuevo hijo en el espacio de intersección de las dos imágenes.

  Prototipo. procedure TfrmMoscaster.OCruce();

- **Mutación.**

  Función. Obtiene un hijo, agregando ruido gaussiano a cada coordenada *(x,y,z)* de una mosca.

  Datos de entrada. Coordenada *(x,y,z)* de la mosca a mutar.

  Datos de salida. Coordenada del nuevo hijo en el espacio de intersección de las dos imágenes.

  Prototipo. procedure TfrmMoscaster.OMutacion();

- **Nueva.**

  Función. Obtiene moscas nuevas.

Datos de entrada. Límites para las coordenadas X, Y y Z.

Datos de salida. Coordenada de la nueva mosca.

Prototipo. procedure TfrmMoscaster.ONueva();

Capítulo 5

Implementación del modelo

5.1. Captura de la imagen

Una de las funciones que se utiliza en todo el programa es Muestra(), la cual va a mostrar la imagen en pantalla como se muestra en la figura 5.1. Para realizar ésta tarea se utiliza la función Image1.Picture.Assign(B) la cual va a mostrar la imagen.

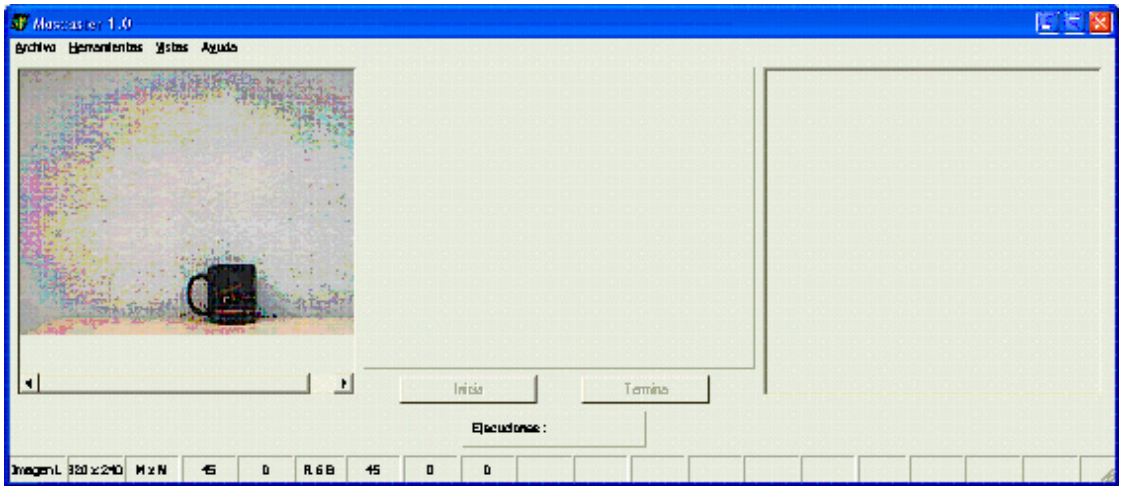

Figura 5.1: Visualizando una imagen.

5.2. Función de Ajuste

En el código siguiente, se muestra la función que realiza el cálculo de la función de Ajuste en cada mosca. En la función F_ajusteClick() se almacenan las imágenes izquierda y derecha para posteriormente ser utilizadas en el cálculo de Sobel. Además realiza el llamado a la función fAjuste(), donde se aplica la ecuación 2.4.

```
procedure TfrmMoscaster.F_ajusteClick(Sender: TObject); var
  G,SobelL, SobelR : real;
  m : integer;
begin
  BM2Mat(B1,MTr1);  //Se almacena la imagenL en el arreglo MTr1
  BM2Mat(B2,MTr2);  //Se almacena la imagenR en el arreglo MTr2
  for m := 1 to NUM_MOSCA do // num. de moscas
  begin
    FAjus[0][m] := m;
  end;
  for m := 1 to NUM_MOSCA do
  begin
    SobelL := fSobel(MTr1,Iancho1,Ialto1,m,1); //Calcula Sobel imagenL
    SobelR := fSobel(MTr2,Iancho1,Ialto1,m,0); //Calcula Sobel imagenR
    G := Abs(SobelL) * Abs(SobelR);
    fAjuste(MTr1,MTr2,G,m,Iancho1,Ialto1);
  end;
end;
```

La función fAjuste() calcula la función de ajuste de cada mosca y el resultado es almacenado en el arreglo FAjus para posteriormente ser utilizados.

```
procedure fAjuste(var ML:Mat3D; var MR:Mat3D; G:real;
i,nc,nr:integer);
begin
  rsum := 0; rsumc := 0;
  vec := 1; //se barre una vecindad de 1
  for c := 0 to 2 do  //por los tres colores RGB
  begin
    for j := -vec to vec do
      for k := -vec to vec do
      begin
        xt := moscaxL[i] + j;
        if(xt < 0) then xt := -xt;
        if(xt >= nc) then xt := 2 * nc - xt - 1;
        yt := moscayL[i] + k;
        if(yt < 0) then yt := -yt;
        if(yt >= nr) then yt := 2 * nr - yt - 1;
        rsum := sqr(ML[xt][yt][c] - MR[xt][yt][c]) + rsum;
      end;
      rsumc := rsum + rsumc;
      rsum := 0;
  end;
  FAjus[1][i] := G / rsumc;
end;
```

Una vez aplicado el valor de ajuste a todas las moscas, se ordenan por región y se desechan las moscas menos aptas. Este valor va a depender del porcentaje de moscas aptas que se requiera tener. Por ejemplo, en la figura 5.2 se muestra el cuadro de diálogo que muestra el número de moscas a utilizar y el color de ellas. En la figura 5.3 se muestran los porcentajes de moscas aptas, moscas regeneradas por cruce, moscas regeneradas por mutación y moscas nuevas.

Figura 5.2: Opciones para elegir el número de moscas y el color de ellas.

Cuando ya obtenemos las moscas aptas, regeneramos la nueva población con mutación, cruce e inmigración a partir de toda la población, no por regiones. Así en cierta medida podemos regenerar moscas entre regiones para evitar que se centralicen en alguna región en particular. En la figura 5.4 se muestra el resultado de aplicar las opciones mencionadas en la figura 5.3 en las dos imágenes. Éstas imágenes fueron capturadas y desplazadas cinco centímetros una de la otra. Además se puede apreciar que las moscas están abarcando toda la imagen, pero la selección de ellas es por regiones, tomando solo el 50 % de las regiones donde se tiene una desviación estándar alta, que es o son las regiones donde es probable que se encuentre el objeto u objetos.

Figura 5.3: Opciones para elegir los porcentajes de moscas aptas, moscas regeneradas por cruce, moscas regeneradas por mutación y moscas nuevas.

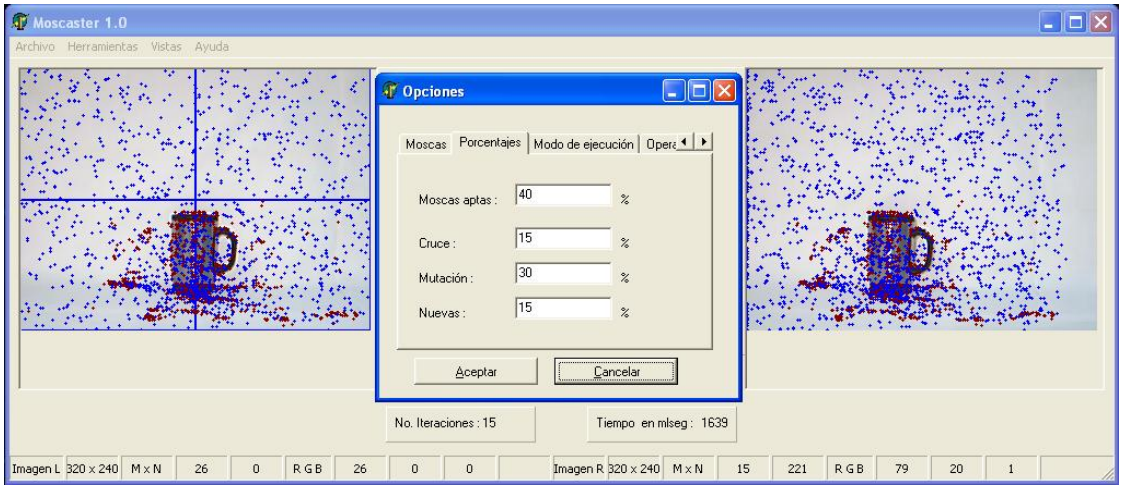

Figura 5.4: Resultado de haber aplicado las opciones de la figura 5.3.

5.3. Función de Cruce

Aquí se toman dos moscas: mosca 1 (x_1, y_1, z_1) y mosca 2 (x_2, y_2, z_2) para generar la mosca 3 (x_3, y_3, z_3), donde x_3 es un punto aleatorio entre x_1 y x_2, y_3 es un punto aleatorio entre y_1 y y_2 y z_3 es un punto aleatorio entre z_1 y z_2. En la figura 5.5 se muestra la ejecución de esta función, se observa que el utilizar solo la función de cruce, limita a las moscas y las acota. Esta acotación se irá reduciendo en cada iteración.

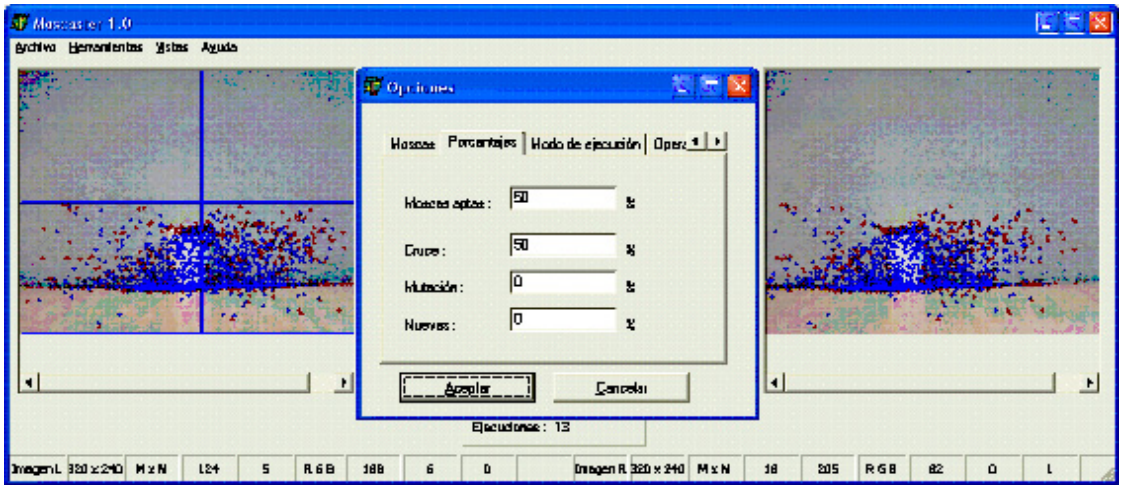

Figura 5.5: Ejecución de la función de cruce.

Posteriormente se muestra el código.

```
procedure Cruce(num,dc,dr : integer);
begin
  rs := (num * 50) div 100; //No. de moscas aptas
  for i := rs*dr*dc+1 to NUM_MOSCA do begin //Genera moscas por cruce
    p1 :=  Trunc(num_genera(1,rs*dr*dc));//elige padre 1 aleatoriamente
    p2 :=  Trunc(num_genera(1,rs*dr*dc));//elige padre 2 aleatoriamente
    valr := num_genera(0,1); //Random
    moscaz[i] := (valr * moscaz[p1]) + ((1 - valr)* moscaz[p2]);
```

```
    moscax[i] := (valr * moscax[p1]) + ((1 - valr)* moscax[p2]);
    moscay[i] := (valr * moscay[p1]) + ((1 - valr)* moscay[p2]);
  end;
end;
```

5.4. Función de Mutación

En esta función se toma cada coordenada de la mosca y se le agrega ruido gaussiano a x, y y z. Ya agregado el ruido a cada una de las tres coordenadas, se obtiene la nueva mosca, posteriormente se realizan las operaciones adecuadas para poder visualizar la mosca en ambas imágenes. En la figura 5.6 se muestra la ejecución de ésta función, se observa que el utilizar solo la función de mutación, las moscas se extienden sobre toda la imagen, esto es abarcando todas las regiones, aunque como opciones se maneja el 50 % de moscas aptas, esas moscas están sobre el 50 % de las regiones donde se tiene una desviación estándar alta. Y el resto de ellas es creado por mutación, las cuales se encuentran sobre toda la imagen.

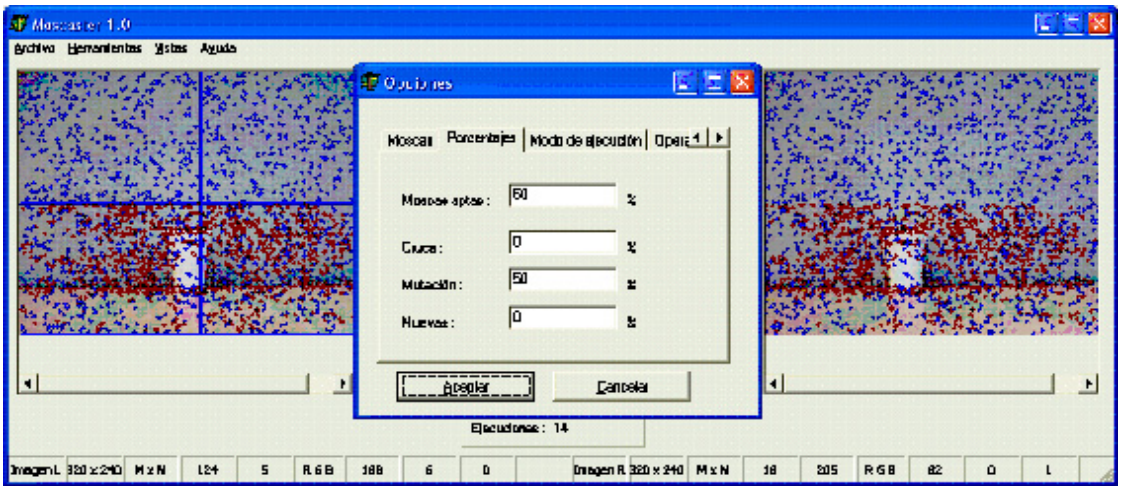

Figura 5.6: Ejecución de la función de mutación.

Posteriormente se muestra el código.

```
procedure Muta(num,dc,dr : integer);
begin
  rs := (num * 50) div 100; //No. de moscas aptas
  for i := rs*dr*dc+1 to NUM_MOSCA do //Genera moscas por mutacion
  begin
    valr := RandG(0,10000);
    moscaz[i] := moscaz[i] + valr;
    moscax[i] := moscax[i] + valr;
    moscay[i] := moscay[i] + valr;
  end;
end;
```

5.5. Función de Generación de moscas nuevas

La función de generación de moscas nuevas, va a generar moscas nuevas aleatoriamente al inicio de la ejecución del proceso, por ejemplo, en la figura 5.7 se muestran 3000 moscas sobre la imagen, donde en cada región se encuentran 750. Además ésta función va a generar moscas nuevas en un porcentaje, en cada una de las itereaciones del proceso, para ampliar el espacio de busqueda en toda la imagen y así evitar concetraciones de moscas en alguna región.

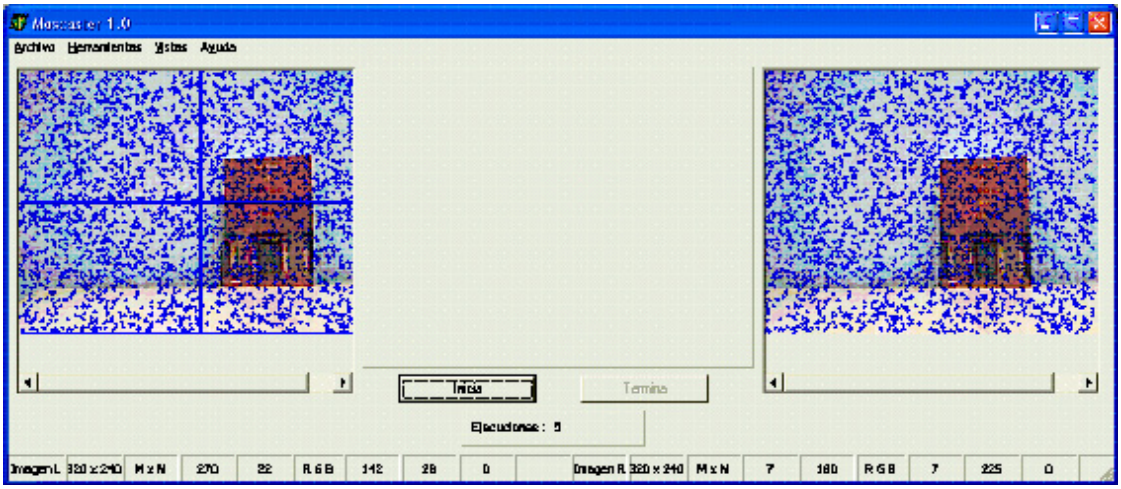

Figura 5.7: Generación de moscas nuevas.

5.6. Función de Correspondencia

La función de correspondencia toma a las moscas que se encuentran en el espacio para realizar la correspondencia de ellas en la imagen izquierda y la imagen derecha, para posteriormente ser proyectadas en las imágenes como se muestra en la figura 5.8.

5.7. Visualización

Para conocer la posición de las moscas en el escenario y realizar la reconstrucción que ellas forman, es necesario visualizarlas en diferentes vistas como lo son XY, XZ, YZ y XYZ.

Todas las moscas se fusionan para obtener la representación del entorno, sin embargo en la reconstrucción del escenario nos encontramos con algunas perturbaciones o algunos agujeros, los cuales se pueden solucionar con solo aplicar algunas operaciones morfológicas, como son apertura y cerradura. Estas operaciones pueden ser aplicadas a las vistas XY, XZ y YZ para mejorar la visualización de ellas.

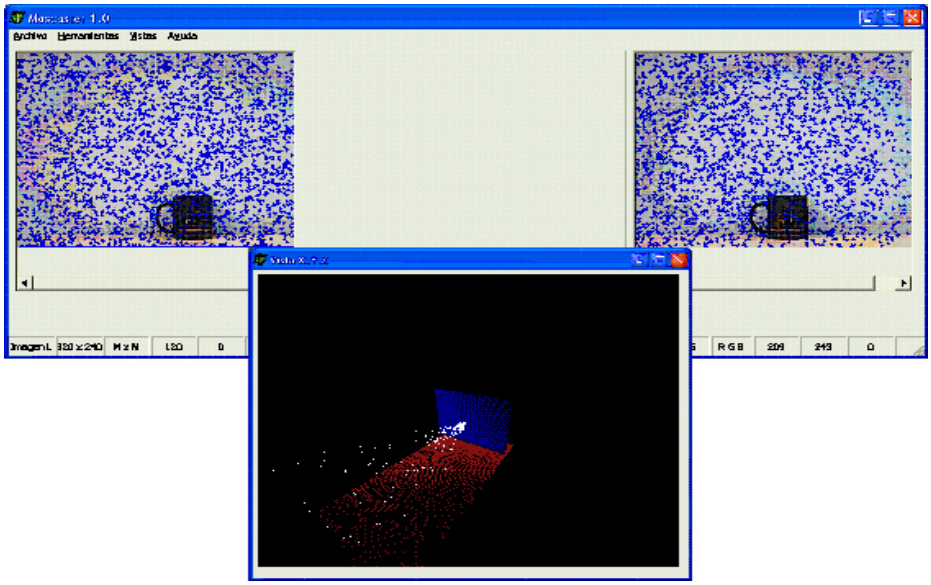

Figura 5.8: Proyección de las moscas en la imagen izquierda e imagen derecha.

Por ejemplo, en la figura 5.9 se muestra el resultado del procesamiento de dos imágenes, se observa que el 90 % de moscas aptas se encuentran posadas en el objeto.

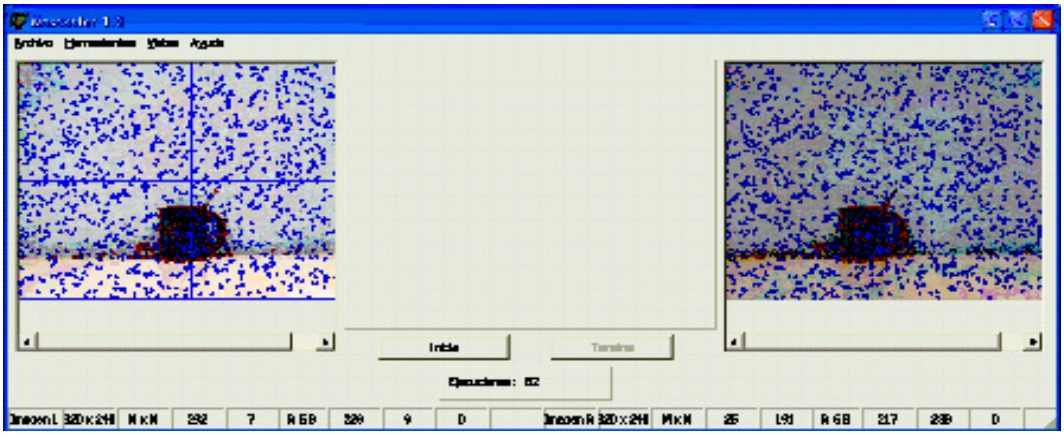

Figura 5.9: Moscas posadas en el objeto.

En la figura 5.10 (a) se muestra la vista XY y en (b) el resultado de haber aplicado una operación de apertura y una operación de cerradura. Además de estas operaciones, la vista XY tiene la opción de poder aplicar la operación de mediana a la imagen y trabajar con todas las moscas aptas o solo con un porcentaje de ellas.

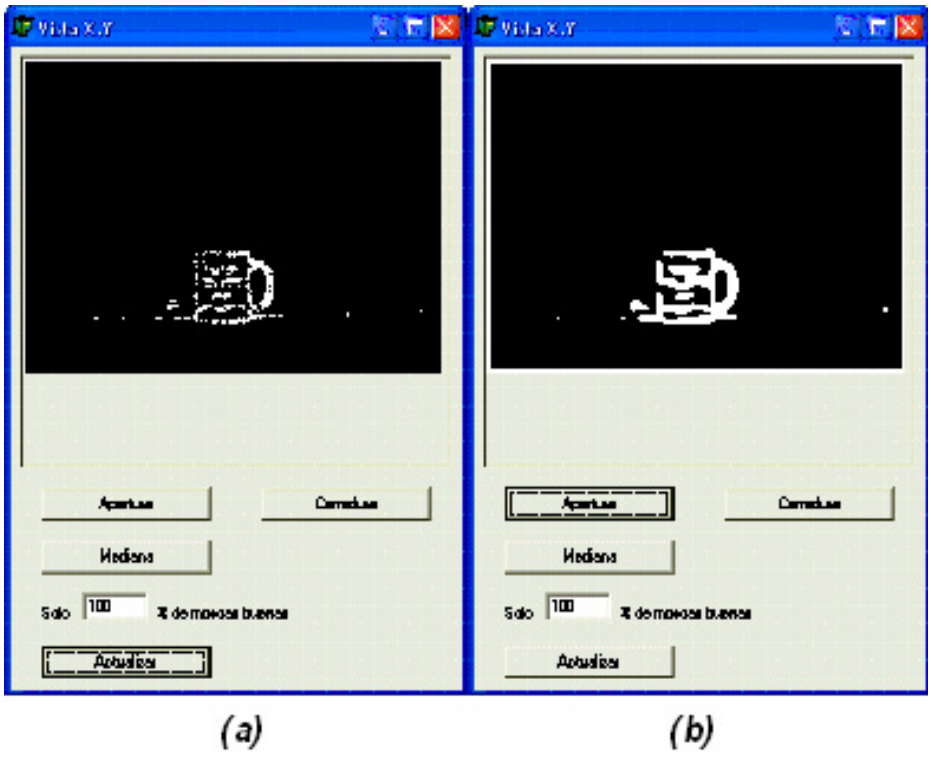

Figura 5.10: (a) Vista XY, (b) Después de aplicar una operación de apertura y una de cerradura.

En la figura 5.11 (a) se muestra la vista XZ y en (b) el resultado de haber aplicado una operación de apertura y una operación de cerradura.

En la figura 5.12 (a) se muestra la vista YZ y en (b) el resultado de haber aplicado una operación de apertura y una operación de cerradura.

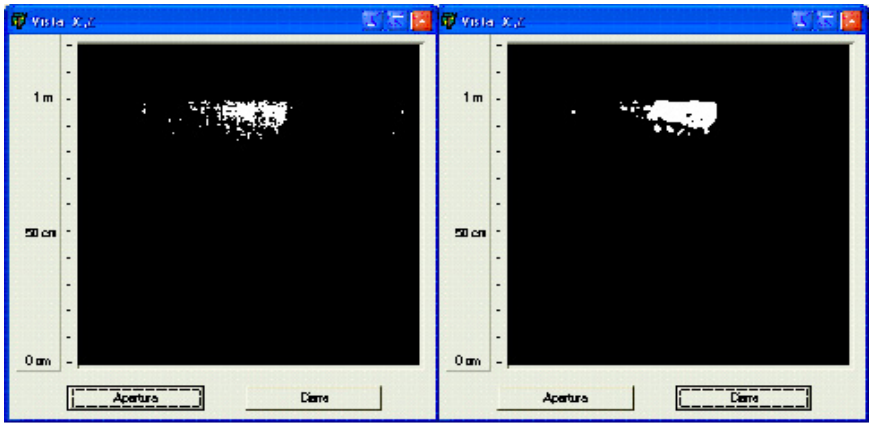

Figura 5.11: (a) Vista *XZ*, (b) Después de aplicar una operación de apertura y una de cerradura.

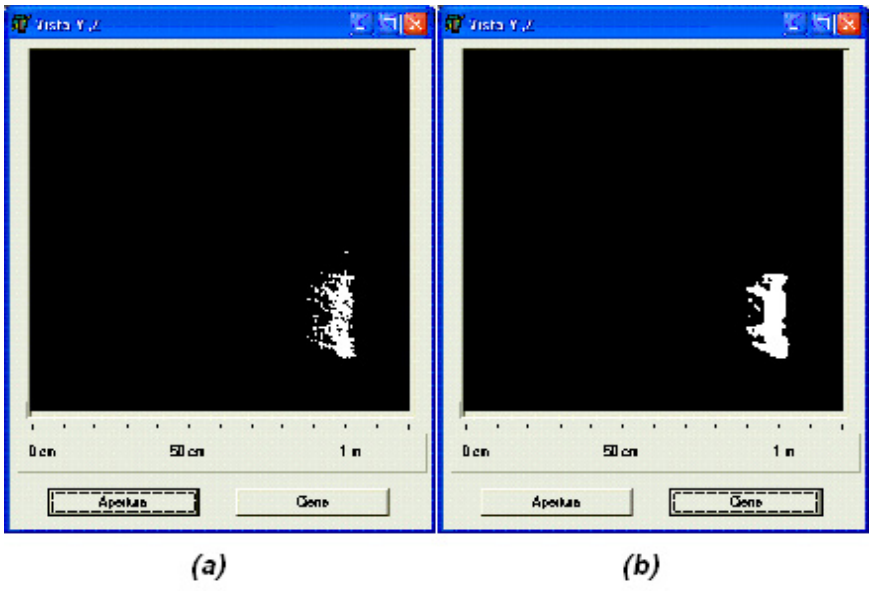

Figura 5.12: (a) Vista *YZ*, (b) Después de aplicar una operación de apertura y una de cerradura.

Por último, en la figura 5.13 se muestra la vista tridimensional de este ejemplo, se puede apreciar tridimensionalmente la reconstrucción que se logra con las moscas y además también se observan algunas moscas solitarias.

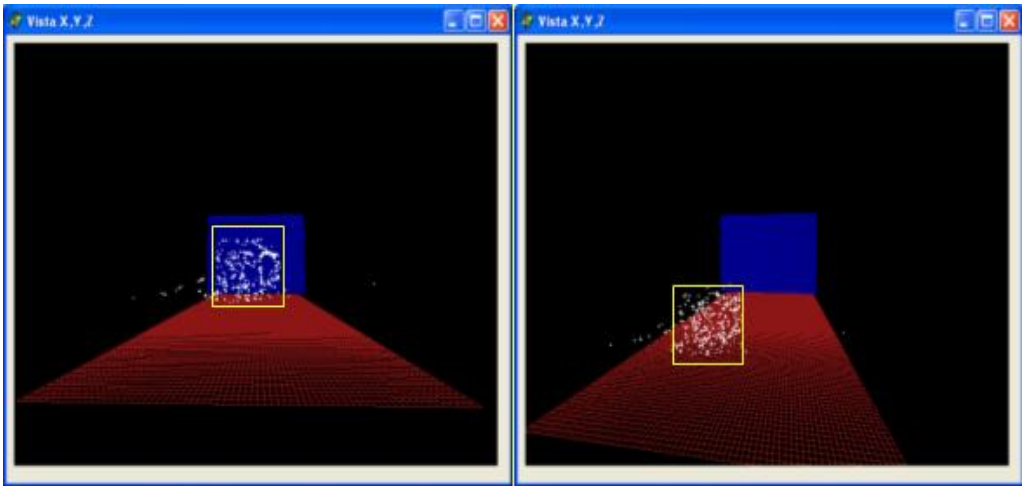

Figura 5.13: Vista *XYZ*.

5.8. Infraestructura

Las pruebas fueron realizadas en una computadora con las siguientes características:

Hardware

- Pentium 4 a 3 GHz.
- 512 MB de Ram.

Software

- Windows XP.
- Delphi 6.0.
- Video Lab 2.2.

- GLScene.

5.8.1. Videlo Lab

VideoLab es un conjunto de componentes de VCL (Biblioteca de Componentes Visuales) para procesar, capturar y devuelve archivos de video. Los componentes se basan en la tecnología libre de OpenWire.

VideoLab se utilizó para capturar la escena en el procesamiento durante la navegación del robot. Los componentes que se utilizaron fueron:

- VLDSImageDisplay. Componente para visualizar los datos del video.
- VLDSCapture. Captura video de la cámara u otro dispositivo de captura.
- VLGenericFilter. Filtra el video definido por el usuario.

5.8.2. GLScene

GLScene es una librería *3D* para Delphi basada en OpenGL que proporciona la posibilidad de renderizado de objetos y escenas *3D* con facilidad.

GLScene se utilizó para la construcción de la vista *XYZ*. Los componentes que se utilizaron fueron:

- GLScene. Es un control que contiene el escenario *3D* como son :
  - GLCamara. Objeto que observa la escena, contiene posición y dirección.
  - GLPoints. Crea serie de puntos o sólo un punto con este objeto. Los puntos son puntos en el espacio.
  - GLPlane. Este Objeto es un polígono compuesto de cuatro vértices y dos triángulos

- GLXYZGrid. Despliega una regilla a lo largo de los ejes X, Y y Z.
- GLLightSource. Objeto que ilumina la escena.
- GLDummyCube. Organiza objetos en grupos.

- GLSceneViewer. Permite visualizar lo que está viendo una cámara dentro del escenario *3D*.

Capítulo 6

Pruebas y Resultados

El modelo se utilizó en las siguientes situaciones:

- Procesando un par de imágenes.
- Procesando una secuencia. En este caso se empleó un conjunto de cinco imágenes almacenadas, que el modelo fue adquiriendo.
- Empleando el sistema en la tarea de navegación del robot. En esta prueba la cámara fue montada en un robot de Lego, el cual se fue desplazando (aproximadamente en pasos de cinco centímetros) y en cada paso capturó las imágenes.

En cada situación se efectuaron dos pruebas relacionadas con la segmentación de las imágenes:

- Dividida en dos regiones, donde cada región fue de 160x240 píxeles de dimensión.
- Dividida en cuatro regiones, donde cada región fue de 160x120 píxeles de dimensión.

Además se emplearon diferentes tamaños de población de moscas, las cuales fueron:

- 500 moscas.

- 1500 moscas.
- 3000 moscas.
- 5000 moscas.

Al inicio de la ejecución del modelo habrá el mismo número de moscas sobre cada región. Como los objetos no se encuentran en todas las regiones, se aplicó una desviación estándar por región, y en donde se tuvo una desviación estándar alta, fue la región o regiones donde se encuentran los objetos.

La selección de moscas aptas se realizó únicamente en el 50 % de regiones en las cuales se tuvo una desviación estándar alta. La regeneración de moscas se realizó sobre toda la imagen.

Las imágenes a utilizar se encuentran desplazadas cinco centímetros una de la otra, correspondiendo a la imagen izquierda e imagen derecha y tomadas a un metro de distancia.

En las tablas 6.1, 6.2 y 6.3 se muestran los parámetros que se utilizaron en cada una de las pruebas. Por ejemplo, en la tabla 6.1 se muestra un grupo de parámetros, con un 40 % de moscas aptas, de estas se obtiene el 10 % de moscas por cruce, el 40 % de moscas por mutación y el 10 % de moscas nuevas.

Parámetros	Porcentajes
Moscas aptas	40
Cruce	10
Mutación	40
Nuevas	10

Tabla 6.1: Porcentajes del primer grupo de parámetros.

Parámetros	Porcentajes
Moscas aptas	50
Cruce	20
Mutación	20
Nuevas	10

Tabla 6.2: Porcentajes del segundo grupo de parámetros.

Parámetros	Porcentajes
Moscas aptas	40
Cruce	20
Mutación	20
Nuevas	20

Tabla 6.3: Porcentajes del tercer grupo de parámetros.

Para obtener el umbral con base al porcentaje de moscas aptas, que fue del 90 % y 95 %, se obtuvo el número de la mosca y su valor con respecto a la función de ajuste, donde aproximadamente dichas moscas se encuentran posadas en el objeto. Se realizaron 16 pruebas por cada grupo de parámetros y se obtuvo el promedio para determinar el valor del umbral por población.

En las tablas 6.4, 6.5 y 6.6 se muestra el umbral para cada población como resultado de haber utilizado los parámetros de las tablas 6.1, 6.2 y 6.3 en el modelo:

Población	500	1500	3000	5000
Umbral	3.2	2.174909	1.950018	2.361161

Tabla 6.4: Valor de umbral para diferentes poblaciones utilizando los parámetros de la tabla 6.1.

Población	500	1500	3000	5000
Umbral	3.01216	1.572001	1.538614	2.231948

Tabla 6.5: Valor de umbral para diferentes poblaciones utilizando los parámetros de la tabla 6.2.

Población	500	1500	3000	5000
Umbral	3.525519	2.098778	2.509325	2.814001

Tabla 6.6: Valor de umbral para diferentes poblaciones utilizando los parámetros de la tabla 6.3.

6.1. Procesamiento con dos imágenes fijas

En esta prueba se realizó el procesamiento con dos imágenes, utilizando dos regiones y cuatro regiones, con tres diferentes grupos de parámetros y con diferentes tamaños de población para la selección y regeneración de las moscas.

6.1.1. Dos regiones

En la tabla 6.7 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.1, se observa que existe una diferencia de una a dos iteraciones en todas las poblaciones a excepción de la población de 500 moscas. Sin embargo, el tiempo aumenta cuando la población se incrementa.

En la figura 6.1 se presenta la reconstrucción del objeto en la vista del plano XY exhibiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.1 (c) y 6.1 (d) se obtiene una mejor reconstrucción del objeto pero en 6.1 (c) se manejan 3000 mosca y en 6.1 (d) 5000 moscas, comparando sus tiempos, al manejar 5000 moscas se tiene un tiempo alto, por lo tanto es conveniente para ésta prueba utilizar 3000 moscas.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	37	1096
	95 %	37	1096
1500	90 %	26	1674
	95 %	26	1693
3000	90 %	25	2896
	95 %	26	2996
5000	90 %	27	4743
	95 %	27	4783

Tabla 6.7: Resultados obtenidos utilizando los parámetros de la tabla 6.1.

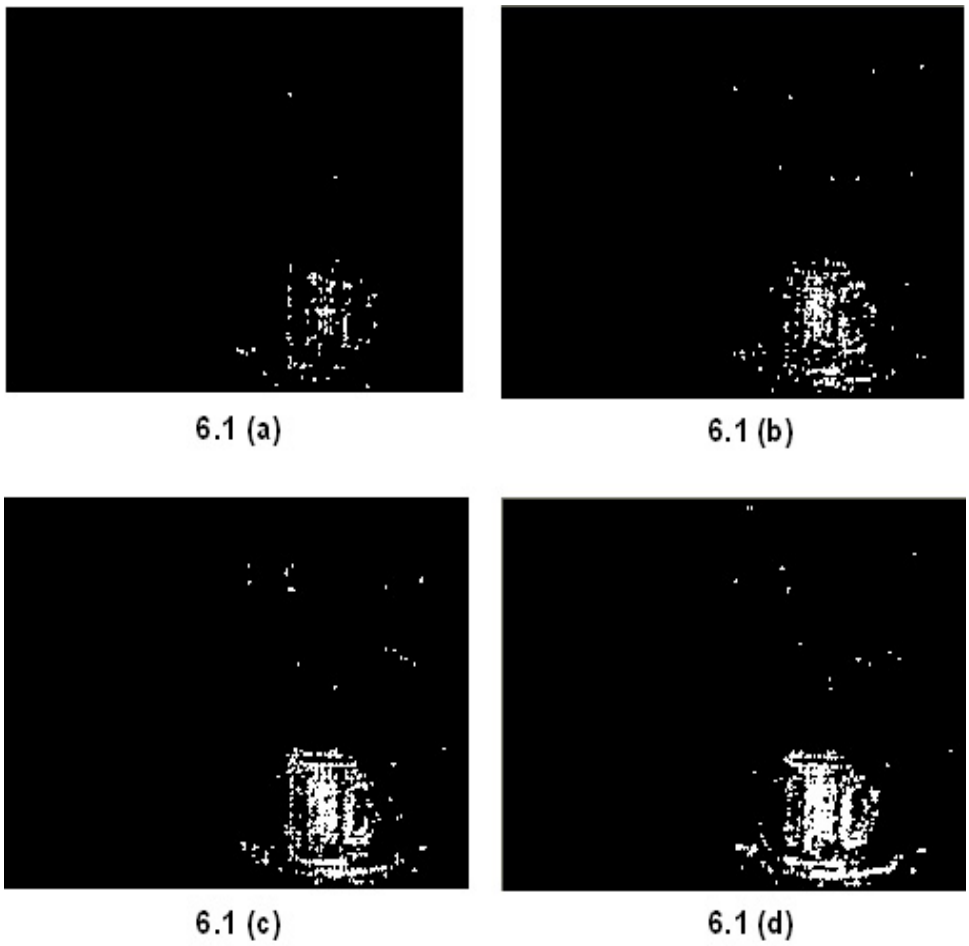

Figura 6.1: Reconstrucción del objeto aplicando los parámetros de la tabla 6.1 en la prueba de dos regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

En la tabla 6.8 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.2. Notamos que el número de iteraciones en la población de 1500 y 3000 moscas es el mismo, pero el tiempo es distinto ya que hay una diferencia de 1157 ms para el concentrado de un 90 % de moscas aptas y 1219 ms para el concentrado de un 95 % de moscas aptas.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	27	813
	95 %	28	818
1500	90 %	22	1427
	95 %	23	1455
3000	90 %	22	2584
	95 %	23	2674
5000	90 %	26	4796
	95 %	27	4868

Tabla 6.8: Resultados obtenidos utilizando los parámetros de la tabla 6.2.

En la figura 6.2 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.2 (c) y 6.2 (d) se obtiene una mejor reconstrucción del objeto pero en 6.2 (c) se manejan 3000 mosca y en 6.2 (d) 5000 moscas. Comparando sus tiempos, al manejar 5000 moscas se incrementa casi el doble, por lo tanto es conveniente para esta prueba utilizar 3000 moscas.

En la tabla 6.9 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.3. Notamos que existe una diferencia de una a tres iteraciones entre cada una de las poblaciones. Sin embargo, el tiempo aumenta cuando la población se incrementa.

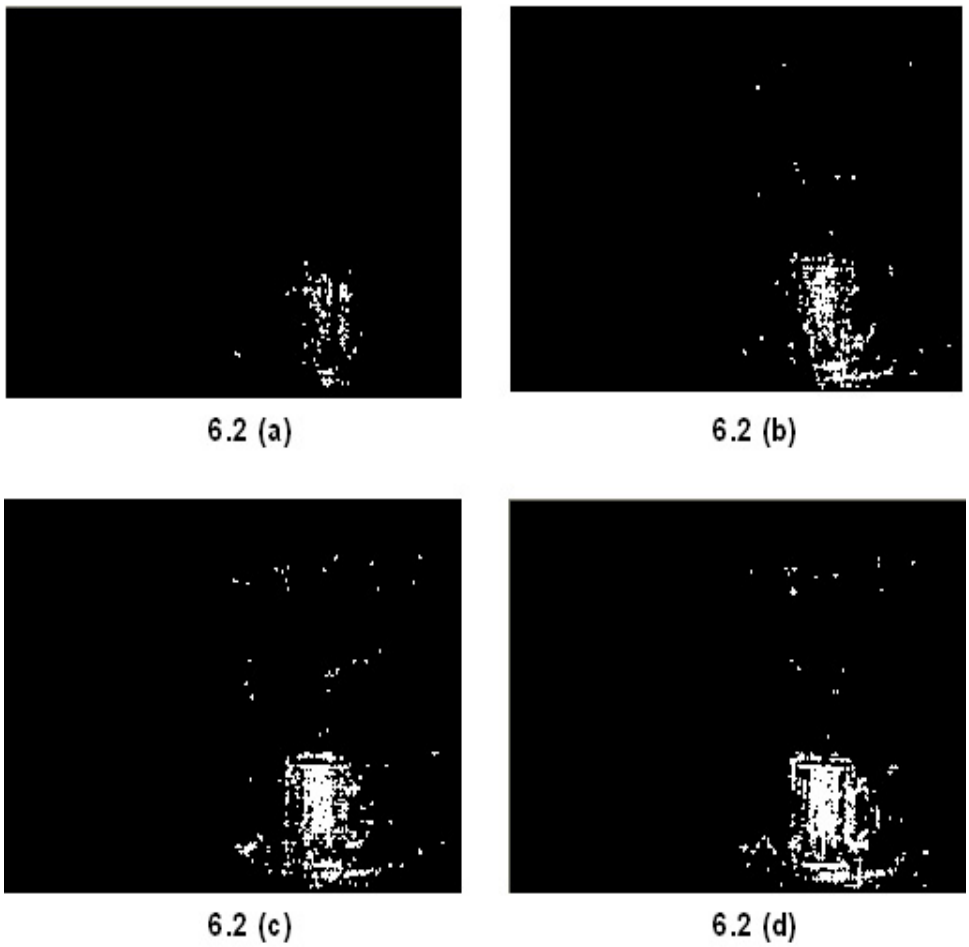

Figura 6.2: Reconstrucción del objeto aplicando los parámetros de la tabla 6.2 en la prueba de dos regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	21	653
	95 %	23	709
1500	90 %	19	1231
	95 %	20	1293
3000	90 %	21	2424
	95 %	21	2437
5000	90 %	21	3793
	95 %	22	4048

Tabla 6.9: Resultados obtenidos utilizando los parámetros de la tabla 6.3.

En la figura 6.3 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.3 (c) y 6.3 (d) se obtiene una mejor reconstrucción del objeto pero en 6.3 (c) se manejan 3000 mosca y en 6.3 (d) 5000 moscas. Comparando sus tiempos, el manejar 5000 moscas se incrementa casi al doble tiempo, por lo tanto es conveniente para esta prueba utilizar 3000 moscas.

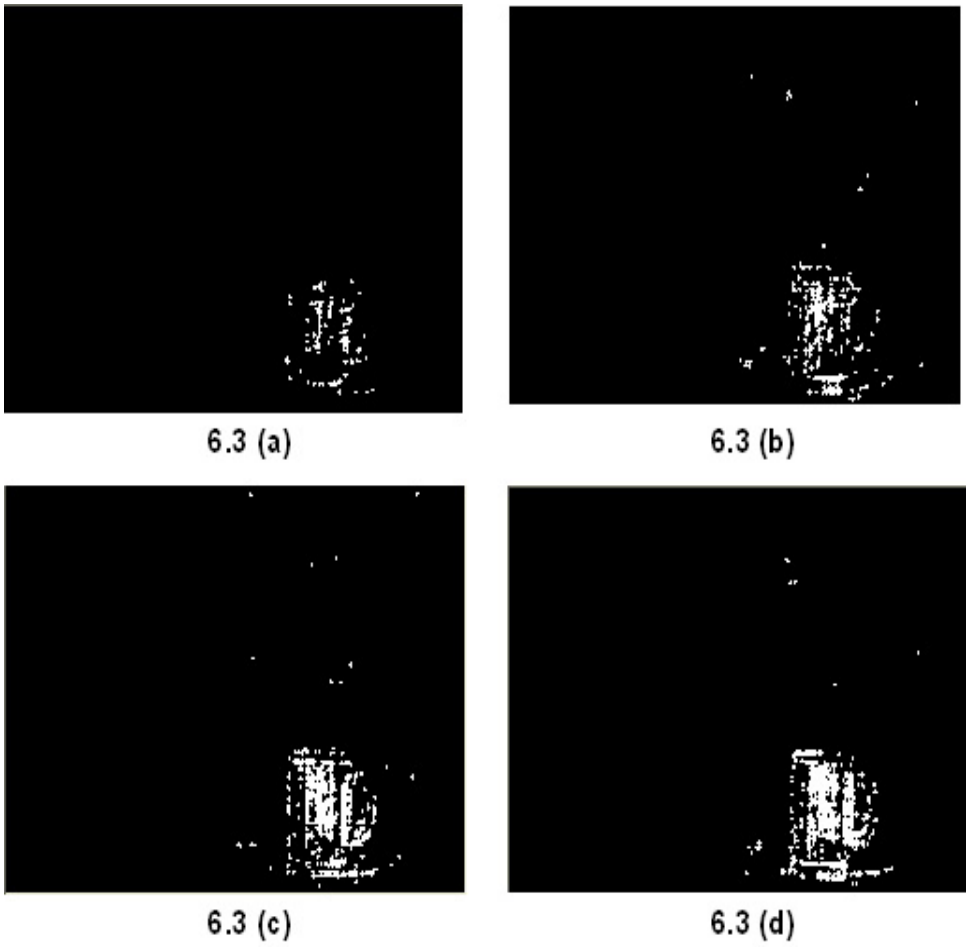

Figura 6.3: Reconstrucción del objeto aplicando los parámetros de la tabla 6.3 en la prueba de dos regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

En las figuras 6.4 y 6.5 se visualizan las gráficas de los resultados de haber aplicado las tablas 6.1, 6.2 y 6.3 con diferentes números de población en el modelo, para obtener la reconstrucción con un porcentaje de 90 % y 95 % de moscas aptas en dos regiones. Se observa que en ambas gráficas el tiempo se incrementa utilizando los parámetros de la tabla 6.1, y además en ambas gráficas con la población de 5000 moscas el tiempo se incrementa utilizando los parámetros de la tabla 6.2.

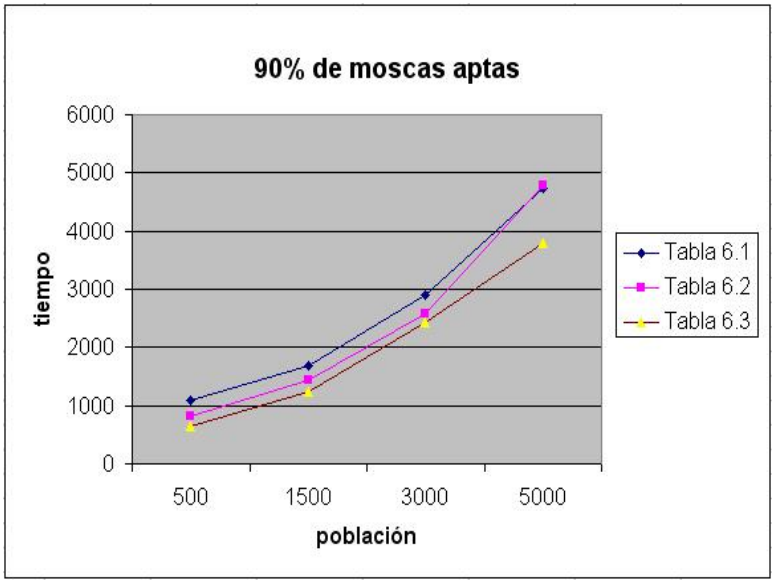

Figura 6.4: Gráfica del 90 % de moscas aptas para dos regiones.

Analizando las figuras 6.1, 6.2 y 6.3 se puede concluir que se obtiene una mejor reconstrucción utilizando cualquier tabla de parámetros con una población de 3000 y 5000 moscas, y se observa que es suficiente utilizar 3000 moscas.

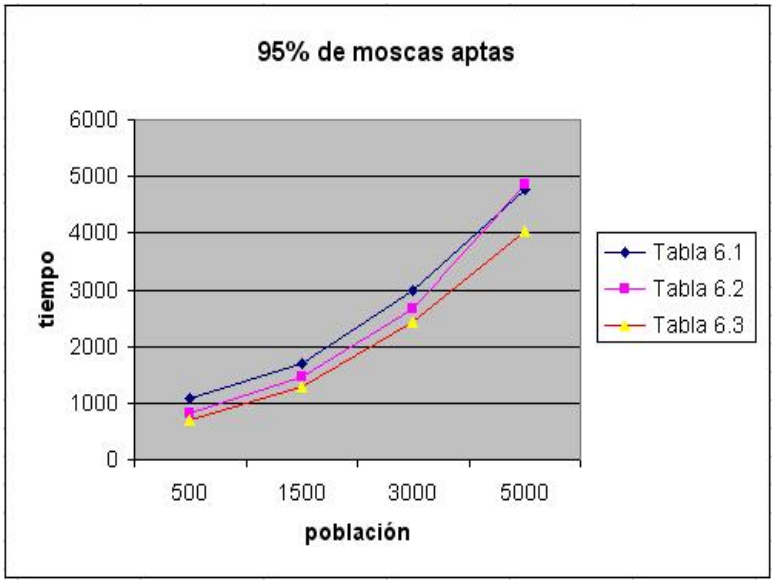

Figura 6.5: Gráfica del 95 % de moscas aptas para dos regiones.

6.1.2. Cuatro regiones

En la tabla 6.10 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.1. Se observa que en las poblaciones de 1500 y 3000 moscas existe una diferencia de una iteración en el concentrado del 90 % y en el concentrado del 95 % el número de iteraciones es el mismo. El tiempo aumenta cuando la población se incrementa.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 % 95 %	26 30	787 906
1500	90 % 95 %	21 22	1368 1449
3000	90 % 95 %	20 22	2294 2490
5000	90 % 95 %	23 24	4180 4409

Tabla 6.10: Resultados obtenidos utilizando los parámetros de la tabla 6.1.

En la figura 6.6 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.6 (c) y 6.6 (d) se obtiene una mejor reconstrucción del objeto pero es suficiente manejar 3000 moscas, ya que al usar un mayor número de moscas también habrá un mayor tiempo como se observó en la tabla.

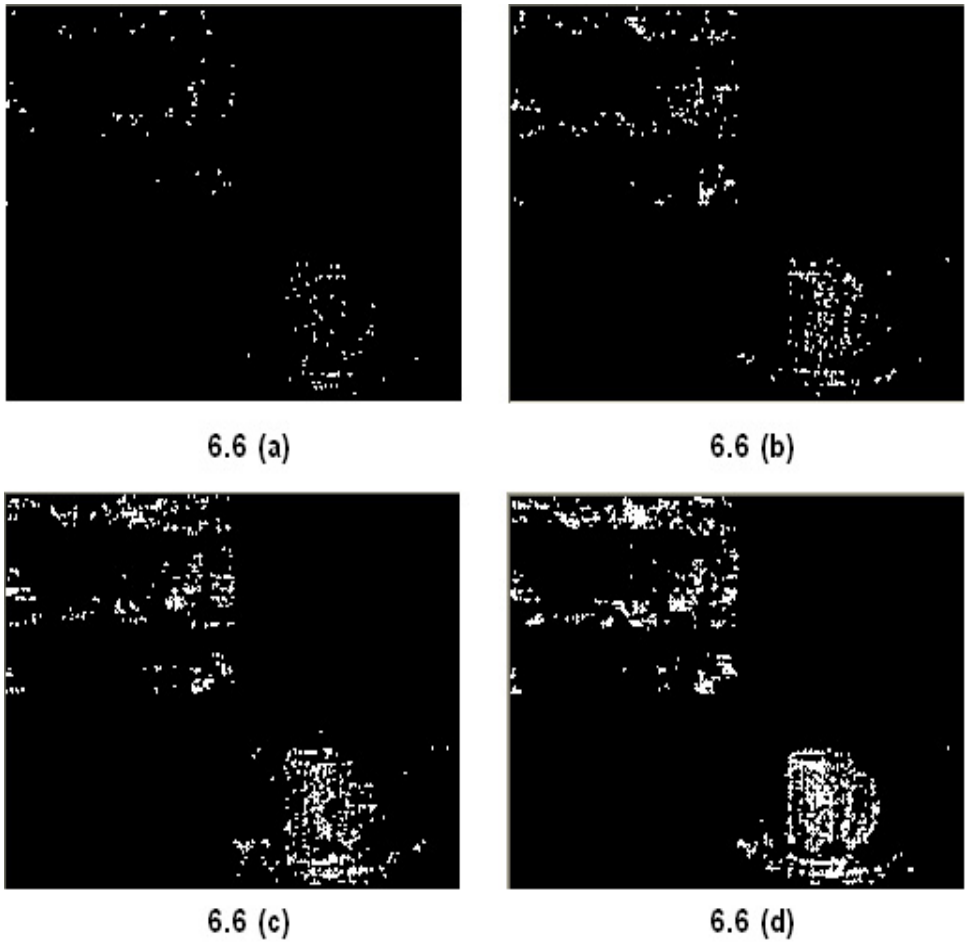

Figura 6.6: Reconstrucción del objeto aplicando los parámetros de la tabla 6.1 en la prueba de cuatro regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

En la tabla 6.11 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.2. Notamos que el número de iteraciones en la población de 1500 y 3000 moscas es el mismo en el concentrado del 95 % de moscas aptas y en el concentrado de 90 % de moscas aptas, existe una diferencia de una iteración. El tiempo aumenta cuando la población se incrementa.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	25	768
	95 %	28	834
1500	90 %	19	1259
	95 %	22	1412
3000	90 %	20	2309
	95 %	22	2566
5000	90 %	24	4262
	95 %	25	4578

Tabla 6.11: Resultados obtenidos utilizando los parámetros de la tabla 6.2.

En la figura 6.7 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.7 c y 6.7 (d) se obtiene una mejor reconstrucción del objeto pero es suficiente utilizar 3000 moscas, ya que al usar un mayor número de moscas también habrá un mayor tiempo como se observó en la tabla.

En la tabla 6.12 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.3. Notamos que el número de iteraciones en la población de 500 y 5000 moscas es igual en el concentrado del 90 %, y además es igual en la población de 1500 moscas con un concetrado del 95 % y la población de 3000 moscas con un concentrado del 90 %. El tiempo aumenta cuando la población se incrementa.

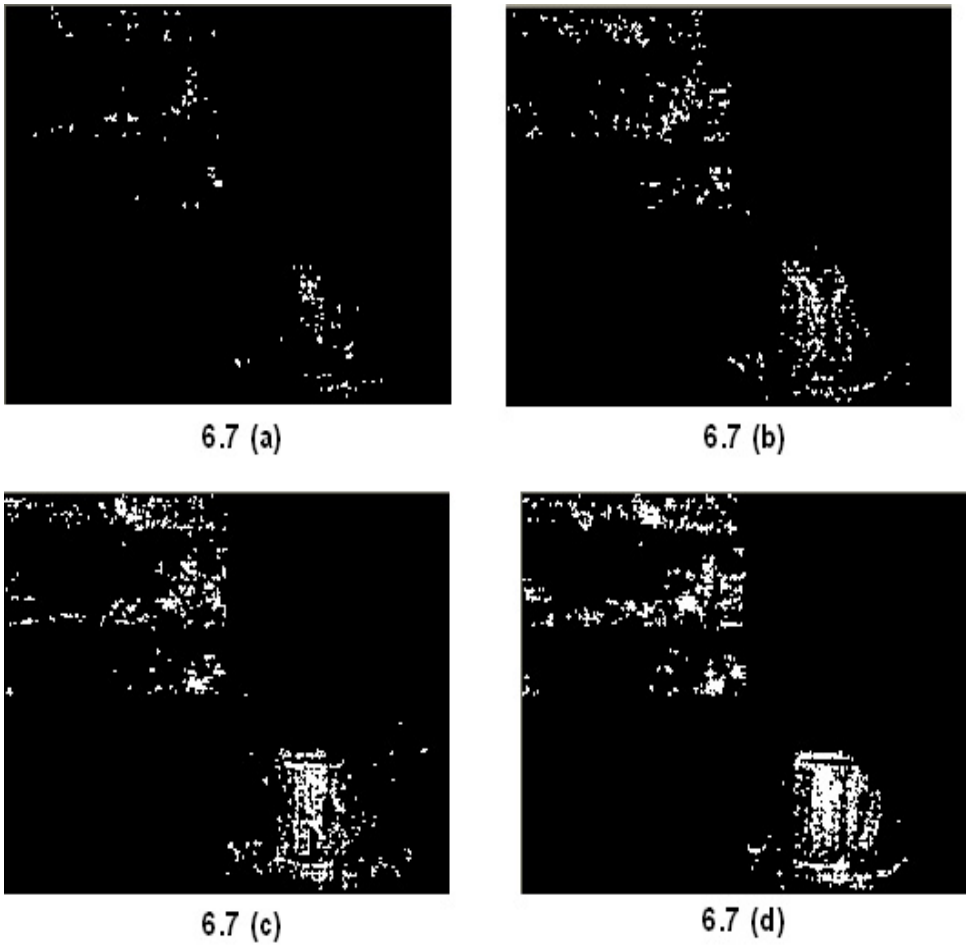

Figura 6.7: Reconstrucción del objeto aplicando los parámetros de la tabla 6.2 en la prueba de cuatro regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	21	665
	95 %	23	696
1500	90 %	18	1193
	95 %	19	1249
3000	90 %	19	2124
	95 %	20	2336
5000	90 %	21	3818
	95 %	22	3827

Tabla 6.12: Resultados obtenidos utilizando los parámetros de la tabla 6.3.

En la figura 6.8 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.8 (c) y 6.8 (d) se obtiene una mejor reconstrucción del objeto pero en 6.8 (c) se manejan 3000 moscas y en 6.8 (d) 5000 moscas, comparando sus tiempos, el manejar 5000 moscas se tiene un tiempo largo, por lo tanto es conveniente para esta prueba utilizar 3000 moscas.

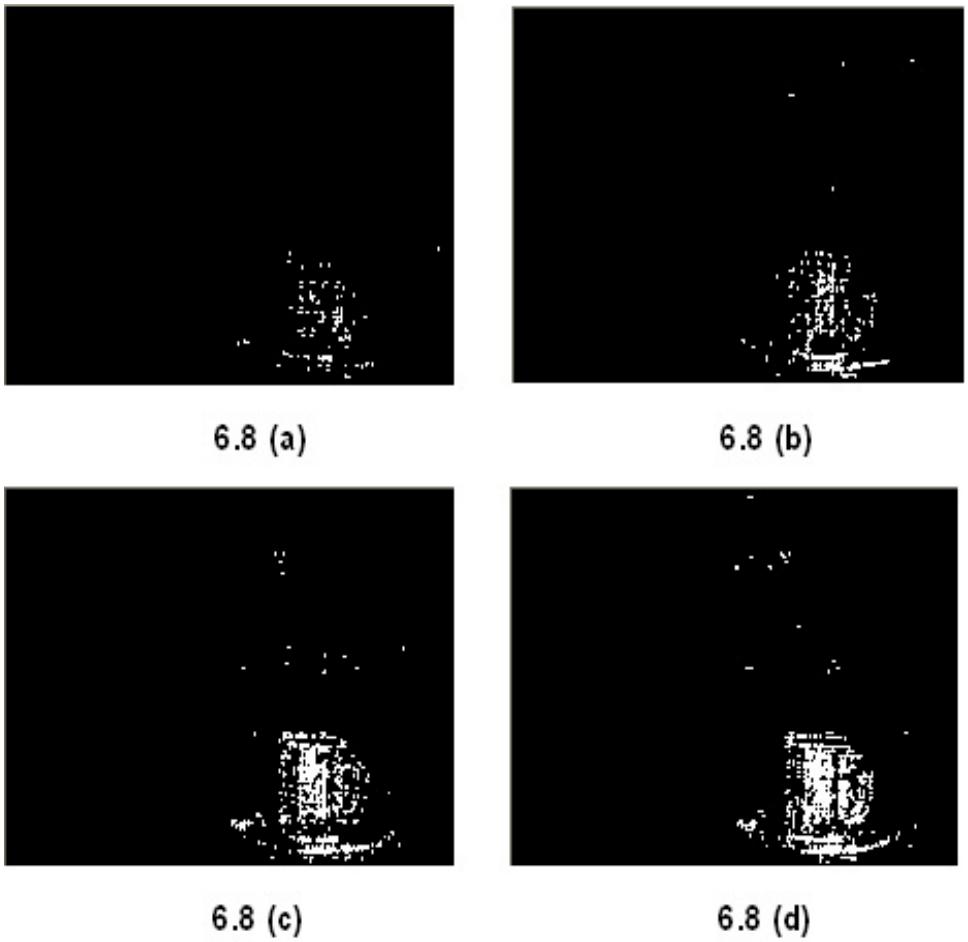

Figura 6.8: Reconstrucción del objeto aplicando los parámetros de la tabla 6.3 en la prueba de cuatro regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

En las figuras 6.9 y 6.10 se visualizan las gráficas de los resultados de haber aplicado las tablas 6.1, 6.2 y 6.3 con diferentes números de población en el modelo, para obtener la reconstrucción con un porcentaje de concentración del 90 % y 95 % de moscas aptas en cuatro regiones. Se observa que al inicio de las pruebas, el uso de los parámetros de la tabla 6.1 se obtiene el tiempo más alto, pero incrementando la población, el tiempo se incrementa en donde se utilizan los parámetros de la tabla 6.2.

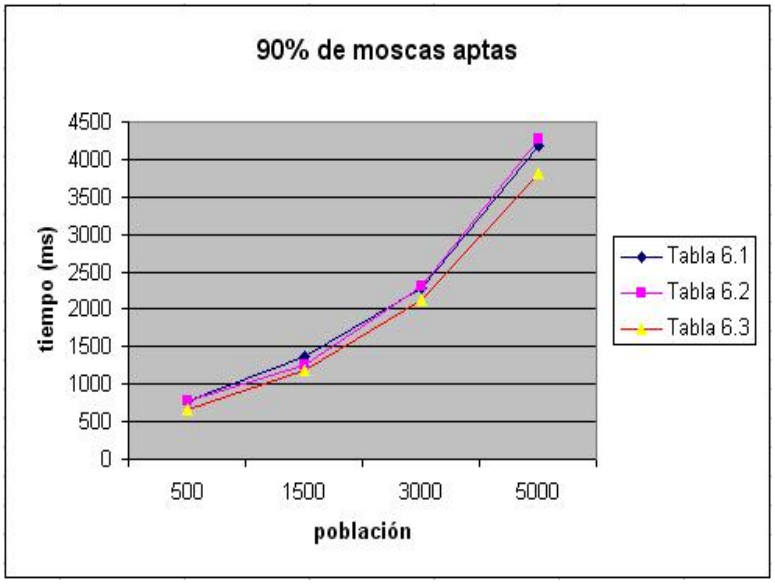

Figura 6.9: Gráfica del 90 % de moscas aptas para cuatro regiones.

Analizando las figuras 6.6, 6.7 y 6.8 y las gráficas de las figuras 6.9 y 6.10 se muestra una mejor reconstrucción utilizando los parámentros de la tabla 6.3, con el menor tiempo de ejecución, y además observamos que al utilizar un grupo de moscas pequeño, por ejemplo 500 moscas el tiempo de cálculo es pequeño, pero no abarca lo suficiente al objeto y al utilizar una población de moscas grande por ejemplo 5000 moscas se abarca lo suficiente al objeto, pero el tiempo de cálculo es grande. Para esta prueba fue suficiente utilizar 3000 moscas, ya que fue una de las poblaciones que cubren adecuadamente al objeto.

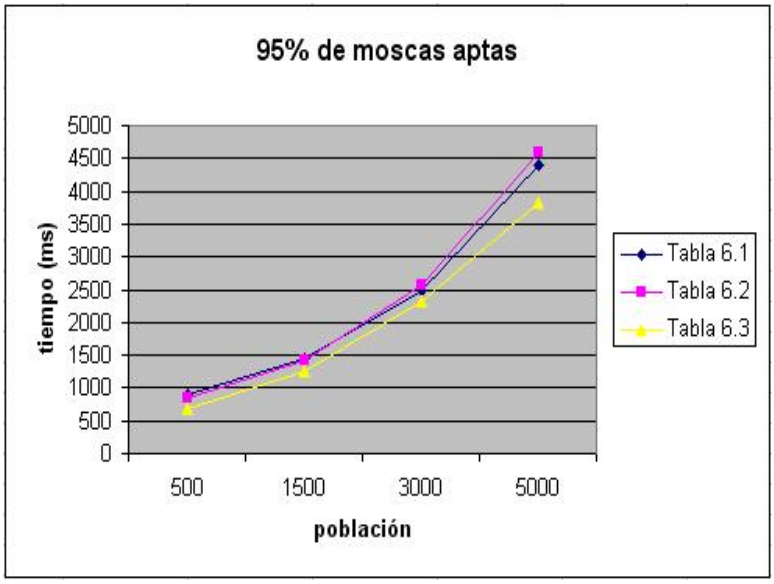

Figura 6.10: Gráfica del 95 % de moscas aptas para cuatro regiones.

6.2. Procesamiento de una secuencia de imágenes

En esta prueba se realizó la reconstrucción con un conjunto de cinco imágenes que el modelo fue adquiriendo, utilizando dos regiones y cuatro regiones, con tres diferentes valores de parámetros y con diferentes tamaños de población para la selección y regeneración de las moscas.

6.2.1. Dos regiones

En la tabla 6.13 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.1. Se observa que el número de iteraciones es variado y se va incrementando cuando se incrementa la población, a excepción cuando la población es de 500 moscas, ya que es la población donde se realiza el número de iteraciones más alto. Además se observa que el tiempo aumenta, cuando la población se incrementa.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	100	2993
	95 %	105	3105
1500	90 %	63	4106
	95 %	76	4903
3000	90 %	80	9212
	95 %	83	9496
5000	90 %	85	15531
	95 %	88	16156

Tabla 6.13: Resultados obtenidos utilizando los parámetros de la tabla 6.1.

En la figura 6.11 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.11 (c) y 6.11 (d) se obtiene una mejor reconstrucción del objeto. Comparando sus tiempos, al manejar 5000 moscas se tiene un tiempo alto, por lo tanto es suficiente para esta prueba utilizar 3000 moscas.

En la tabla 6.14 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.2. Se observa que el número de iteraciones en la población de 1500 y 3000 moscas es similar, pero el tiempo es distinto. Además se observa que el tiempo aumenta cuando la población se incrementa.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	73	2238
	95 %	86	2587
1500	90 %	60	3943
	95 %	63	4090
3000	90 %	61	6968
	95 %	62	7162
5000	90 %	70	12534
	95 %	71	12875

Tabla 6.14: Resultados obtenidos utilizando los parámetros de la tabla 6.2.

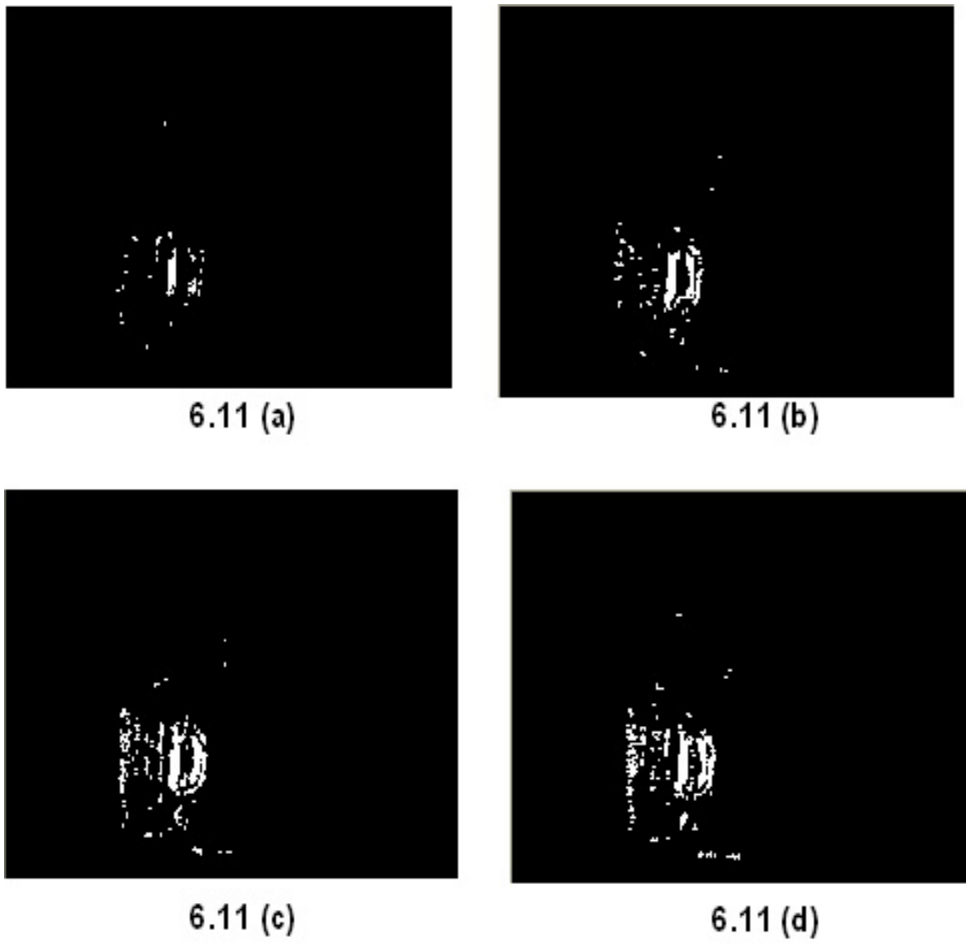

Figura 6.11: Reconstrucción del objeto aplicando los parámetros de la tabla 6.1 en la prueba de dos regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

En la figura 6.12 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en la figura 6.12 (d) se obtiene una mejor reconstrucción del objeto.

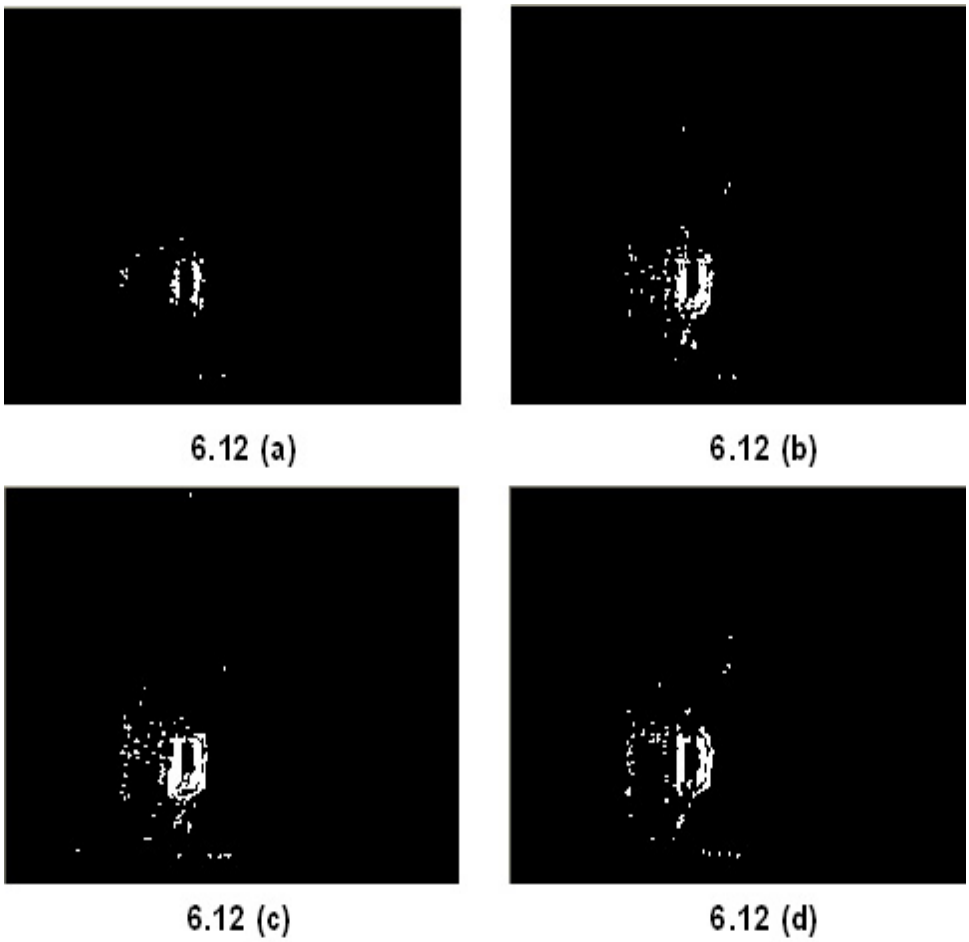

Figura 6.12: Reconstrucción del objeto aplicando los parámetros de la tabla 6.2 en la prueba de dos regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

En la tabla 6.15 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.3. Observamos que el número de iteraciones para cada una de las poblaciones varía de una a cuatro iteraciones. Además se observa que el tiempo aumenta cuando la población se incrementa.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	70	2125
	95 %	72	2156
1500	90 %	54	3521
	95 %	58	3728
3000	90 %	59	6767
	95 %	62	6855
5000	90 %	61	11134
	95 %	63	11318

Tabla 6.15: Resultados obtenidos utilizando los parámetros de la tabla 6.3.

En la figura 6.13 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en la figura 6.13 (d) se obtiene una mejor reconstrucción del objeto.

En las figuras 6.14 y 6.15 se visualizan las gráficas de los resultados de haber aplicado las tablas 6.1, 6.2 y 6.3 con diferentes números de población en el sistema, para obtener la reconstrucción con un porcentaje de concentración del 90 % y 95 % de moscas aptas en dos regiones. Se observa que los parámetros de la tabla 6.1 involucran más tiempo.

Analizando las figuras 6.11, 6.12 y 6.13 y las gráficas de las figuras 6.14 y 6.15, se concluye que se obtiene una mejor reconstrucción utilizando los parámetros de la tabla 6.1 con 3000 moscas, no obstante que el tiempo es el más largo.

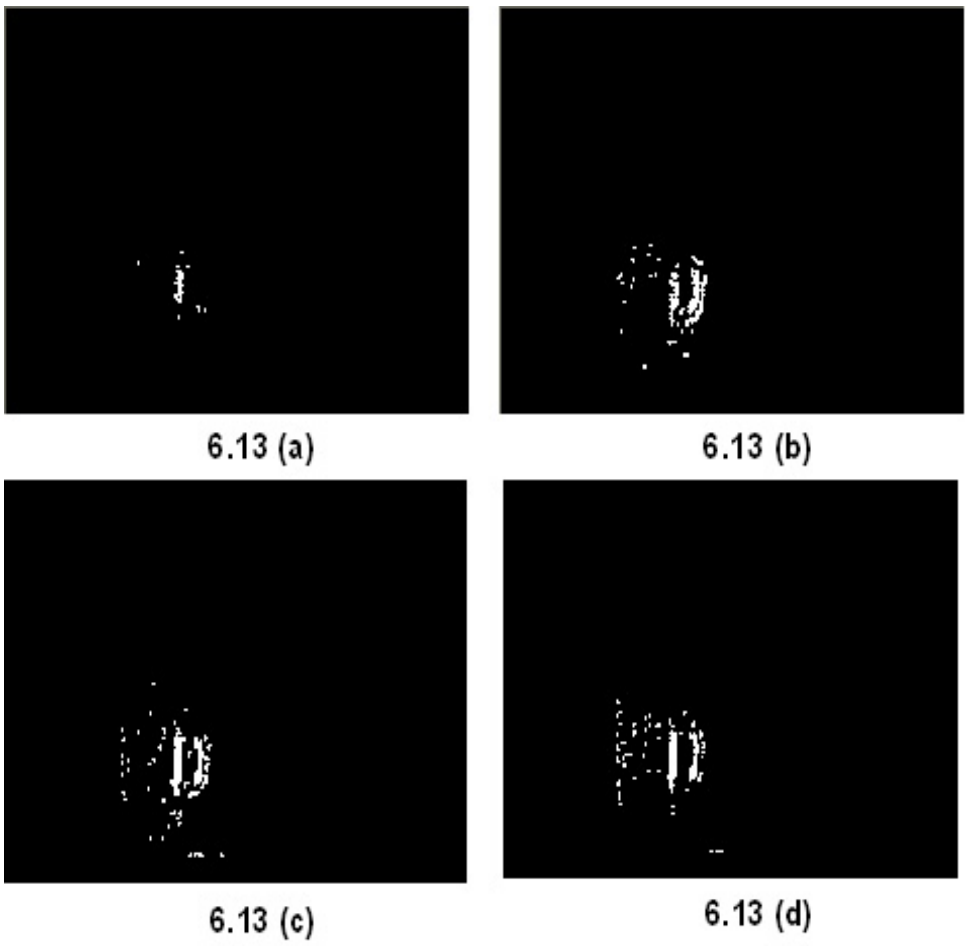

Figura 6.13: Reconstrucción del objeto aplicando los parámetros de la tabla 6.3 en la prueba de dos regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

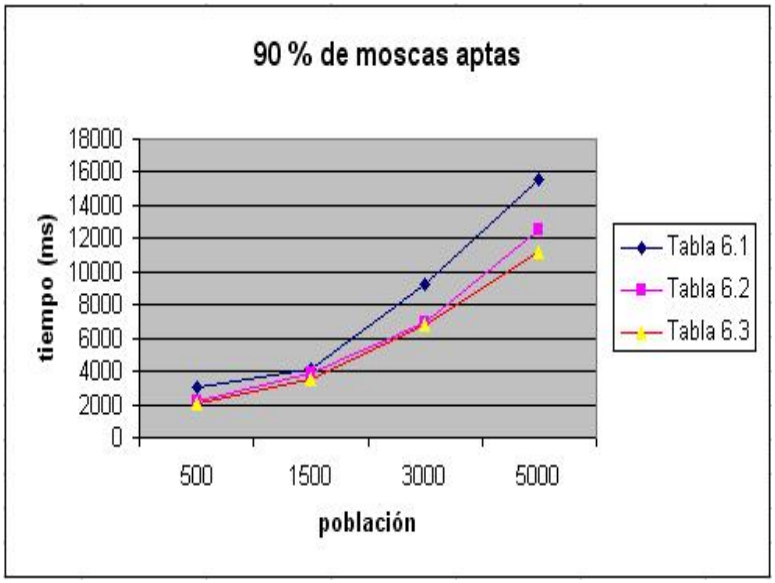

Figura 6.14: Gráfica del 90 % de moscas aptas para dos regiones.

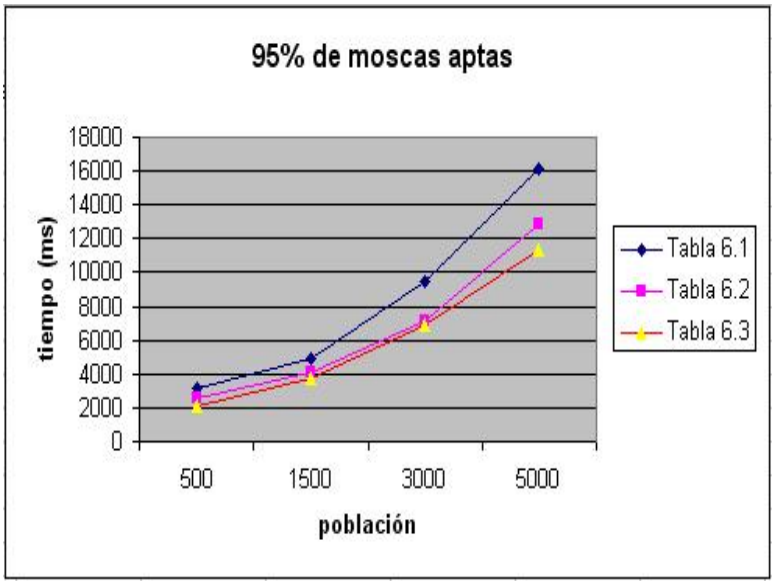

Figura 6.15: Gráfica del 95 % de moscas aptas para dos regiones.

6.2.2. Cuatro regiones

En la tabla 6.16 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.1. Se observa que el número menor de iteraciones sucede en la población con 3000 moscas, y además se observa que el tiempo aumenta cuando la población se incrementa.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	81	2415
	95 %	94	2731
1500	90 %	80	4940
	95 %	83	5143
3000	90 %	72	7915
	95 %	78	8577
5000	90 %	83	14458
	95 %	88	15334

Tabla 6.16: Resultados obtenidos utilizando los parámetros de la tabla 6.1.

En la figura 6.16 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.16 (c) y 6.16 (d) se obtiene una mejor reconstrucción del objeto pero es suficiente manejar 3000 moscas, ya que al utilizar un mayor número de moscas se requerirá un mayor tiempo, como se observó en la tabla.

En la tabla 6.17 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.2. Observamos que en la población de 1500 y 3000 moscas existe una diferencia de dos iteraciones. Además se observa que el tiempo aumenta cuando la población se incrementa.

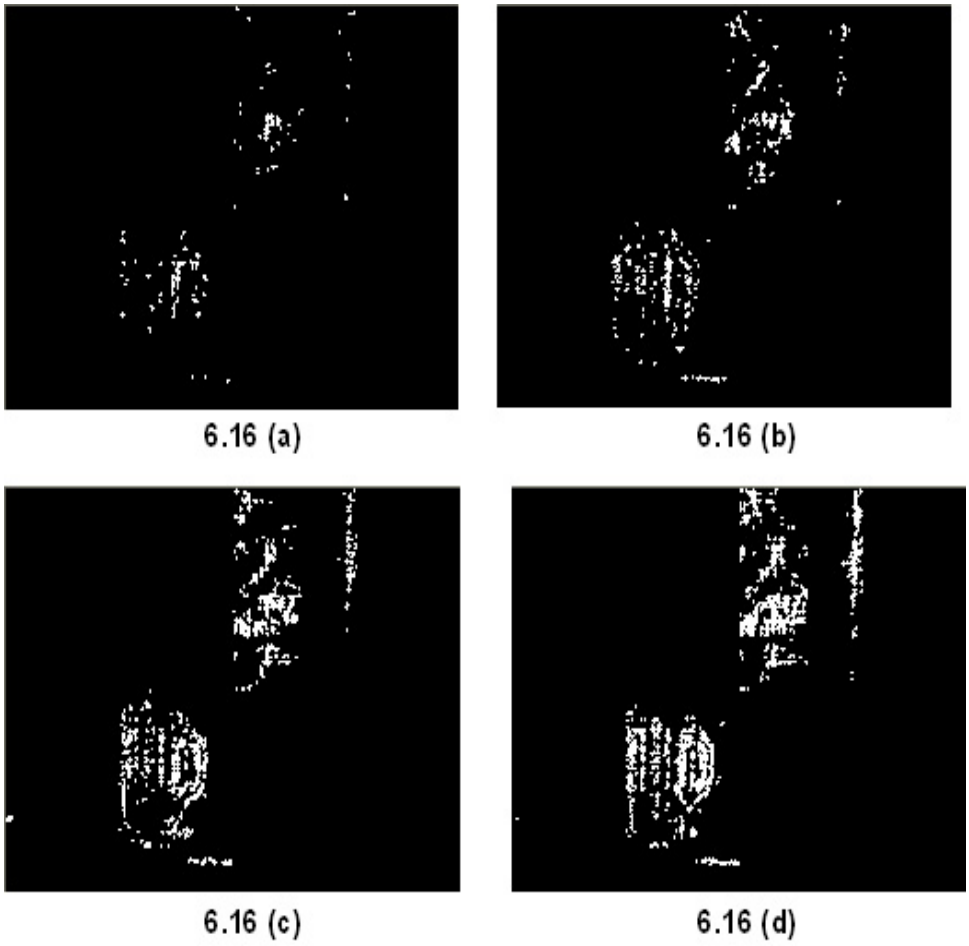

Figura 6.16: Reconstrucción del objeto aplicando los parámetros de la tabla 6.1 en la prueba de cuatro regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	78	2309
	95 %	93	2696
1500	90 %	56	3534
	95 %	61	3805
3000	90 %	54	6047
	95 %	59	6581
5000	90 %	72	12453
	95 %	74	13512

Tabla 6.17: Resultados obtenidos utilizando los parámetros de la tabla 6.2.

En la figura 6.17 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.17 (c) y 6.17 (d) se obtiene una mejor reconstrucción, pero es preferible manejar una población menor ya que el tiempo es menor, por lo tanto es suficiente utilizar la población de 3000 moscas.

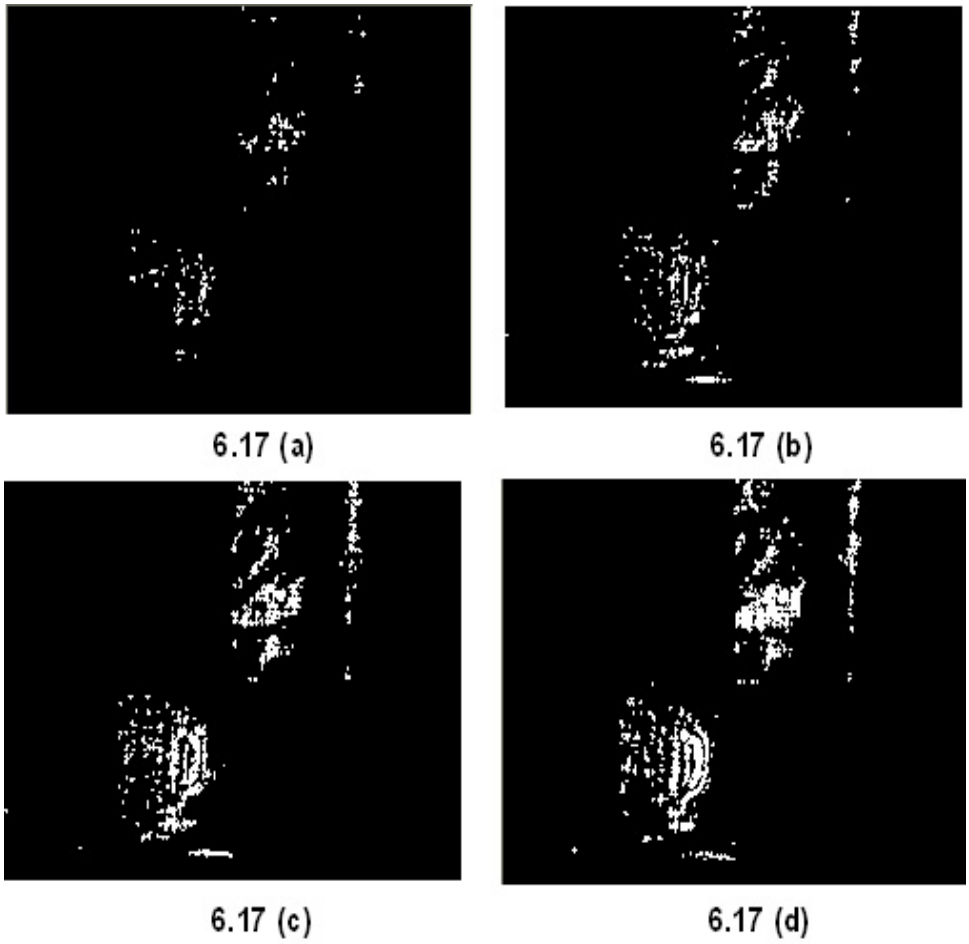

Figura 6.17: Reconstrucción del objeto aplicando los parámetros de la tabla 6.2 en la prueba de cuatro regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

En la tabla 6.18 se muestran los resultados obtenidos utilizando los parámetros de la tabla 6.3. Notamos que el número de iteraciones en la población de 1500 y 3000 moscas tienen una variación de cinco y seis iteraciones y la población de 3000 y 5000 moscas tienen una variación de cuatro iteraciones, solo donde se maneja la concentración del 90 % de moscas aptas. Además se observa que el tiempo aumenta cuando la población se incrementa.

No. de moscas	Moscas aptas	No. de iteraciones	Tiempo (ms)
500	90 %	82	2500
	95 %	84	2549
1500	90 %	58	3834
	95 %	62	4068
3000	90 %	63	7353
	95 %	68	7800
5000	90 %	67	12077
	95 %	68	12849

Tabla 6.18: Resultados obtenidos utilizando los parámetros de la tabla 6.3.

En la figura 6.18 se presenta la reconstrucción del objeto en la vista del plano XY exponiendo diferentes tamaños de población y el concentrado de un 90 % de moscas aptas. Se observa que en las figuras 6.18 (c) y 6.18 (d) se obtiene una mejor reconstrucción del objeto pero en 6.18 (c) se manejan 3000 moscas y en 6.18 (d) 5000 moscas. Comparando sus tiempos, al manejar 5000 moscas se tiene un tiempo alto, por lo tanto es suficiente para esta prueba utilizar 3000 moscas.

En las figuras 6.19 y 6.20 se visualizan las gráficas de los resultados obtenidos utilizando los parámetros las tablas 6.1, 6.2 y 6.3 con diferentes números de población en el modelo, para obtener la reconstrucción con un porcentaje de concentración del 90 % y 95 % de moscas aptas en cuatro regiones. Se observa que al utilizar los parámetros de la tabla 6.1 se involucra más tiempo.

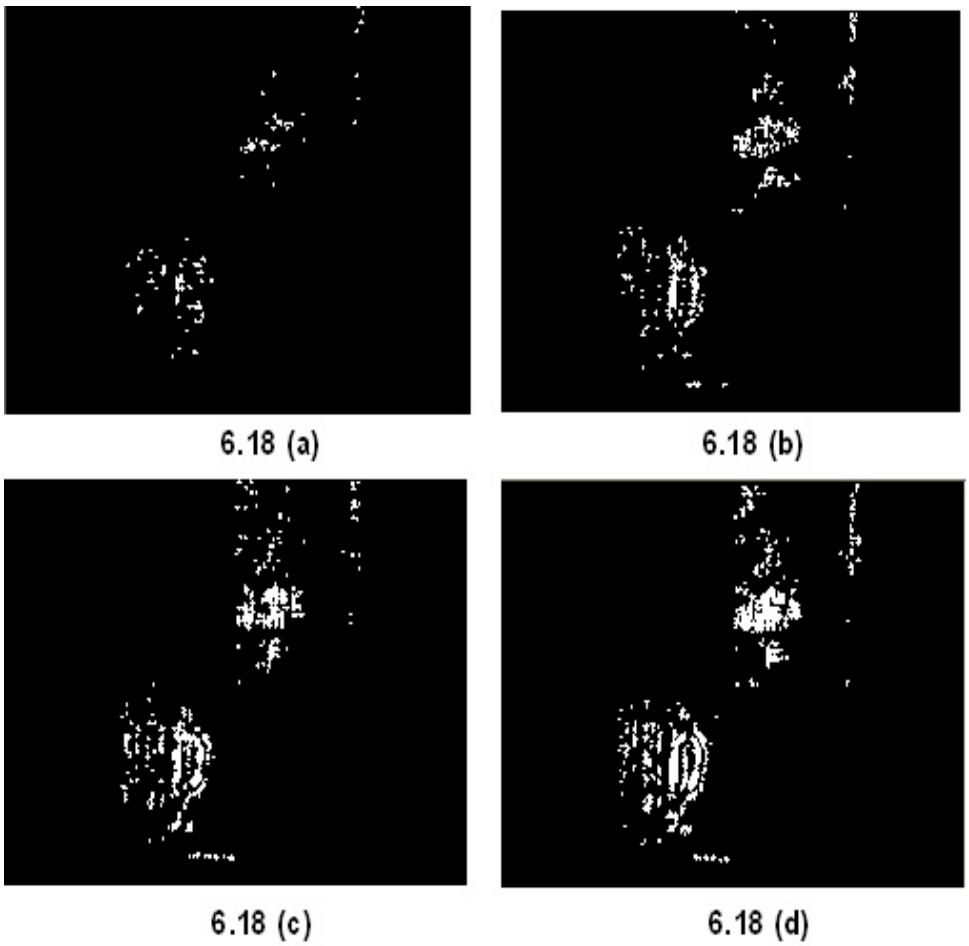

Figura 6.18: Reconstrucción del objeto aplicando los parámetros de la tabla 6.3 en la prueba de cuatro regiones con (a) 500 moscas, (b) 1500 moscas, (c) 3000 moscas y (d) 5000 moscas.

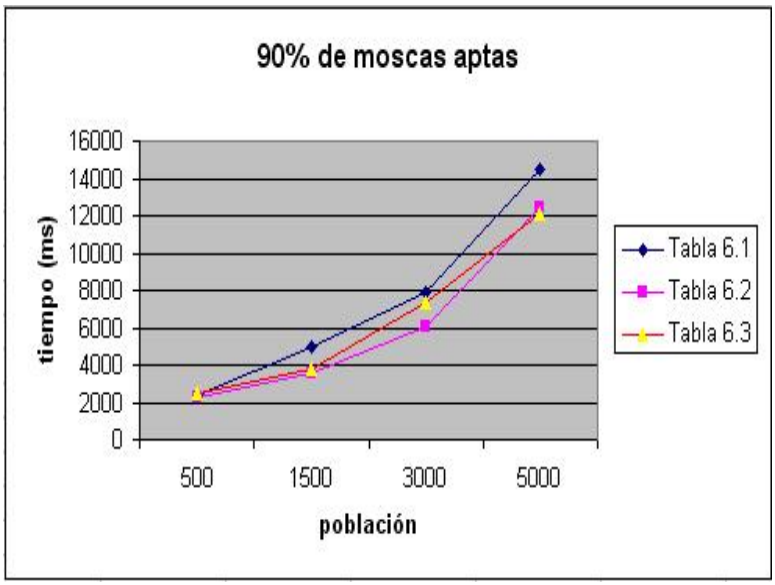

Figura 6.19: Gráfica del 90 % de moscas aptas para cuatro regiones.

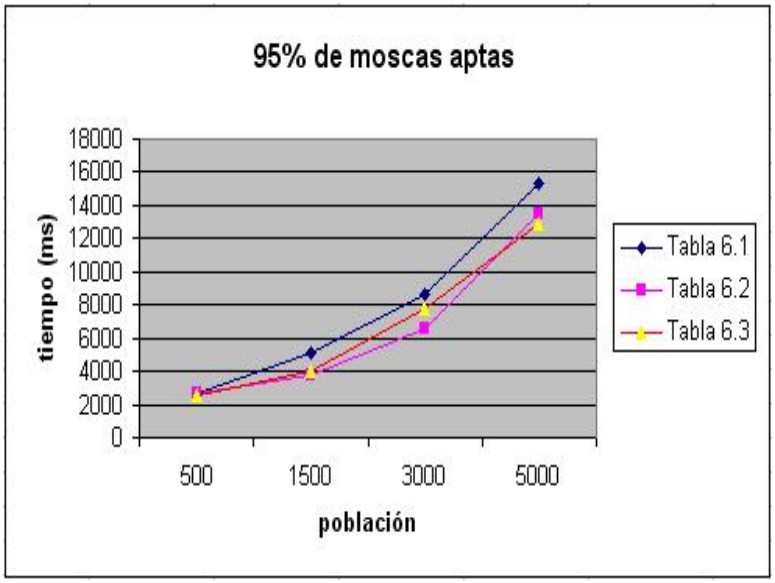

Figura 6.20: Gráfica del 95 % de moscas aptas para cuatro regiones.

Analizando las figuras 6.16, 6.17 y 6.18 y las gráficas 6.19 y 6.20, se concluye que se obtiene una mejor reconstrucción utilizando los parámetros de la tabla 6.1, con 3000 moscas, no obstante que el tiempo es el más largo.

Observamos que al utilizar una población de moscas pequeña, el tiempo de cálculo es pequeño, pero no abarca lo suficiente al objeto y al utilizar una población de moscas grande, el tiempo de cálculo es alto y abarca lo suficiente al objeto. En esta prueba fue suficiente utilizar 3000 moscas utilizando los parámetros de la tabla 6.1 y utilizando cuatro regiones, ya que permitió que la población mantuviera una mejor distribución de las moscas, obteniendo como resultado una mejor reconstrucción, no obstante que el tiempo fue el más alto.

6.3. Procesamiento durante la navegación del robot

Para esta situación se contemplaron tres estrategias de desplazamiento del robot para probar el modelo, las cuales son:

- Desplazamiento horizontal.
- Desplazamiento sobre el eje Z.
- Desplazamiento horizontal y sobre el eje Z.

Por los resultados de las pruebas mencionadas anteriormente, se emplearon los parámetros de la tabla 6.1, se observó que fueron los parámetros que mejor realizaron la reconstrucción del objeto utilizando una población de 3000 moscas en 4 regiones para obtener una mejor distribución de las moscas.

Se utilizó un robot de Lego en el cual se colocó una cámara. El robot fue desplazado en un área marcada con líneas negras, la distancia que separa a éstas líneas una de la

otra es de cinco centímetros aproximadamente. En cada línea negra, el robot se detuvo un tiempo, en el cual capturó la imagen que fue utilizada en el sistema siguiendo el diagrama de la figura 3.2.

6.3.1. Desplazamiento horizontal

En la figura 6.21 se muestra el estado inicial y el estado final del robot en el desplazamiento horizontal.

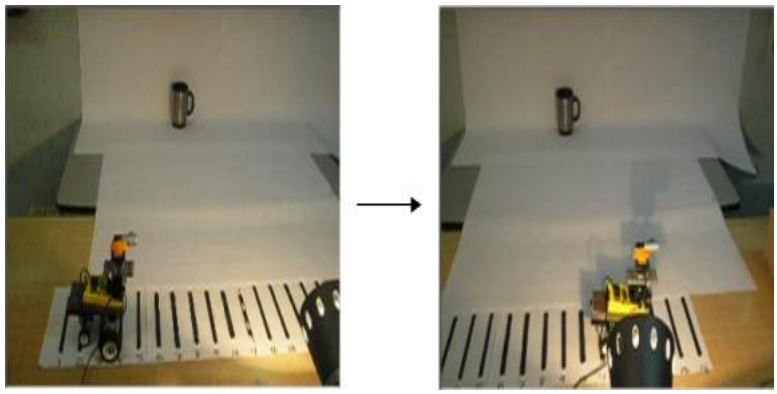

Figura 6.21: Desplazamiento del robot sobre el eje horizontal.

En la figura 6.22 se muestra la ubicación inicial de las moscas proyectadas en la imagen izquierda e imagen derecha con los parámetros de la tabla 6.1, utilizando 3000 moscas y cuatro regiones para obtener una mejor distribución.

En la figura 6.23 se presenta el resultado de la ejecución del modelo una vez que el robot ha finalizado su recorrido sobre el eje horizontal. Observamos que las moscas aptas estan presentes solo en el 50 % de las regiones. Además se observa la concentración de moscas sobre el objeto.

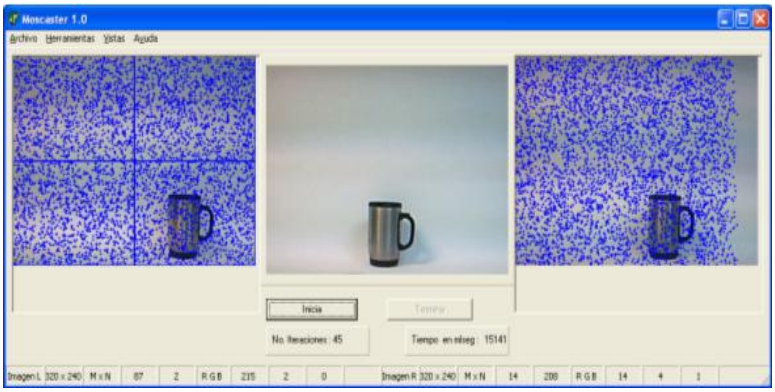

Figura 6.22: Ubicación inicial de las moscas proyectadas en la imagen izquierda e imagen derecha.

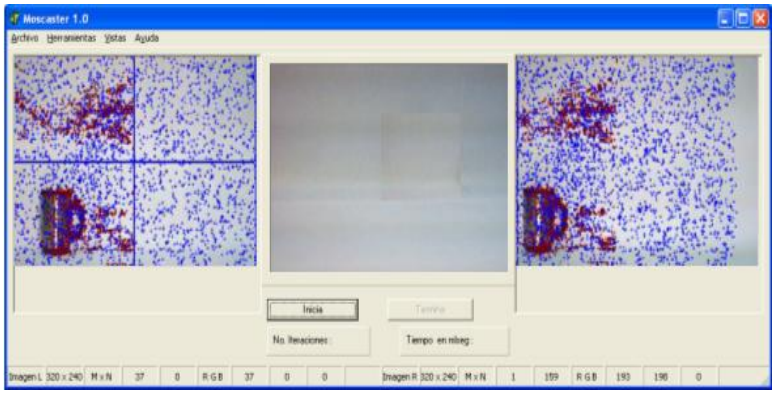

Figura 6.23: Resultado del recorrido sobre el eje horizontal.

Las vistas *XY*, *XZ* y *YZ* se muestran en las figuras 6.24 (a), 6.25 (a) y 6.26 (a) sin ninguna operación de mejora de la visualización. En 6.24 (b), 6.25 (b) y 6.26 (b) se observan después de aplicar una operación de apertura y una de cerradura.

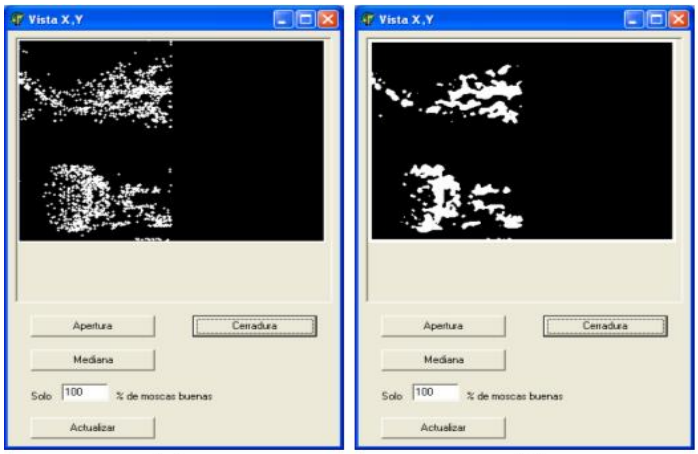

Figura 6.24: (a) Vista *XY*, (b) Después de aplicar una operación de apertura y una de cerradura.

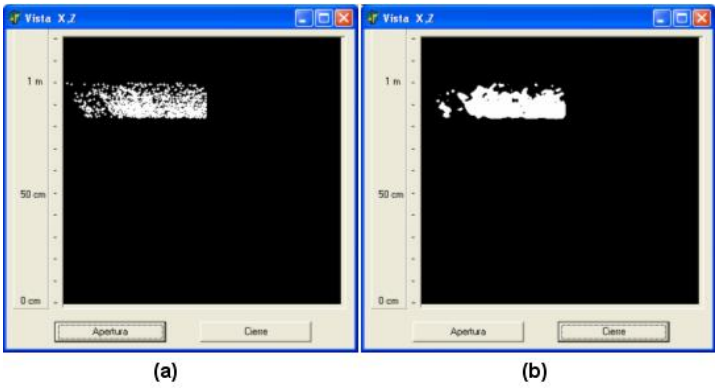

Figura 6.25: (a) Vista *XZ*, (b) Después de aplicar una operación de apertura y una de cerradura.

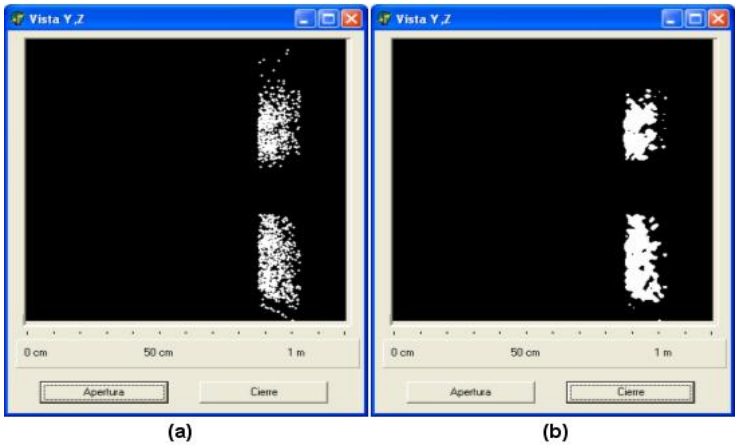

Figura 6.26: (a) Vista *YZ*, (b) Después de aplicar una operación de apertura y una de cerradura .

En la figura 6.27 se muestra la vista tridimensional, en la cual se observa la reconstrucción que se realiza con la ubicación de la población de las moscas, el objeto es reconstruido por las moscas que se encuentran dentro del recuadro.

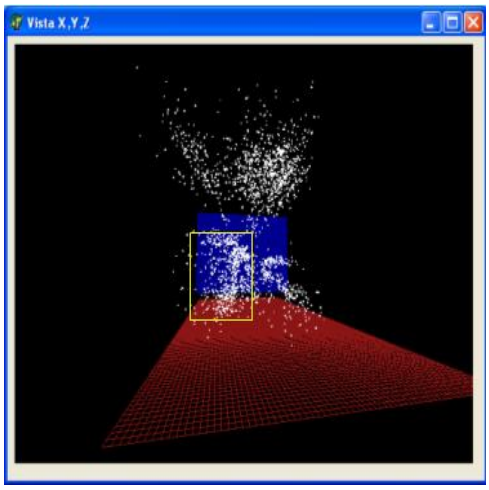

Figura 6.27: Vista *XYZ*.

6.3.2. Desplazamiento sobre el eje Z

En la siguiente situación el robot fue desplazado sobre el eje Z siguiendo el diagrama de la figura 3.2. Iniciando a 1.35 m y acercandose hasta 75 cm de distancia del objeto. En la figura 6.28 se muestra el estado inicial y el estado final del robot.

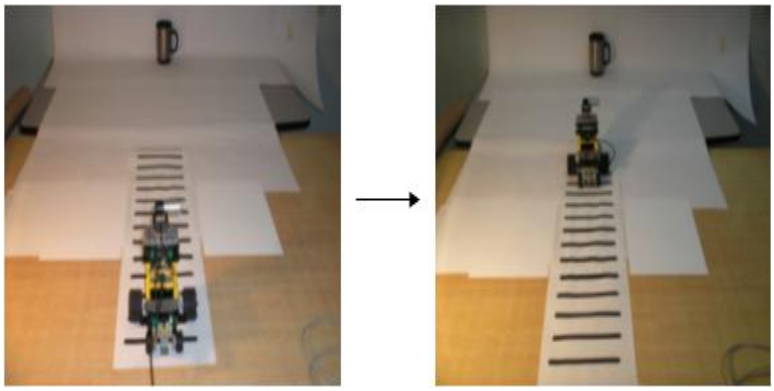

Figura 6.28: Desplazamiento del robot sobre el eje Z.

En la figura 6.29 se muestra la ubicación inicial de las moscas proyectadas en la imagen izquierda e imagen derecha con los parámetros de la tabla 6.1, utilizando 3000 moscas y cuatro regiones para obtener una mejor distribución.

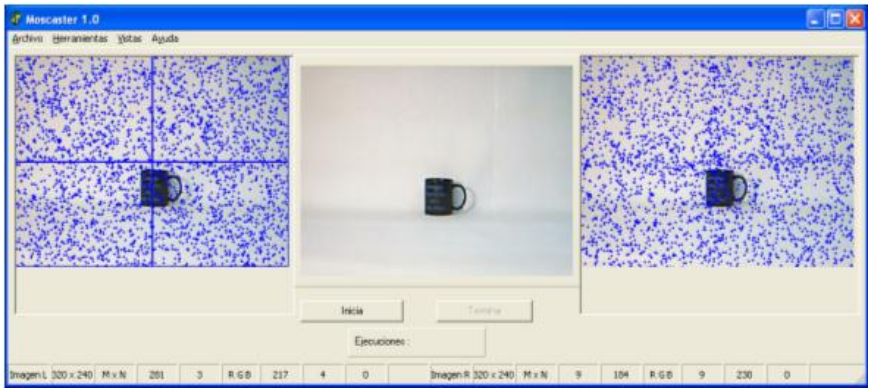

Figura 6.29: Inicio de la prueba sobre el eje Z.

En la figura 6.30 se presenta el resultado de la ejecución del modelo una vez que el robot ha finalizado su recorrido sobre el eje Z. Observamos que las moscas aptas estan presentes solo en el 50 % de las regiones. Además se observa la concentración de moscas sobre el objeto.

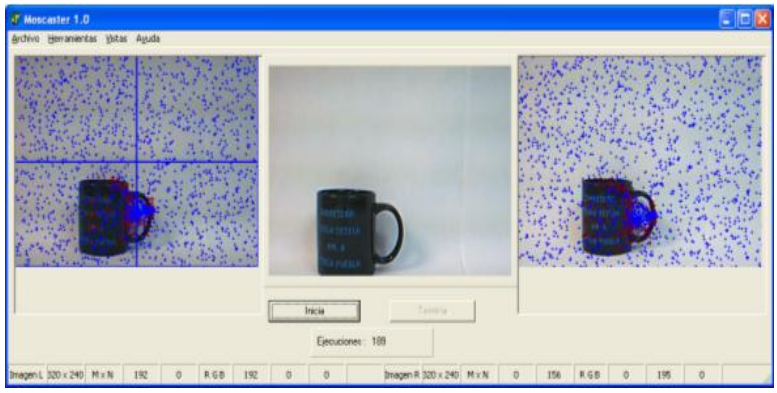

Figura 6.30: Resultado del desplazamiento sobre el eje Z.

6.3.3. Desplazamiento horizontal y sobre el eje Z

En la siguiente situación el robot fue desplazado horizontalmente y sobre el eje Z, siguiendo el diagrama de la figura 3.2. En la figura 6.31 se muestra el estado inicial y el estado final del robot.

En la figura 6.32 se muestra la ubicación inicial de las moscas proyectadas en la imagen izquierda e imagen derecha con los parámetros de la tabla 6.1, utilizando 3000 moscas. Para esta prueba fue conveniente utilizar dos regiones.

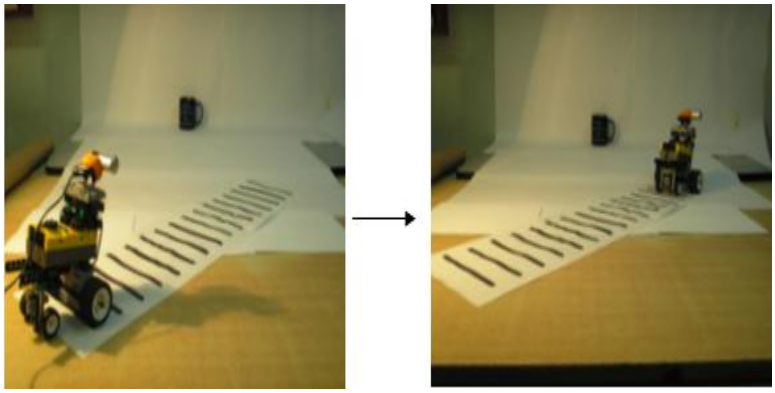

Figura 6.31: Desplazamiento del robot horizontalmente y sobre el eje *Z*.

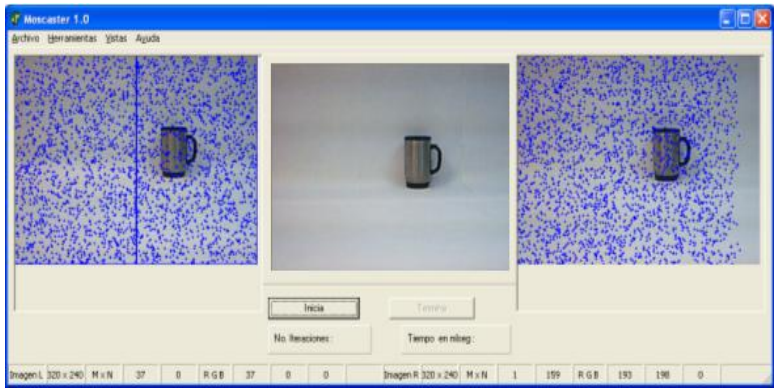

Figura 6.32: Inicio de la prueba sobre el eje horizontal y sobre el eje *Z*.

En la figura 6.33 se presenta el resultado de la ejecución del modelo una vez que el robot ha finalizado su recorrido sobre el eje horizontal y sobre el eje de la Z. Observamos que las moscas aptas estan presentes solo en el 50 % de las regiones. Además se observa la concentración de moscas sobre el objeto.

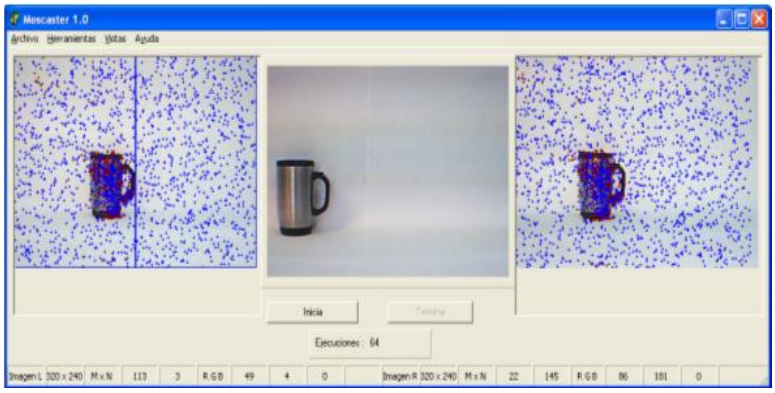

Figura 6.33: Resultado del desplazamiento sobre el eje horizontal y sobre el eje Z.

La reconstrucción del objeto se muestra en la vista XY, en la figura 6.34 (a) sin ninguna operación de mejora de la visualización. En 6.34 (b)se observa después de aplicar una operación de apertura y una de cerradura.

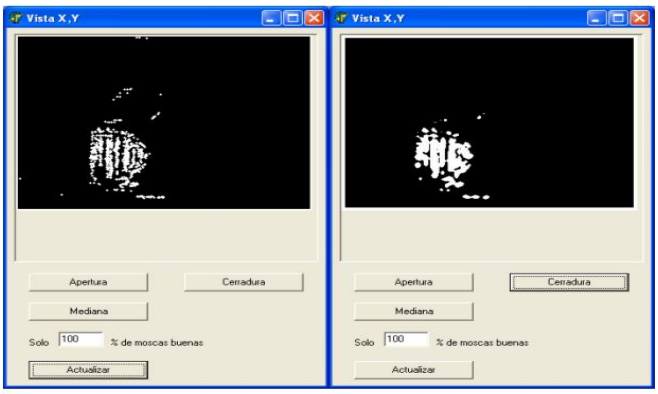

Figura 6.34: (a) Vista XY, (b) Después de aplicar una operación de apertura y una de cerradura.

En la figura 6.35 se muestra la vista tridimensional, en la cual se observa la reconstrucción que se realiza con la ubicación de la población de las moscas, el objeto es reconstruido por las moscas que se encuentran dentro del recuadro.

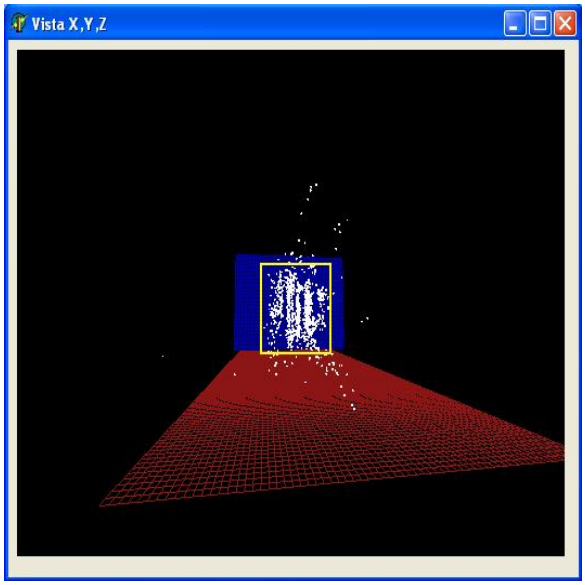

Figura 6.35: Vista *XYZ*.

Capítulo 7

Conclusiones y trabajo a futuro

Conclusiones

Se ha presentado un sistema para la reconstrucción *3D* parcial basado en el algoritmo de las moscas. Se empleó visión monocular en lugar de la visión estéreo ya que hemos usado solo una cámara. La cámara fue desplazada en diferentes distancias conocidas y controladas.

Se realizaron dos pruebas para la selección de los parámetros y el número de regiones que mejor realizara la reconstrucción, las cuales fueron:

- Procesando un par de imágenes.
- Procesando una secuencia de imágenes.

En estas pruebas se aplicaron tres grupos de parámetros, utilizando dos y cuatro regiones con poblaciones de 500, 1500, 3000 y 5000 moscas. Con relación a los resultados obtenidos podemos establecer lo siguiente:

- El grupo de parámetros del cual se obtuvo una mejor reconstrucción fueron los de la tabla 6.1, utilizando el 40 % de moscas aptas, el 10 % de moscas regeneradas por cruce, el 40 % de moscas regeneradas por mutación y el 10 % de moscas nuevas.

- El número de regiones en donde se obtuvo una mejor distribución de las moscas fueron cuatro, ya que evitó una concentración de ellas en ciertas posiciones innecesarias.

- Las poblaciones que mejor realizaron la reconstrucción del objeto fueron 3000 y 5000 moscas, no obstante el utilizar una población de moscas grande se invierte más tiempo.

Utilizando los parámetros de la tabla 6.1 y utilizando cuatro regiones se realizó la prueba de Procesamiento durante la navegación del robot, obteniendo la reconstrucción del objeto.

En la visualización de la reconstrucción se incluyeron las operaciones morfológicas de apertura y cerradura para apreciar mejor dicha reconstrucción en las vistas *XY*, *XZ* y *YZ*.

Además se presentó la vista *XYZ*, para tener una mejor apreciación de la ubicación de las moscas en el espacio *3D*.

Finalmente, este sistema es considerado para ser usado en robótica móvil con visión monocular para la tarea de navegación.

Trabajo a futuro

Fusionar en el modelo el control del robot con la captura de imágenes a través de una arquitectura adecuada que permita con la reconstrucción, la navegación del robot en una trayectoria segura.

Extender la aplicación a reconocimiento de objetos en la escena a través de características como color, forma y texturas.

Analizar otros algoritmos como el de las hormigas o abejas y realizar una comparación con el algoritmo de las moscas.

Ampliar las posibilidades en el sistema para que pueda efectuar la reconstrucción con base en el análisis de frecuencias a través de Tranformadas de Fourier o Transformadas de Wavelet junto con el algoritmo de las moscas.

Bibliografía

[1] Boumaza A. M. and Louchet J. Dynamic flies: Using real-time parisian evolution in robotics. *EvoWorkshops*, pages 288–297, 2001.

[2] Boumaza A. M. and Louchet J. Mobile robot sensor fusion using flies. *EvoWorkshops, LNCS*, 2611:357–367, 2003.

[3] Boumaza A. M. and Louchet J. Robot perception using flies. *Proceedings of IEEE-SETIT, Sousse Tunisia*, Marzo 2003.

[4] Mery D. Visión por computador, apuntes, departamento de ciencia de la computación. *Universidad Católica de Chile.*, 2005.

[5] Kursawe F. A variant of evolution strategies for vector optimization proceedings of the tenth. *International Conference on Multiple Criteria Decision Making*, pages 187–193, 1992.

[6] Pajares G. and M. de la Cruz J. *Visión por Computador, Imágenes digitales y aplicaciones*. Alfaomega, Ra-Ma, 2002.

[7] Seiji H., Yoko S., Syoji K., and Kenichi K. An application of genetic algorithm to extract the shape of transparent objects. *IPSJ SIGNotes Computer Vision*, 097, 1995.

[8] Louchet J. Stereo analysis using individual evolution strategy. *International conference on pattern recognition. Barcelona.*, Septiembre 2000.

[9] Louchet J. Using an individual evolution strategy for stereovision. *Genetic Programming and Evolvable Machines 2(2)*, pages 101–109, Enero 2001.

[10] Louchet J., Guyon M., Lesot M., and Boumaza A. L'algorithme des mouches dynamiques guider un robot par évolution artificielle en temps réel. *Apprentissage et évolution*, pages 115–130, Enero 2001.

[11] Louchet J., Guyon M., Lesot M., and Boumaza A. Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. *Pattern Recognition Letters 23*, pages 335–345, Enero 2002.

[12] González J. J. *Visión por Computador.* Paraninfo., 1999.

[13] Sáez J. M., Penalver A., and Escolano F. Estimación de las acciones de un robot utilizando visión estéreo. *Departamento de Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante.*

[14] Koza J. R. Proceedings of the first international conference on genetic programming. *Stanford, MIT Press.*, Julio 1996.

[15] Topiltzin L. Visión estéreo mediante estrategias evolutivas. Febrero 2005.

[16] Pauplin O., Louchet J., Lutton E., and Parent M. Obstacle detection by evolutionary algorithm: the fly algorithm. *2nd International Conference on Autonomous Robots and Agents*, Diciembre 2004.

[17] Barrera P. and Canas J. M. Seguimiento tridimensional usando dos cámaras. *Reports on Systems and Communications*, IV. Number 7, Diciembre 2004.

[18] Ji Q. and Zhang. Camera calibration with genetic algorithms. *IEEE Transactions on systems man and cybernetics-Part A systems and humans*, 31, 2001.

[19] Klette R., Schluns K., and Koshan A. *Computer Vision Three-Dimensional Data from Images*. Springer., 1998.

[20] Molina R. Del procesamiento a la visión artificial. *Depto. Ciencias de la Computación e I. A. Universidad de Granada.*

[21] Poli R. Genetic programming for image analisis. *Proceedings of the First International Conference on Genetic Programming,Stanford, MIT Press*, pages 363–368, Julio 1996.

[22] González R. C. and Woods R. E. *Tratamiento digital de imágenes.* Addison-Wesley/Díaz de Santos., 1996.

[23] Zhang Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 22(11):1330–1334, 2000.