UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

CARRERA DE INGENIERÍA MECATRÓNICA

SEGUIMIENTO DE MULTIPLES OBJETOS EN MOVIMIENTO PARA LA NAVEGACIÓN EN ROBÓTICA MÓVIL

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA MECATRÓNICA

SEBASTIAN JAVIER BECERRA CAMACHO

DIRECTOR: ING. GUILLERMO MOSQUERA MSC.

Quito, Octubre 2017

DECLARACIÓN

Yo, **Becerra Camacho Sebastián Javier,** declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

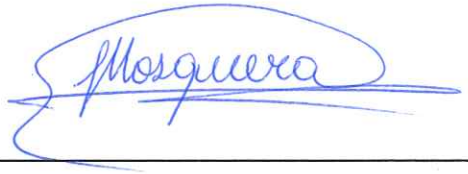
La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Sebastián Javier Becerra Camacho

C.I. 1719654038

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título **"Seguimiento de múltiples objetos en movimiento para Navegación en robótica móvil"**, que, para aspirar al título de **Ingeniero Mecatrónica** fue desarrollado por **Sebastián Javier Becerra Camacho,** bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 19,27 Y 28.



Ing. Guillermo Mosquera MsC.

DIRECTOR DEL TRABAJO

C.I. 0802613059

DEDICATORIA

A mis padres, por el cariño y el amor que me brindan cada momento de mi vida; por su apoyo incondicional, por siempre creer en mí, y por ser mi motivación para alcanzar mis metas y triunfos.

AGRADECIMIENTO

A Dios, por haberme dado salud y vida para poder culminar mis estudios universitarios.

A mi padre, por brindarme los recursos económicos necesarios para mi formación académica, por su confianza y apoyo durante el curso de mi vida universitaria

A mi madre, por apoyarme y darme sus sabios consejos cuando más los necesitaba y por el amor incondicional.

A la Universidad Tecnológica Equinoccial y a sus docentes, por la formación académica que recibí en sus aulas durante estos 5 años. Agradezco de manera especial al Ing. Guillermo, por la dirección de este proyecto; por haber dedicado su tiempo y conocimiento, experiencia e investigación en el proceso de desarrollo del presente trabajo.

Finalmente agradezco a mis amigos y amigas de la Universidad, que supieron ayudarme en su respectivo momento. Gracias por su amistad.

FORMULARIO DE REGISTRO BIBLIOGRÁFICO
PROYECTO DE TITULACIÓN

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	171965403-8
APELLIDO Y NOMBRES:	Becerra Camacho Sebastián Javier
DIRECCIÓN:	Loja Oe9-306 y Garcia Moreno
EMAIL:	trivuler@gmail.com
TELÉFONO FIJO:	2825-382
TELÉFONO MOVIL:	0998965569

DATOS DE LA OBRA	
TITULO:	Seguimiento de múltiples objetos en movimiento para Navegación en robótica móvil
AUTOR O AUTORES:	Sebastián Javier Becerra Camacho
FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN:	2017/10/12
DIRECTOR DEL PROYECTO DE TITULACIÓN:	Ing. Guillermo Mosquera MsC.
PROGRAMA	PREGRADO ■ POSGRADO ☐
TITULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
RESUMEN:	En el presente proyecto se realizó el estudio de un sistema de reconocimiento de objetos, basado en el análisis en tiempo real de imágenes adquiridas por medio de una cámara digital y su procesamiento mediante un computador. El análisis se desarrolló con el uso de algoritmos especiales para la generación de trayectorias, conocidos como RRT's (Rapidly-exploring random tree). Para el presente proyecto se implementó un sistema que unifique los algoritmos RRT's con visión artificial como medio de adquisición de datos y así permitir al sistema móvil la toma de decisiones respecto de su trayectoria. Para la adquisición de imágenes se utilizó una cámara web y el código implementado para este sistema se desarrolló en la plataforma de Matlab. También se aplicó la metodología mecatrónica como guía para el desarrollo del sistema. Se obtuvo como resultado un sistema capaz de observar y reconocer elementos específicos de las imágenes procesadas de tal manera que el sistema móvil pueda tomar decisiones sobre su trayectoria en tiempo real basado en las observaciones realizadas. El sistema desarrollado se centra en el reconocimiento de patrones geométricos como cuadrados, triángulos, rombos, trapecios, etc. Estos objetos serán identificados como obstáculos que el algoritmo RRT debe esquivar para crear la trayectoria del sistema móvil.

PALABRAS CLAVES:	Algoritmo, Robótica, Mecatrónica, Visión artificial, Matlab.
ABSTRACT:	The present project was performed the study of an object recognition system, based on the real time analysis of images acquired through a digital camera and its processing by means of a computer. The analysis was developed with the use of special algorithms for the generation of trajectories, known as RRT (Rapidly-exploring random tree). For the current project is implemented a system that unifies RRT algorithms with artificial vision as a means of data acquisition and thus allow the mobile system to make decisions regarding its tray. For the acquisition of images a webcam is used and the code implemented for this system was developed on the Matlab platform. The mechatronic methodology for the development of the system is also applied. The result was a system capable of observing and recognizing specific elements of the processed images in such a way that the mobile system can make decisions on real time impersonations based on the observations made. The developed system focuses on the recognition of geometric patterns such as squares, triangles, diamonds, trapezoids, etc. These objects will be identified as obstacles that the RRT algorithm must dodge to create the trajectory of the mobile system.
KEYWORDS	Algorithm, Robotics, Mechatronics, Artificial vision, Matlab.

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio Digital de la Institución.

f:_____

BECERRA CAMACHO SEBASTIÁN JAVIER

CI 171965403-8

DECLARACIÓN Y AUTORIZACIÓN

Yo, **Sebastián Javier Becerra Camacho,** CI 171965403-8 autor del proyecto titulado: "**Seguimiento de múltiples objetos en movimiento para Navegación en robótica móvil**" previo a la obtención del título de **Ingeniero en Mecatrónica** en la Universidad Tecnológica Equinoccial.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Autorizo a la BIBLIOTECA de la Universidad Tecnológica Equinoccial a tener una copia del referido trabajo de graduación con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, 05 de Octubre de 2017

f:_____

BECERRA CAMACHO SEBASTIÁN JAVIER

CI 171965403-8

ÍNDICE DE CONTENIDOS

PÁGINA

RESUMEN v
ABSTRACT vi
1. INTRODUCCIÓN 4
1.1. ALGORITMO RRT BÁSICO 4
1.2. ALGORITMO RRT-CONNECT 6
1.2.1. Desarrollo de trayectorias en entornos 2D y 3D 7
2. METODOLOGÍA Y DISEÑO 10
2.1. REQUERIMIENTOS DEL SISTEMA 10
2.2. DISEÑO DEL SISTEMA DE CONTROL 11
2.3. DIAGRAMA DE FLUJO DEL SISTEMA 12
PROCESO 13
2.5. IMPLEMENTACIÓN DEL ALGORITMO RRT 15
2.6. INTERFAZ DE USUARIO 19
2.7. UNIFICACIÓN DE LOS SISTEMAS DE VISIÓN ARTIFICIAL GENERACIÓN DE TRAYECTORIAS RRT E INTERFAZ DE USUARIO 21
3. ANÁLISIS DE RESULTADOS 21
4. CONCLUSIONES Y RECOMENDACIONES 27
BIBLIOGRAFÍA 29

ÍNDICE DE FIGURAS

PÁGINA

Figura 1. Esquema general de tratamiento de imágenes..... 1
Figura 2. Esquema de fases del reconocimiento de imágenes..... 2
Figura 3. Construcción del RRT básico..... 5
Figura 4. Función EXTENDER del algoritmo RRT básico..... 5
Figura 5. Representación gráfica de la operación EXTENDER del algoritmo básico RRT 5
Figura 6. Algoritmo RRT-Connect..... 6
Figura 7. Representación de la conexión entre arboles realizada con RRT-Connect. (a) Creación de un punto aleatorio q y su nodo más cercano $qnear$ del árbol Ta. (b) Creación de $qnew$ hacia q. (c) y (d) Representación de la función CONECTAR. 6
Figura 8. Funcionamiento del RRT-Connect en un entorno bidimensional con tres obstáculos poligonales. (a) Calculo de la trayectoria con un ε o ancho de búsqueda corto. (b) Calculo de la trayectoria con un ε o ancho de búsqueda largo. (c) y (d) Trayectorias resultantes con pos procesamiento..... 7
Figura 9. Funcionamiento del RRT-Connect en un entorno tridimensional con tres obstáculos poligonales. (a) Trayectoria resultante al aplicar RRT-Connect. (b) Trayectoria resultante con pos-procesamiento. 8
Figura 10. Representación de la operación de pos-procesado en un espacio de configuración bidimensional (a). (b) muestra la ruta calculada por RRT-Connect y (h) la ruta final 1-3-4-8..... 8
Figura 11. Diagrama para el cálculo de trayectoria libre de colisiones para implementación..... 9
Figura 12. Metodología de diseño mecatrónico modelo en V 10
Figura 13. Diagrama de flujo del sistema..... 13
Figura 14. Imagen transformada a escala de grises 14
Figura 15. Suma lógica entre la imagen de fondo y los objetos colocados. 15
Figura 16. Objetos identificados en la imagen de fondo 15
Figura 17. Líneas de código correspondiente a la inicialización de datos... 15
Figura 18. Comparación entre el número de nodos y las interacciones creadas..... 16
Figura 19. Comando FOR para la generación de nodos..... 17
Figura 20. Bucle FOR para determinar la posición de los nodos 17
Figura 21. Condicionante IF para cálculo de colisiones..... 17
Figura 22. Bucle FOR para determinar el camino más corto de llegada..... 18
Figura 23. Comprobación de nodos..... 18
Figura 24. Generación del diagrama de trayectorias 19
Figura 25. Plano de trayectoria generado en Matlab 19
Figura 26. Pantalla de inicio de la interfaz de usuario..... 20

Figura 27. Pantalla de selección de fondo 20
Figura 28. Pantalla de selección de puntos de partida y llegada 20
Figura 29. Selección de puntos partida y llegada..... 21
Figura 30. Cálculo de trayectorias con visión artificial y algoritmo RRT..... 21
Figura 31. Imagen de la suma del fondo, los obstáculos y la trayectoria 22
Figura 32. Imagen de fondo detectada por cámara 25
Figura 33. Selección del punto de partida y llegada para el cálculo de trayectoria 25
Figura 34. Trayectoria final calculada con dos obstáculos..... 25
Figura 35 Trayectoria final calculada con cuatro obstáculos..... 26

ÍNDICE DE TABLAS

PÁGINA

Tabla 1. Alternativas de software para el desarrollo del sistema..... 11
Tabla 2. Tabla de pruebas con FPS para el sistema de visión artificial..... 23
Tabla 3. Tabla de pruebas de número de nodos para el algoritmo RRT 24

RESUMEN

En el presente proyecto se realizó el estudio de un sistema de reconocimiento de objetos, basado en el análisis en tiempo real de imágenes adquiridas por medio de una cámara digital y su procesamiento mediante un computador. El análisis se desarrolló con el uso de algoritmos especiales para la generación de trayectorias, conocidos como RRT's (Rapidly-exploring random tree). Para el presente proyecto se implementó un sistema que unifique los algoritmos RRT's con visión artificial como medio de adquisición de datos y así permitir al sistema móvil la toma de decisiones respecto de su trayectoria. Para la adquisición de imágenes se utilizó una cámara web y el código implementado se desarrolló en la plataforma de Matlab. También se aplicó la metodología mecatrónica como guía para el desarrollo del sistema. Se obtuvo como resultado un sistema capaz de observar y reconocer elementos específicos de las imágenes procesadas de tal manera que el sistema móvil pueda tomar decisiones sobre su trayectoria en tiempo real basado en las observaciones realizadas. El sistema desarrollado se centra en el reconocimiento de patrones geométricos como cuadrados, triángulos, rombos, trapecios, etc. Estos objetos serán identificados como obstáculos que el algoritmo RRT debe esquivar para crear la trayectoria del sistema móvil.

Palabras clave: Algoritmo, Robótica, Mecatrónica, Visión artificial, Matlab.

ABSTRACT

The present project was performed the study of an object recognition system, based on the real time analysis of images acquired through a digital camera and its processing by means of a computer. The analysis was developed with the use of special algorithms for the generation of trajectories, known as RRT (Rapidly-exploring random tree). For the current project is implemented a system that unifies RRT algorithms with artificial vision as a means of data acquisition and thus allow the mobile system to make decisions regarding its tray. For the acquisition of images a webcam is used and the code implemented for this system was developed on the Matlab platform. The mechatronic methodology for the development of the system is also applied. The result was a system capable of observing and recognizing specific elements of the processed images in such a way that the mobile system can make decisions on real time impersonations based on the observations made. The developed system focuses on the recognition of geometric patterns such as squares, triangles, diamonds, trapezoids, etc. These objects will be identified as obstacles that the RRT algorithm must dodge to create the trajectory of the mobile system.

Keywords: Algorithm, Robotics, Mechatronics, Artificial vision, Matlab.

1. INTRODUCCIÓN

La detección y reconocimiento de objetos es una valiosa herramienta para el desarrollo de nuevas aplicaciones e implementación de nuevas tecnologías, que se puede aplicar a diversas áreas, tales como la medicina, en la industria alimenticia, en empresas de seguridad, etc. Actualmente se están desarrollando diversos proyectos en todo el mundo, enfocados en el campo en la robótica móvil, donde el reconocimiento de objetos o distintas señales, que están dentro de un sistema móvil, permiten definir la trayectoria de un robot por distintas zonas, permitiendo movimientos más apropiados y con un óptimo desempeño, agilitando las operaciones que este realice y aumentando la productividad en el trabajo que este desempeñe. Todo esto se realiza a través de un software de computadora, basado en algoritmos de programación, que permiten entrenar al sistema, para que el actué según el requerimiento (Figura 1).

En la actualidad se puede encontrar una gran cantidad de robots móviles, pero su acceso es limitado, debido a la falta de aplicación de esta tecnología en el área de investigación dentro del Ecuador. Lo cual crea una alta demanda por la creación de nuevos prototipos relacionados con el tema, para su exploración y aplicación en la industria, ofreciendo una gama nuevos productos y aplicaciones dentro del mercado local.

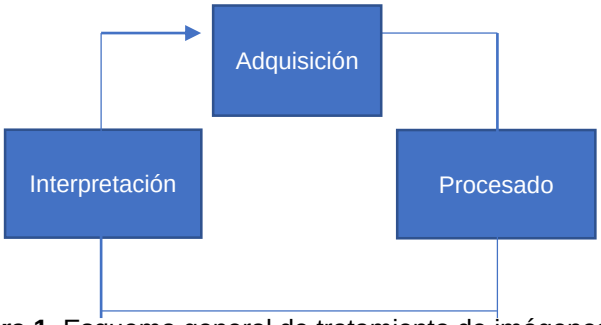


Figura 1. Esquema general de tratamiento de imágenes
(Domingo, 2004)

En este proyecto se busca dar una descripción detallada de un sistema reconocedor de objetos en tiempo real. Teniendo como objetivo principal de este el desarrollo de una aplicación capaz de diferenciar y reconocer objetos en movimiento. Los objetos estudiados se caracterizan por ser formas cerradas de baja complejidad. La aplicación desarrollada se puede dividir en tres etapas cíclicas: La adquisición, el procesado, y la interpretación.

La comunicación entre el usuario y el dispositivo autónomo, es controlada mediante una interfaz de usuario y una cámara web, la cual recogerá información en tiempo real del entorno, y ajustará las actividades del dispositivo. En la Figura 2 se puede observar las fases de reconocimientos de imágenes

En la primera fase del sistema se realiza la adquisición de la imagen por medio de un dispositivo óptico de video, para posteriormente ser digitalizado y

enviado a un ordenador donde será procesada la información mediante el uso de un logaritmo. En específico, para este proyecto, se ha utilizado una web cam con una resolución de 5 megapíxeles.

La segunda fase consiste en la aplicación del logaritmo diseñado para el procesado de los objetos, el algoritmo se programará en un entorno de Matlab versión R2016A. Se realizará un programa de segmentación capaz de diferenciar el fondo y los objetos y otro de extracción de características capaz de encontrar en la imagen todos aquellos elementos que pueden ser de interés para el reconocimiento de objetos contenidos en la imagen.

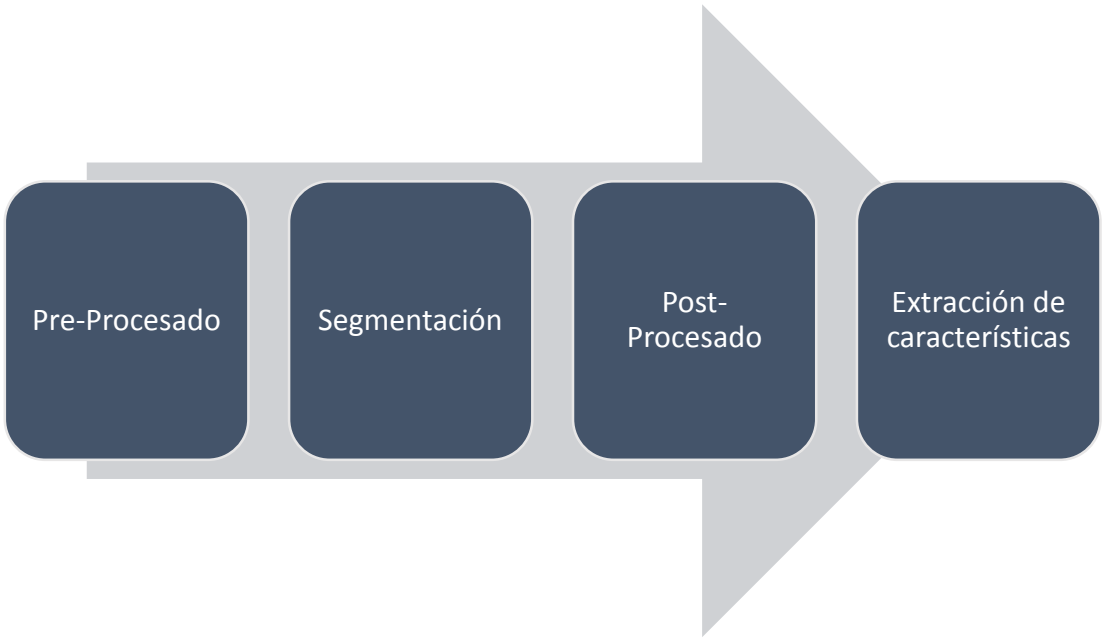


Figura 2. Esquema de fases del reconocimiento de imágenes
(Ana González Marcos, 2006)

La tercera fase descrita como la interpretación se basa en la comparación de los objetos encontrados en la imagen con aquellos objetos existentes en una base de datos que se actualizará conforme se procesan las imágenes. El seguimiento de estos objetos ocurrirá únicamente en planos paralelos al de la imagen, es decir no se estudiarán distorsiones visuales debidos a la perspectiva.

Tecnológicamente en el país se busca fomentar e impulsar la robótica móvil, debido a la cantidad de aplicaciones interesantes que se puede implementar en la industria con este tipo de tecnología además de ofrecer nuevos ámbitos de investigación a los profesionales de diferentes áreas. Actualmente diversas actividades realizadas dentro de la industria están relacionadas con el uso de máquinas programadas para realizar tareas específicas pero que necesitan de tiempo y cuidado del personal para que puedan realizar las tareas adecuadamente, en este punto es donde interviene la inteligencia artificial y

se desea crear sistemas de control de autoaprendizaje donde la maquina logre mejorar continuamente los procesos. El presente trabajo centro su investigación en el aprendizaje de ciertos patrones, que permitan a la máquina la toma de decisiones en el transcurso del trabajo, mejorando la eficiencia de la máquina y evitando cualquier riesgo o peligro en el proceso de producción para el operario de una empresa, fabrica o cualquier tipo de industria, en la que se necesite la cooperación de un sistema robótico.

La visión artificial ha tenido un avance tecnológico en las últimas décadas en el campo de la ingeniería mecatrónica. La tecnología, y en especial los ordenadores, están cada vez más integrados en nuestra vida cotidiana. La mayoría de los dispositivos móviles poseen hoy en día más capacidad de computo que cualquier ordenador de hace una década.

Sin duda alguna, uno de los campos interesantes es la robótica y su estrecha relación con la visión artificial. Ya no hace falta dominar un lenguaje computacional para interactuar con los sistemas inteligentes. Los sistemas de inteligencia artificial ya son capaces de entender y procesar órdenes empleando el lenguaje natural. Se acostumbra ver equipos inteligentes completamente autónomos, en entornos estrictamente industriales o por lo menos altamente controlados como fábricas de coches. La aplicación de estos sistemas en entornos más dinámicos como hogares u oficinas, necesita de una enorme cantidad de sensores que permita al sistema inteligente percibir el entorno con toda su complejidad para interactuar con él.

Igual que en los humanos, la entrada de datos más importante es la visión. En un sistema inteligente, normalmente se trata de una cámara compuesta por multitud de lentes y sensores específicos para cada sistema y campo de aplicación. Una vez realizados todos los ajustes mecánicos, la visión artificial permite al sistema detectar la mayoría de los objetos para decidir cómo reaccionar y actuar frente a ellos.

La visión artificial engloba a cualquier proceso óptico mediante el cual un sistema inteligente es capaz de extraer información de un entorno para su interpretación mediante el uso de la computadora. En un principio los sistemas de visión artificial estaban estrechamente basados en la visión humana, pero debido a la falta de entendimiento de los procesos que tienen lugar en el cerebro a la hora de interpretar los datos del sistema visual, estos sistemas artificiales resultaron básicamente imprácticos. El reconocimiento de objetos mediante webcam en tiempo real, en combinación con el sistema visual del que dispone el cuerpo humano, permite un reconocimiento y una interpretación de su entorno mucho más flexible y adaptable a cambios que cualquier sistema de visión artificial. Sin embargo, los sistemas de visión artificial son capaces de procesar cantidades de datos mucho mayores y en menos tiempo si se trata de tareas repetitivas. La precisión matemática, permite un estudio y análisis mucho más detallado y extenso por parte de un

sistema basado en la visión artificial. Esta característica permite la creación de sistemas muy diferentes en cuanto a componentes y capacidades, según la tarea a implementar.

El presente trabajo se centra en el sistema de control debido a que la implementación del mismo se realizará en un modelo virtual o de simulación.

Para el desarrollo se propone el siguiente objetivo general: Diseñar un sistema de seguimiento de múltiples objetos en movimiento para navegación en robótica móvil.

Para el cumplimiento de dicho objetivo se plantearon los siguientes objetivos específicos:

- Diseñar un sistema de detección de movimiento de objetos
- Estudiar la captura en tiempo real mediante el Image acquisition toolbox de MatLab
- Diseñar una interfaz gráfica para la manipulación de la aplicación
- Implementar un algoritmo RRT para la generación de trayectorias

1.1. ALGORITMO RRT BÁSICO

El RRT (Figura 3) es un algoritmo capaz de explorar eficientemente el espacio de configuración desde un punto de partida q_{start} a un punto de llegada q_{goal}. El RRT básico se basa en la construcción de un árbol *T* compuesto por nodos y enlaces, que se incrementan aleatoriamente desde un punto de origen. En este algoritmo se selecciona una configuración aleatoria q_{rand} que es la base para crecer el árbol hacia esta configuración, para eso se usa una función conocida como EXTENDER.

La función EXTENDER (Figura 4 y Figura 5) comienza con la selección de q_{near} , que es el nodo de *T* más próximo a *q*, con la función VECINOMASCERCANO. Seguidamente, la función NUEVOESTADO calcula una nueva configuración q_{new}, como un salto de magnitud ε en dirección a *q* desde q_{near}. Adicionalmente, NUEVOESTADO debe validar el enlace entre q_{near} y q_{new} ; si no existe colisión, se agrega al árbol *T* tanto q_{new} como el enlace, usando AGREGARNODO y AGREGARENLACE. Si la configuración aleatoria *q* se encuentra dentro de un círculo de centro q_{near} y radio ε, entonces se considera que se ha alcanzado a *q*. Si por el contrario, no se ha producido el alcance, entonces se devolverá el valor extendido. Por último, en caso de que la función NUEVOESTADO haya advertido la existencia de algún obstáculo en el camino de q_{near} a q_{new}, se informa de que no ha habido nuevas ramas y el nuevo nodo no es agregado al árbol *T*.

La naturaleza del RRT le hace avanzar con más rapidez en aquellas zonas donde haya un mayor espacio libre, pues es allí donde hay más posibilidades de encontrar puntos factibles. También se ha demostrado la capacidad del

RRT para explorar cualquier espacio de búsqueda de n-dimensiones de forma uniforme respecto a diferentes regiones.

Para encontrar una trayectoria libre de colisiones, el RRT inicia su exploración en como raíz del árbol T. Si el crecimiento de este árbol alcanza q_{goal} o se cumple un número límite de interacciones K, el algoritmo RRT culmina y luego se traza una ruta desde q_{start} a q_{goal}. En otras palabras, se puede establecer un camino continuo entre nodos empezando desde las ramas, pasando por el tronco, hasta llegar a la raíz. (López García , 2012)

BUILD_ RRT

1. $T(0) \leftarrow q_{start}$
2. **for** $i = 1$ **to** K **do**
3. $\quad q_{rand} \leftarrow$ PUNTOALEATORIO
4. $\quad$ EXTENDER (T, q_{rand})
5. devuelve T

Figura 3. Construcción del RRT básico
(López García , 2012)

EXTENDER

1. $q_{near} \leftarrow$ VECINOMASCERCANO (T, q)
2. **if** NUEVOESTADO (q, q_{near}, q_{new}) **then**
3. $\quad$ AGREGARNODO (T, q_{new})
4. $\quad$ AGREGARENLACE (T, q_{near}, q_{new})
5. $\quad$ **if** $q_{new} = q$ **then**
6. $\quad\quad$ devuelve *Alcanzado*
7. $\quad$ **else**
8. $\quad\quad$ devuelve *Extendido*
9. devuelve *Rechazado*

Figura 4. Función EXTENDER del algoritmo RRT básico.
(López García , 2012)

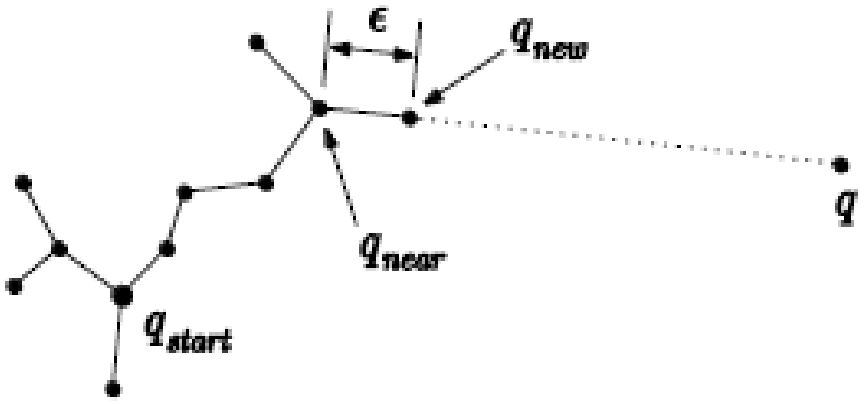


Figura 5. Representación gráfica de la operación EXTENDER del algoritmo básico RRT
(López García , 2012)

1.2. ALGORITMO RRT-CONNECT

En los últimos años se han producido diferentes mejoras al algoritmo RRT básico con resultados mejores y probados en práctica. El nuevo algoritmo ha demostrado ser una de las variantes más confiables para todo tipo de aplicaciones. En la Figura 6 y Figura 7 se observa la composición y representacion del algoritmo teniendo como base lo explicado anteriormente.

CONECTAR (T, q)

1. **repeat**
2. $S \leftarrow$ EXTENDER (T, q_{rand})
3. **until not** $(S = Extendido)$
4. devuelve S

RRT-Connect (q_{start}, q_{goal})

1. $T_a(0) \leftarrow q_{start}$, $T_b(0) \leftarrow q_{goal}$
2. **for** $i = 1$ **to** K **do**
3. $q_{rand} \leftarrow$ PUNTOALEATORIO
4. **if not** (EXTENDER $(T_a, q_{rand}) = Rechazado$) **then**
5. **if** (CONECTAR $(T_b, q_{new}) = Alcanzado$) **then**
6. devuelve RUTA (T_a, T_b)
7. INTERCAMBIAN (T_a, T_b)
8. devuelve *Fallido*

Figura 6. Algoritmo RRT-Connect
(López García , 2012)

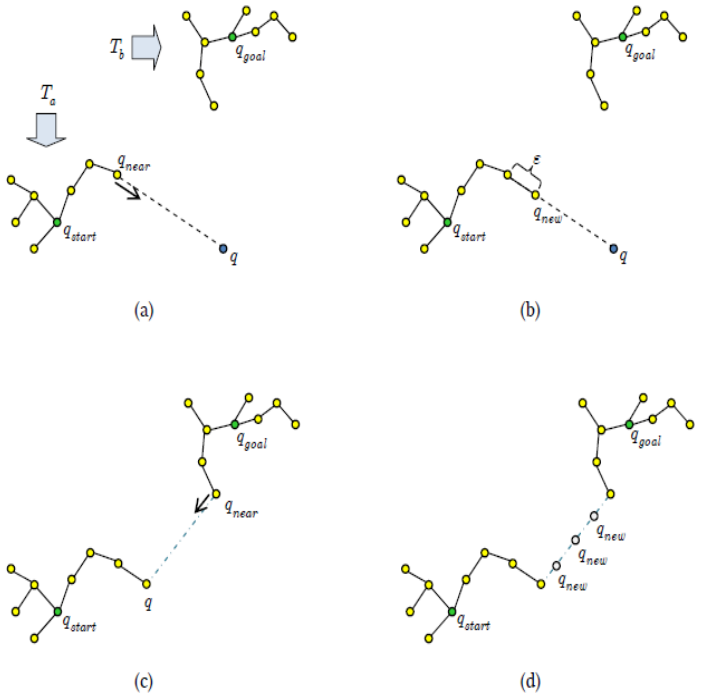


Figura 7. Representación de la conexión entre arboles realizada con RRT-Connect. (a) Creación de un punto aleatorio q y su nodo más cercano $qnear$ del árbol Ta. (b) Creación de $qnew$ hacia q. (c) y (d) Representación de la función CONECTAR.
(López García , 2012)

La idea principal detrás del RRT-Connect, es el desarrollo de dos árboles que inician en los nodos inicial y final denominados T_a y T_b. Ambos arboles crecen simultáneamente uno hacia el otro, permitiendo una mejora en la convergencia del algoritmo, así como en el tiempo de computo, debido al menor número de comprobaciones de colisión. (López García , 2012)

1.2.1. Desarrollo de trayectorias en entornos 2D y 3D

Para verificar el comportamiento del algoritmo RRT-Connect, se utiliza una simulación donde se puede observar a un robot que navega en un espacio representado por un plano. Los obstáculos son caracterizados por polígonos o poliedros sólidos en 2D o en 3D. Sin importar la forma del robot, se comprueban las colisiones que se realizan directamente entre el punto y uno de los polígonos, en el plano o espacio, mediante técnicas de geometría computacional. Estas simulaciones son realizadas mediante software especializado como es Matlab. En la Figura 8, Figura 9 y Figura 10 se puede ver algunos ejemplos.

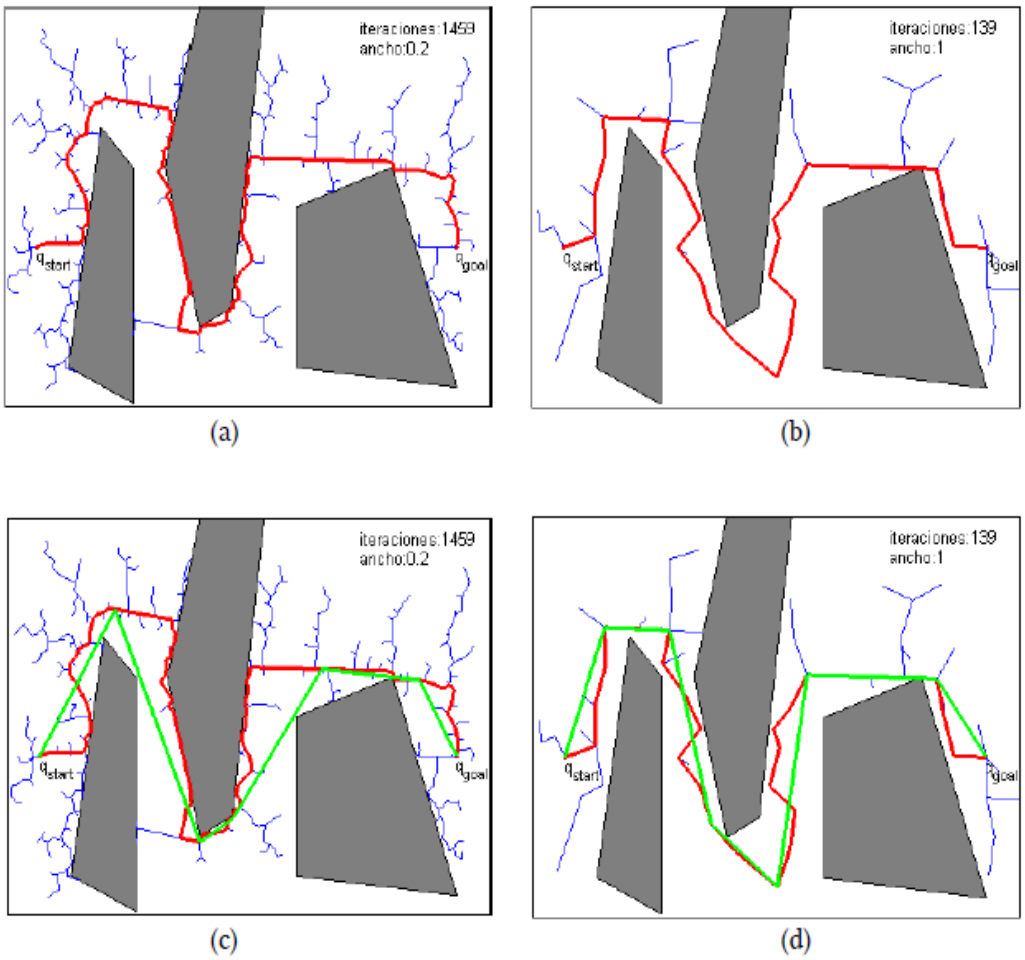


Figura 8. Funcionamiento del RRT-Connect en un entorno bidimensional con tres obstáculos poligonales. (a) Calculo de la trayectoria con un ε o ancho de búsqueda corto. (b) Calculo de la trayectoria con un ε o ancho de búsqueda largo. (c) y (d) Trayectorias resultantes con pos procesamiento
(López García , 2012)

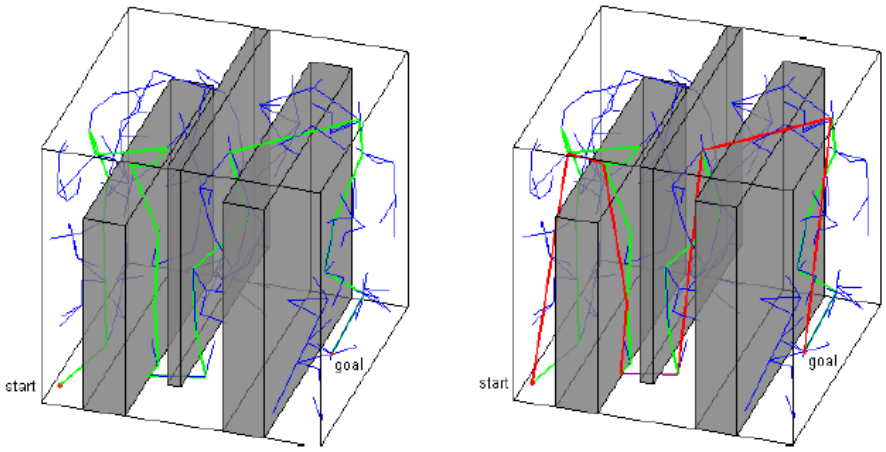


Figura 9. Funcionamiento del RRT-Connect en un entorno tridimensional con tres obstáculos poligonales. (a) Trayectoria resultante al aplicar RRT-Connect. (b) Trayectoria resultante con pos-procesamiento.
(López García , 2012)

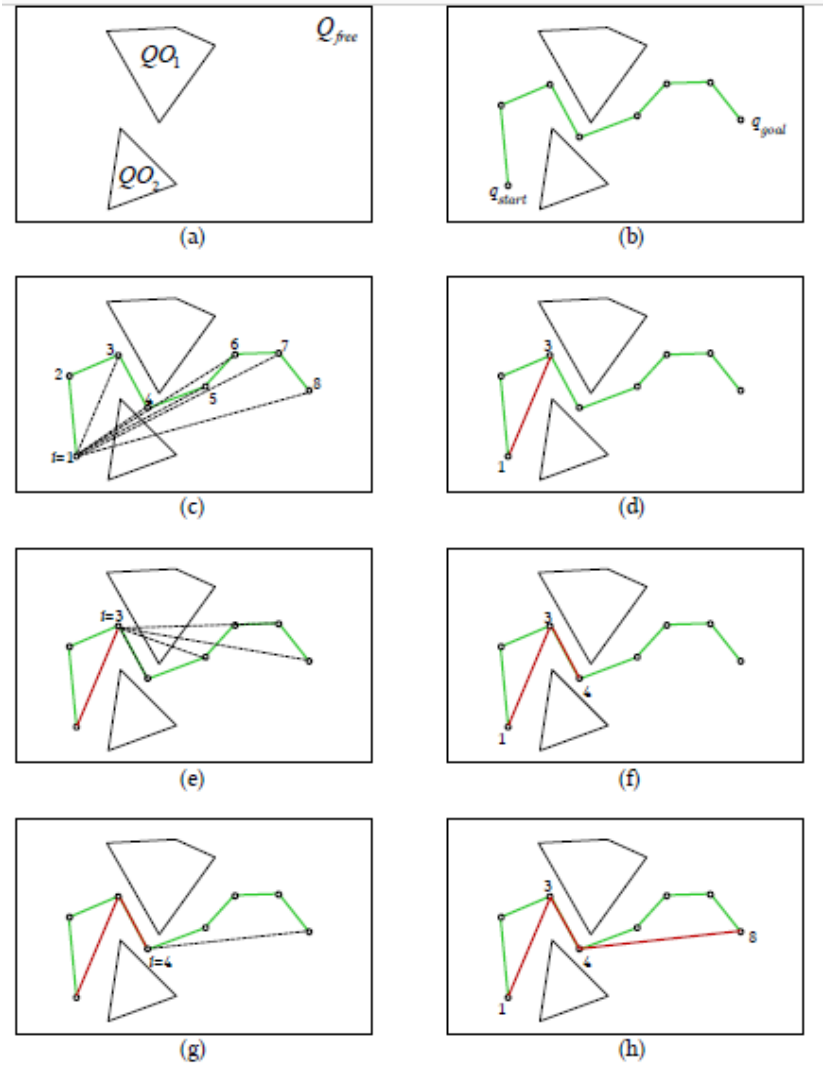


Figura 10. Representación de la operación de pos-procesado en un espacio de configuración bidimensional (a). (b) muestra la ruta calculada por RRT-Connect y (h) la ruta final 1-3-4-8.
(López García , 2012)

Para la construcción de estas simulaciones se tiene como referencia de operación el diagrama de flujo mostrado en la Figura 11.

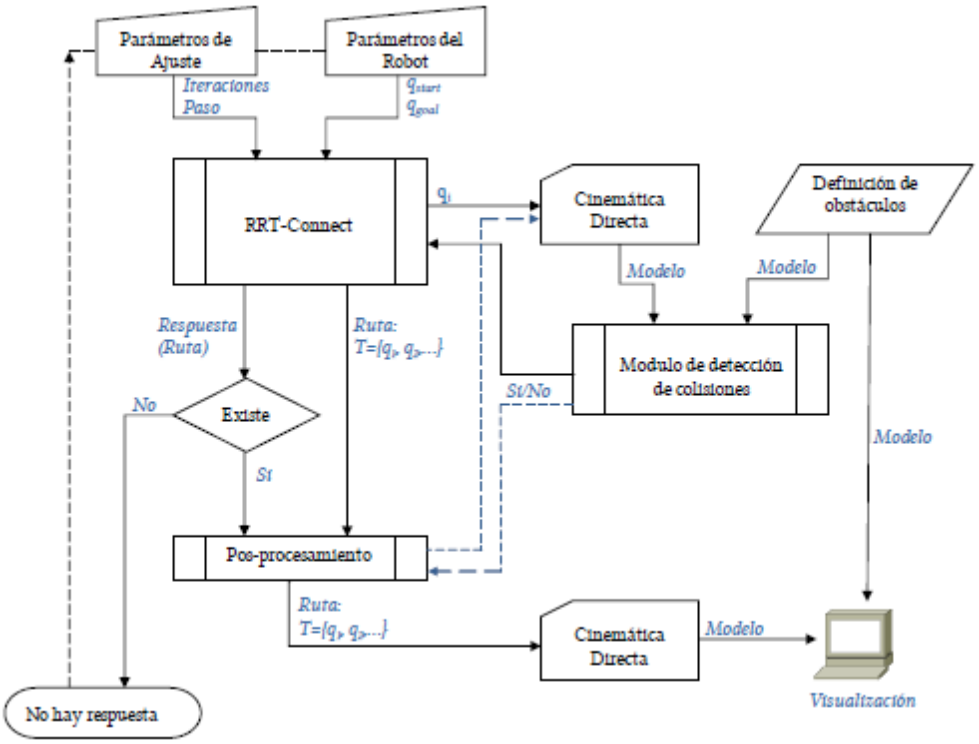


Figura 11. Diagrama para el cálculo de trayectoria libre de colisiones para implementación Computacional
(López García , 2012)

2. METODOLOGÍA Y DISEÑO

El desarrollo del proyecto se realizará mediante la metodología mecatrónica del modelo en “VV” que se presenta en la Figura 12. El proceso comienza definiendo los requerimientos del sistema los mismos que servirán como base para el diseño. Con el diseño establecido se procede a implementar el modelo virtual del sistema, con el que se realizarán todos los análisis y pruebas necesarias que garanticen el cumplimiento de los requerimientos previamente establecidos. El resultado del proceso será un sistema funcional.

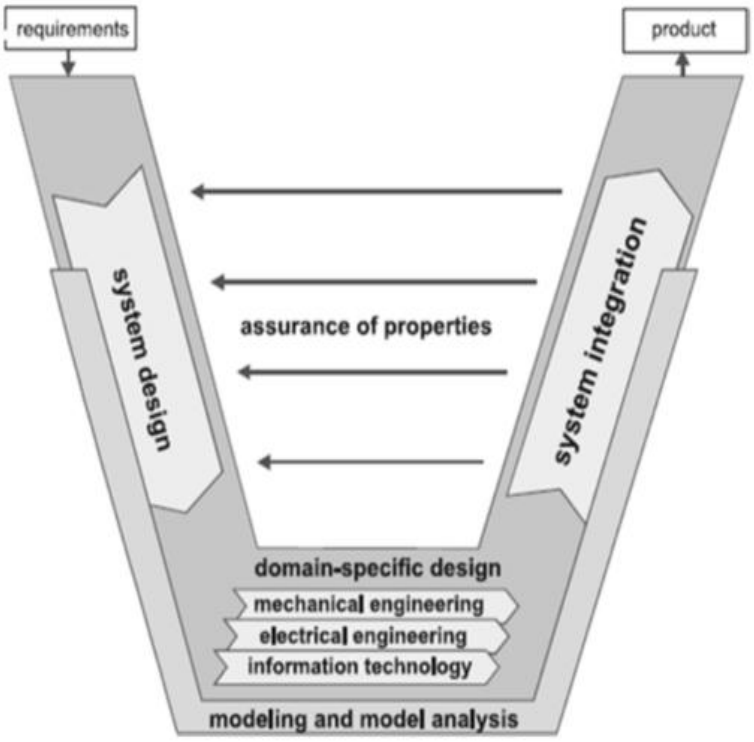


Figura 12. Metodología de diseño mecatrónico modelo en V
(Verein Deutscher Ingenieure,2004)

2.1. REQUERIMIENTOS DEL SISTEMA

Entre los requerimientos del sistema se investigó las características fundamentales de operación del sistema de visión artificial, en los cuales se tiene distintas funciones, como por ejemplo: el tratamiento de la imagen, tipos de imagen digital, el color de un objetivo y las operaciones básicas del tratamiento digital.

Los requerimientos del sistema son los siguientes:

- sistema debe detectar obstáculos en una imagen de fondo mediante el uso de visión artificial.
- El plano de visión y de detección de objetos debe ser un área de 320x240 pixeles o 11.29x8.47 cm, debido a que esa es la resolución de la cámara web.

- El sistema de visión artificial debe trabajar con imágenes de tipo mapa de pixeles y en colores RGB.
- El sistema móvil debe ser capaz de reconocer los objetos dentro de su campo visual y evitar colisiones, con tiempos de reacción en el orden de los milisegundos.
-

2.2. DISEÑO DEL SISTEMA DE CONTROL

Para el diseño de control fue necesario establecer la herramienta con las mejores prestaciones para el desarrollo del programa, por lo que se consideraron tres alternativas y se analizaron algunas variables para la selección. Las alternativas analizadas se encuentran en la Tabla 1.

Tabla 1. Alternativas de software para el desarrollo del sistema

	Matlab	**Labview Completo**	**OpenCV**
Sistemas operativos soportados	Windows Linux Mac	Windows Linux Mac	Windows Linux Mac IOS Android
Tipo de software	De pago	De pago	Software libre
Lenguajes de programación	Nativo	Nativo	C++/C Phyton Java
Ejecución en tiempo real del código	Si, Real-Time Simulation and Testing	Requiere complemento: LabVIEW Real-Time	No
Librerías de visión artificial	Si, Image Processing and Computer Vision	Requiere complemento: Módulo NI Vision Development	Si, OpenCV
Experiencia previa en el uso del software	Si	Si	No
Tipo de licencia	30 días de prueba versión completa Licencia para estudiantes	30 días de prueba versión básica Licencia para estudiantes	Software libre
Precio en dólares	55.00	120.00	Gratis

El mayor inconveniente del software de pago es su precio, pero es posible conseguir versiones de pruebas con menos herramientas para el desarrollo de este tipo de proyectos. En el caso de Matlab cuenta con una versión para

estudiantes con la librería de visión artificial, mientras que la versión de prueba de Labview no tiene los complementos necesarios para el desarrollo del sistema. Por último el uso de OpenCV se descartó debido a que no tiene un sistema de análisis en tiempo real propio, requiere el uso de otras herramientas de programación, lo que causaría retrasos en el desarrollo, además de no tener experiencia previa en el uso de esta librería. Según los elementos analizados se seleccionó como herramienta de desarrollo Matlab.

El sistema de control está compuesto por dos procesos; el primero es el proceso de imágenes. Encargado de adquirir, transformar, procesar y resolver imágenes. La tarea que se busca implementar es, obtener una imagen desde la cámara web que será utilizada de fondo, luego cualquier objeto que entre en rango de visión de la cámara y que no se encuentra en la imagen de referencia será identificado y transformado en una variable de operación para ser usado en el siguiente proceso.

El segundo proceso es el generador de trayectorias. Con el que se busca crear el camino óptimo desde un punto inicial a un punto final dentro de la imagen de fondo, el proceso es capaz de generar una trayectoria evitando los objetos considerados obstáculos, que fueron identificados fuera de la imagen de fondo.

2.3. DIAGRAMA DE FLUJO DEL SISTEMA

Para comprender el sistema y tener una visión establecida del resultado deseado, es necesario diseñar el diagrama de flujo que representa el funcionamiento fundamental del mismo. Se obtuvo el diagrama mostrado en la Figura 13.

El flujo del sistema inicia con el arranque del programa, seguido de la detección e inicialización de la cámara web para la adquisición de imágenes. Luego es necesario definir la imagen de fondo que servirá como referencia para la detección de los objetos. El programa debe tomar una decisión relacionada a la detección de la imagen de fondo no se seguirá hasta que este definida. Posteriormente se tiene la tarea de detección de objetos en la imagen de fondo, el programa observa todos los objetos que no forman parte de la imagen definida como fondo y los identifica, estos objetos son guardados como datos de obstáculos para la siguiente tarea.

Para la segunda parte del diagrama de flujo se tiene un subproceso que es el módulo de detección de colisiones que recibe como dato inicial los objetos identificados por el proceso de visión artificial. Al ser identificados los obstáculos, se procede a aplicar el algoritmo RRT para generar una ruta. Si la ruta no existe, porque se detectó alguna colisión o no se llegó al destino, se da por terminado el flujo del programa y no se tiene una solución. Si la ruta

existe, la solución es enviada a un subproceso donde la interfaz de usuario presenta la solución en pantalla, y se da por finalizado el flujo de programa.

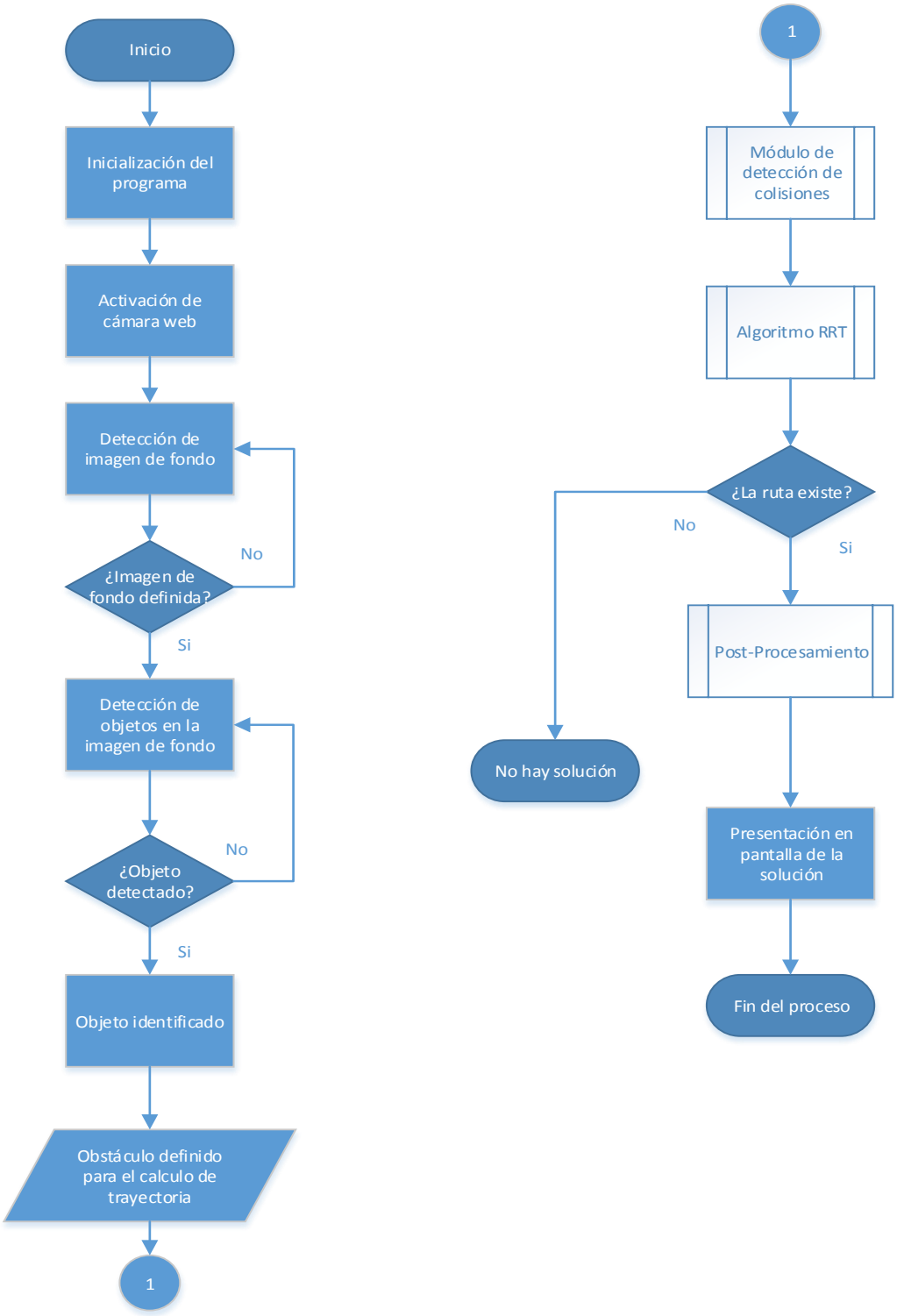


Figura 13. Diagrama de flujo del sistema

PROCESOLa primera fase del sistema de visión artificial es la adquisición de las imágenes, en el sistema diseñado esto se realiza mediante una cámara web. Posterior a la adquisición es necesario almacenar la información de las imágenes dentro de variables para su procesamiento. En este caso se tienen

dos variables, una que representa el fondo o background, que sirve como referencia para la detección de objetos, esta imagen es estática y será definida por el usuario; la otra variable contiene los datos de las imágenes que observa la cámara en tiempo real y servirán para detectar los objetos que no se encuentran en la imagen de fondo.

Para poder comparar las imágenes es necesario realizar algunas operaciones con el fin de facilitar los cálculos. La primera operación consiste en transformar las imágenes RGB a escalas de grises (Figura 14), debido a que uno de los métodos con mejores resultados para detección de objetos se realiza mediante el análisis de la luminancia de las imágenes.

Con las imágenes en escala de grises se obtiene una matriz de colores con diferentes tonalidades de blanco y negro. Los datos de la matriz toman valores entre 0 y 255, las dimensiones de la matriz dependen del número de pixeles de la imagen. En este caso se cuenta con una cámara que genera imágenes de 320x240 pixeles, por lo que la matriz generada tiene la misma dimensión.

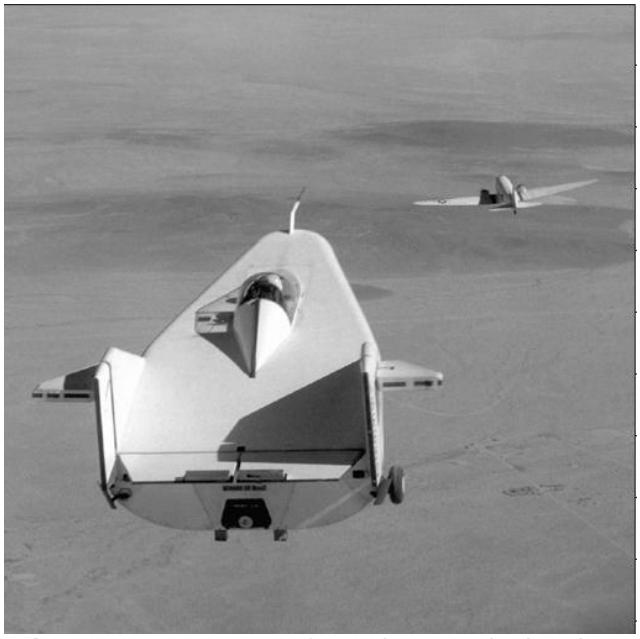


Figura 14. Imagen transformada a escala de grises

Para el proceso de comparación es necesario reducir el número de datos que se desea procesar de cada imagen, la siguiente operación es transformar los datos de la matriz de imagen a números binarios. De esta manera se pueden utilizar comparadores booleanos para determinar el estado de cada pixel de la imagen y tener un mejor resultado en la detección de los objetos.

Mediante una suma lógica de las imágenes con una comparación XOR, es posible determinar las diferencias que existe entre cada tonalidad de los pixeles que componen ambas imágenes. La información obtenida se almacena en una nueva variable que forma una imagen donde se podrá observar los objetos identificados por la operación (Figura 15 y Figura 16).

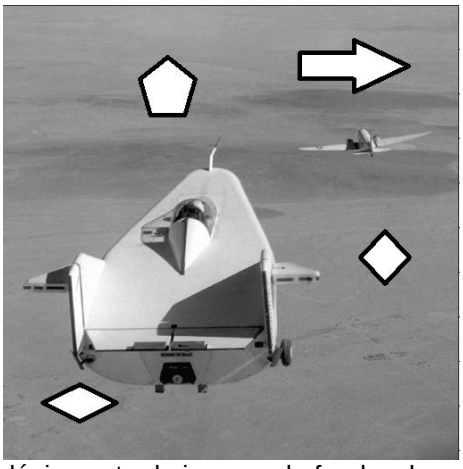


Figura 15. Suma lógica entre la imagen de fondo y los objetos colocados

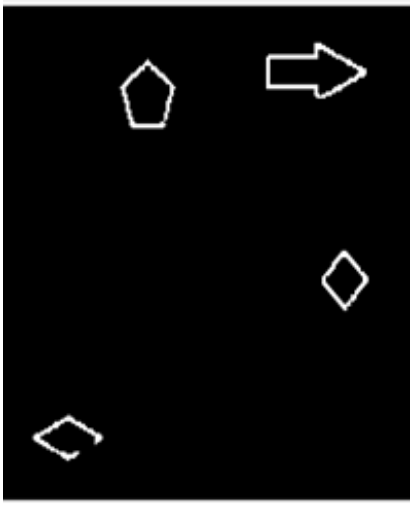


Figura 16. Objetos identificados en la imagen de fondo

2.4. IMPLEMENTACIÓN DEL ALGORITMO RRT

La implementación del algoritmo RRT se simulo, mediante el siguiente código de generación de trayectorias, con el mismo se observó el posible comportamiento de un sistema móvil. En la Figura 17 se puede observar la inicialización de variables del sistema.

```
x_max = 1000;
y_max = 1000;
obstacle1 = [400,150,300,200];
obstacle = [0,350,200,300];
obstacle2= [300,550,400,400];
EPS = 40;
numNodes = 500;

q_start.coord = [0 0];
q_start.cost = 0;
q_start.parent = 0;
q_goal.coord = [999 999];
q_goal.cost = 0;
```

Figura 17. Líneas de código correspondiente a la inicialización de datos

La primera parte del código define las condiciones iniciales del sistema, definiendo algunos factores. El espacio o plano donde se iba a realizar la simulación tiene una dimensión de 1000x1000 unidades. Se crearon tres obstáculos con formas rectangulares de diferentes tamaños, los puntos de referencia en el plano se los puede observar en la Figura 17. La coordenada de inicio de la trayectoria se la situó en el punto (0,0) mientras que la coordenada de llegada se la colocó en el punto (999,999) del plano. El número máximo de nodos que se generaran son 500, este número está relacionado al espacio que puede analizar el algoritmo y al número de interacciones que se generaran para el sistema como se muestra en la Figura 18. El resto de variables necesarias para la generación del RRT se inicializo en 0.

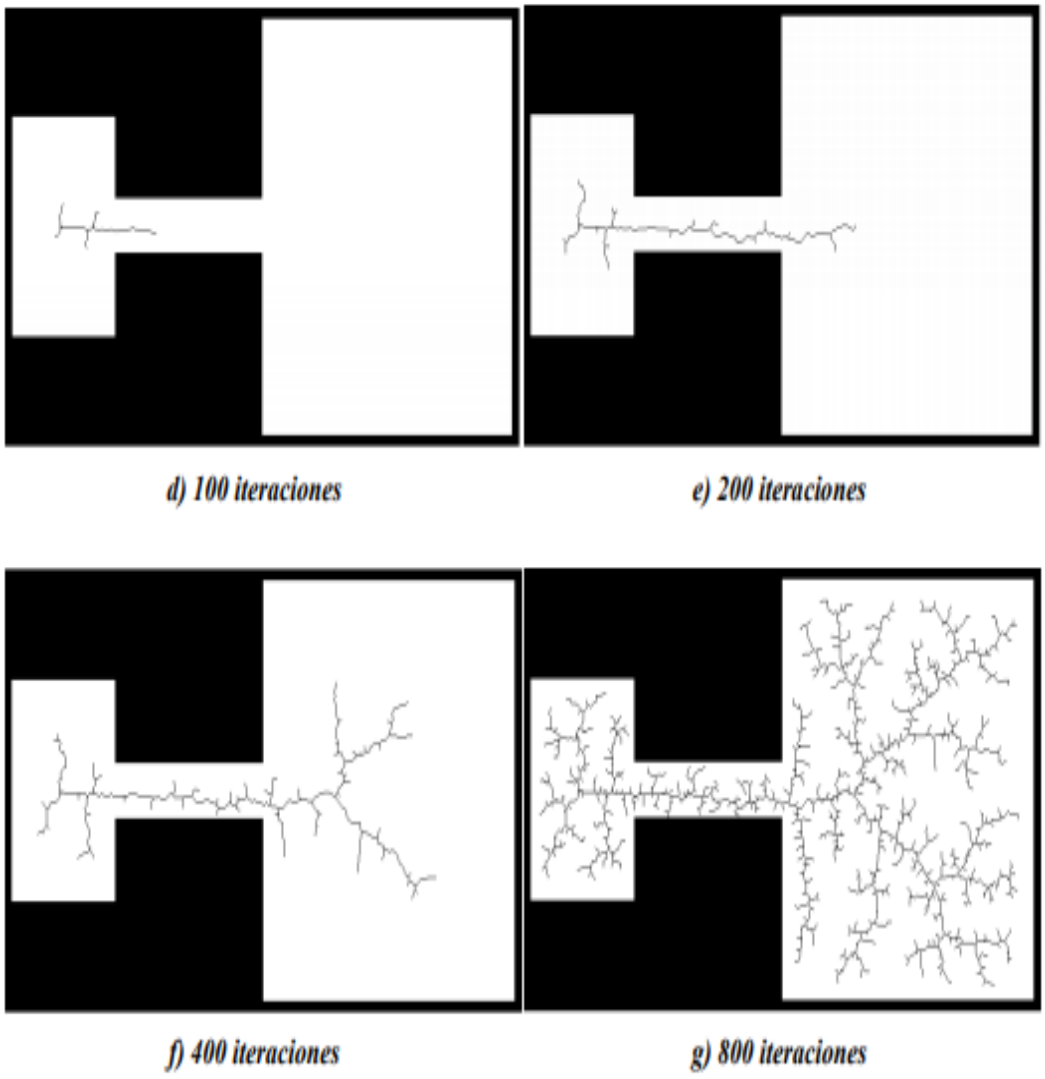


Figura 18. Comparación entre el número de nodos y las interacciones creadas
(López García , 2012)

Luego de establecer estos parámetros se genera un bucle FOR (Figura 19) para diagramar todos los nodos generados de manera aleatoria hasta encontrar el nodo de llegada, una vez encontrado ese punto el programa termina.

```
for i = 1:1:numNodes
    q_rand = [floor(rand(1)*x_max) floor(rand(1)*y_max)];
    plot(q_rand(1), q_rand(2), 'x', 'Color',  [0 0.4470 0.7410])

    % Termina si el nodo de llegada fue encontrado
    for j = 1:1:length(nodes)
        if nodes(j).coord == q_goal.coord
            break
        end
    end
```

Figura 19. Comando FOR para la generación de nodos

Se determina la posición de cada nodo mediante otro bucle FOR (Figura 20) con el fin de determinar posteriormente la trayectoria más corta generada por el algoritmo RRT y obtener el resultado final

```
% Selecciona el nodo más cercano al último generado
ndist = [];
for j = 1:1:length(nodes)
    n = nodes(j);
    tmp = dist(n.coord, q_rand);
    ndist = [ndist tmp];
end
[val, idx] = min(ndist);
q_near = nodes(idx);
q_new.coord = steer(q_rand, q_near.coord, val, EPS);
```

Figura 20. Bucle FOR para determinar la posición de los nodos

Una vez determinados los nodos es necesario indicar las zonas de colisión con los objetos establecidos, donde no se deben generar nodos ni trayectorias. Esto se realiza mediante un condicional que analiza los nodos en un radio conocido, este radio se establece con un valor inicial de 100 unidades debido a que entre mayor sea la distancia hacia la meta los nodos deben tener una mayor separación, sin embargo este valor disminuye entre más cerca se este del punto final con el propósito de crear una mejor ruta, por esta razón se puede observar que en el espacio cercano al punto de partida casi no existen nodos, pero cerca del punto de llegada existe una gran concentración de nodos generados. Los nodos también determinan si existe alguna colisión entre el nodo generado y la posición de los obstáculos en el plano como se muestra en la Figura 21.

```
if noCollision(q_rand, q_near.coord, obstacle) && noCollision(q_rand, q_near.coord, obstacle1)&& noCollision(q_rand, q_near.coord, obstacle
    line([q_near.coord(1), q_new.coord(1)], [q_near.coord(2), q_new.coord(2)], 'Color', 'k', 'LineWidth', 2);
    drawnow
    hold on
    q_new.cost = dist(q_new.coord, q_near.coord) + q_near.cost;

    % Dentro del radio r se encuentran todos los nodos existentes
    q_nearest = [];
    r = 100;
    neighbor_count = 1;
    for j = 1:1:length(nodes)
        if noCollision(nodes(j).coord, q_new.coord, obstacle) && dist(nodes(j).coord, q_new.coord) <= r
            q_nearest(neighbor_count).coord = nodes(j).coord;
            q_nearest(neighbor_count).cost = nodes(j).cost;
            neighbor_count = neighbor_count+1;
        end
```

Figura 21. Condicionante IF para cálculo de colisiones

El siguiente paso es crear el camino más corto entre el nodo de salida y el de llegada. Esto se realiza analizando la posición de cada nodo con todos los demás nodos cercanos. Mediante un bucle FOR (Figura 22) se determina esta variable.

```
% Se inicializa los valores de los nodos con los menores conocidos
q_min = q_near;
C_min = q_new.cost;

% Interactuan todos los nodos cercanos entre si para encontra el
% camino mas corto entre en nodo de partida y el de llegada

for k = 1:1:length(q_nearest)
    if noCollision(q_nearest(k).coord, q_new.coord, obstacle) && q_nearest(k).cost + dist(q_nearest(k).coord, q_new.coord) < C_min
        q_min = q_nearest(k);
        C_min = q_nearest(k).cost + dist(q_nearest(k).coord, q_new.coord);
        line([q_min.coord(1), q_new.coord(1)], [q_min.coord(2), q_new.coord(2)], 'Color', 'g');
        hold on
    end
end
```

Figura 22. Bucle FOR para determinar el camino más corto de llegada

Se comprueba que las distancias sean las menores posibles con el uso de un IF (Figura 23) y se almacenan los nodos generados para el cálculo de la trayectoria.

```
        for j = 1:1:length(nodes)
            if nodes(j).coord == q_min.coord
                q_new.parent = j;
            end
        end

        % Se añaden los nodos
        nodes = [nodes q_new];
    end
end

D = [];
for j = 1:1:length(nodes)
    tmpdist = dist(nodes(j).coord, q_goal.coord);
    D = [D tmpdist];
end
```

Figura 23. Comprobación de nodos

Por último se realiza el código que genera el diagrama que corresponde a la trayectoria generada por la unión de los nodos (Figura 24).

```
% Se diagrama el camino óptimo desde el punto de partida hasta el punto llegada
[val, idx] = min(D);
q_final = nodes(idx);
q_goal.parent = idx;
q_end = q_goal;
nodes = [nodes q_goal];
while q_end.parent ~= 0
    start = q_end.parent;
    line([q_end.coord(1), nodes(start).coord(1)], [q_end.coord(2), nodes(start).coord(2)], 'Color', 'r', 'LineWidth', 2);
    hold on
    q_end = nodes(start);
end
```

Figura 24. Generación del diagrama de trayectorias

Como resultado final para la generación de la trayectoria se obtuvo lo mostrado en la Figura 25.

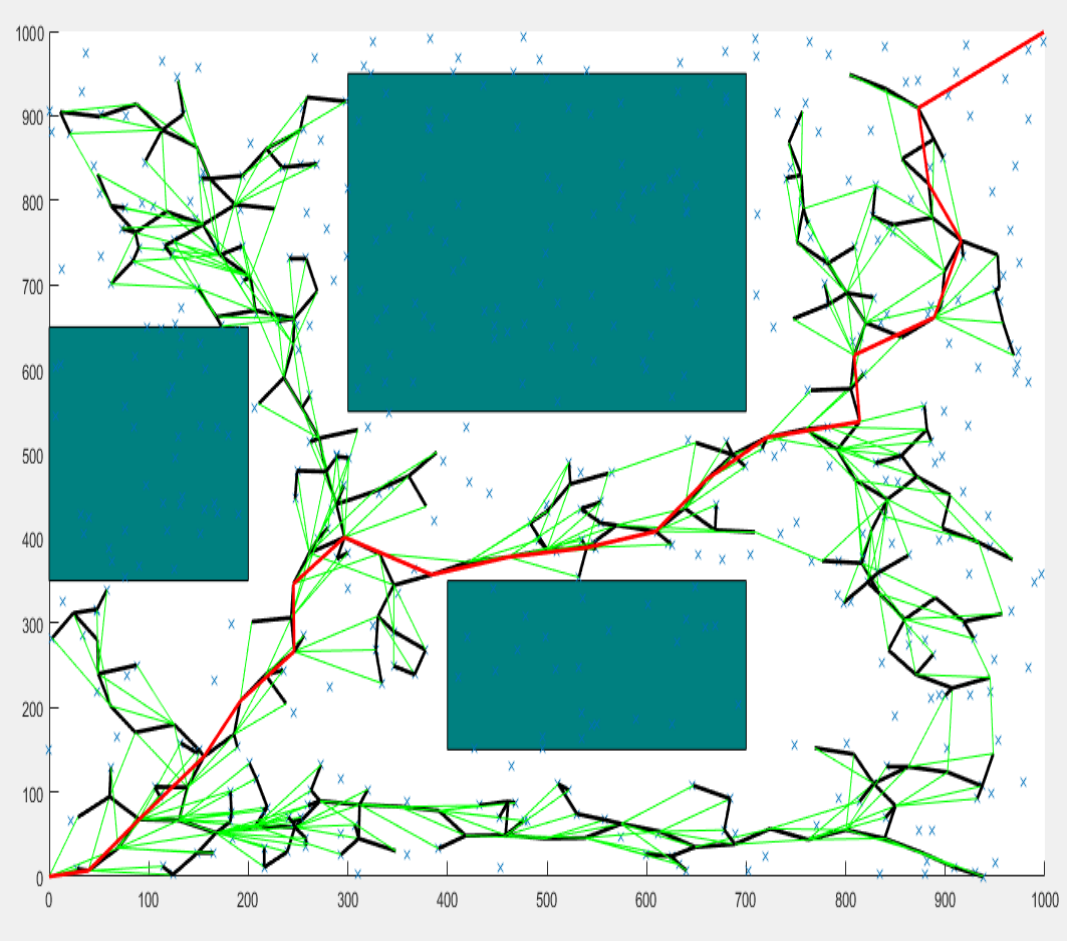


Figura 25. Plano de trayectoria generado en Matlab

2.5. INTERFAZ DE USUARIO

Para la interfaz de usuario se buscó un sistema donde se pueda visualizar la imagen de fondo y la imagen de los obstáculos. En la interfaz de usuario se debe seleccionar la cámara que se usará para adquirir las imágenes (Figura 26). Luego se debe determinar la imagen de referencia o fondo. Las imágenes de los objetos se generaran de manera automática por el sistema de visión artificial en base a lo que observe la cámara en tiempo real (Figura 27).

Figura 26. Pantalla de inicio de la interfaz de usuario

Figura 27. Pantalla de selección de fondo

El usuario debe seleccionar dos puntos de la imagen mostrada el primero representa el punto de partida y el segundo el punto de llegada (Figura 28).

En la pantalla de inicio se encuentra dos imágenes y un puntero de selección de coordenadas.

Figura 28. Pantalla de selección de puntos de partida y llegada

El usuario debe elegir dos puntos mediante el selector de coordenadas como se muestra en la siguiente imagen. El primero se tomará como punto de partida y el segundo como punto de llegada como se muestra en la Figura 29.

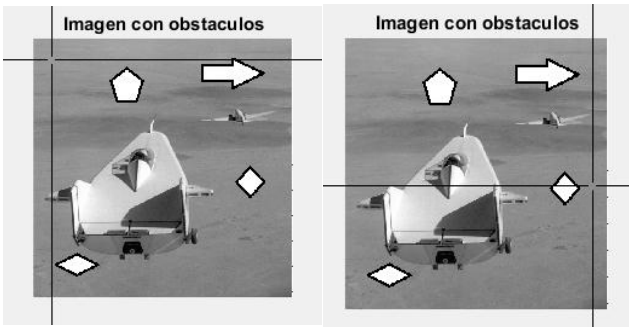


Figura 29. Selección de puntos partida y llegada

Por último el usuario puede generar la ruta detectada para las imágenes establecidas y los puntos seleccionados, obteniendo como resultado la ruta final calculada por el programa.

2.6. UNIFICACIÓN DE LOS SISTEMAS DE VISIÓN ARTIFICIAL GENERACIÓN DE TRAYECTORIAS RRT E INTERFAZ DE USUARIO

Para la prueba se colocó una imagen de fondo, posteriormente se sumó otra imagen con figuras de obstáculos. Estos obstáculos serán identificados por el sistema como objetos que no estaban en la imagen inicial y los usará para el análisis de trayectoria. Mediante los procesos de operaciones con imágenes se determinará el espacio del plano y los marcos de los objetos que deben ser resueltos por el algoritmo RRT para generar la trayectoria. El proceso de resolución se muestra en la Figura 30.

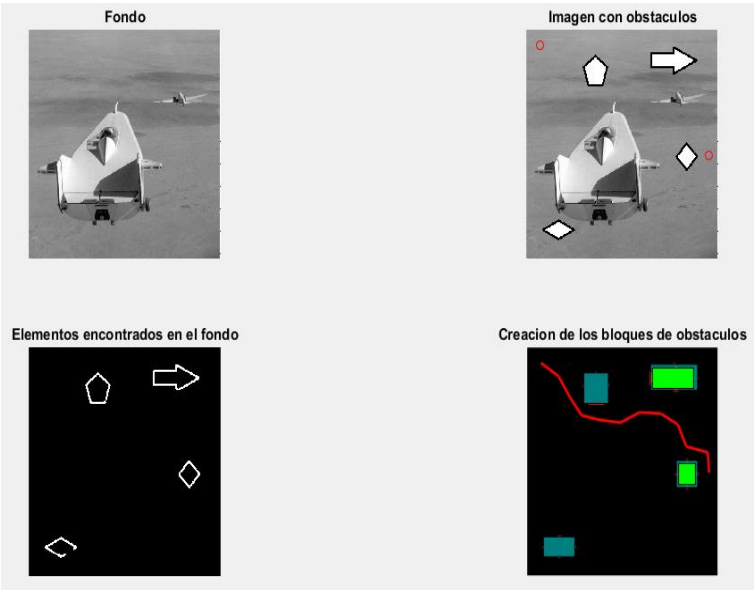


Figura 30. Cálculo de trayectorias con visión artificial y algoritmo RRT

El principio de funcionamiento del algoritmo RRT es el mismo antes explicado, pero esta vez los objetos que representan obstáculos no serán inicializados mediante líneas de código, sino que serán determinados previamente por el sistema de visión artificial. De igual manera mediante la interfaz de usuario se determina el punto de partida y de llegada para el cálculo. El algoritmo es capaz de entender estos objetos como limitantes del camino y evitará las colisiones con los mismos, generando la trayectoria más corta desde el punto de partida hasta el punto de llegada. Simulando el sistema se obtuvo el resultado mostrado en el Figura 31.

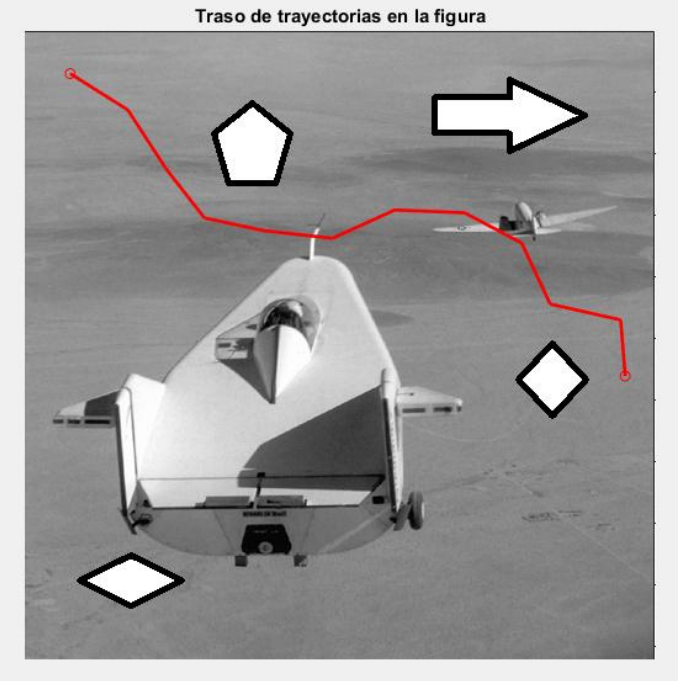


Figura 31. Imagen de la suma del fondo, los obstáculos y la trayectoria

3. ANÁLISIS DE RESULTADOS

Para comprobar el funcionamiento del sistema se procedió a realizar algunas pruebas con el fin de determinar la fiabilidad del mismo. Las pruebas se centraron en la modificación de fotogramas por segundo analizado por el sistema de visión artificial y la cantidad de nodos generados por el código RRT para el cálculo de trayectorias. Los resultados se muestran en la Tabla 2:

Tabla 2. Tabla de pruebas con FPS para el sistema de visión artificial

FPS	Número de objetos colocados	Número de objetos detectados	Tiempo de respuesta en milisegundos (ms)
1	4	0	1
5	4	2	100
10	4	4	500
15	4	4	600
20	4	4	700
30	4	4	1000
40	4	4	1500
50	4	4	5000
60	4	4	10000

Como resultado de la variación de FPS se establece que si el número es menor a 10 fpsmenor a 10 fps, aunque exista la respuesta de tiempo este entre 1-500 milisegundos, el sistema no alcanza a reconocer o diferenciar todos los objetos dentro de la imagen de fondo, esto se debe a que la cantidad de fotogramas analizados no proporciona la información suficiente sobre el entorno. Por otra parte, si la cantidad de fotogramas es mayor a 50 fpsmayor a 50 fps, se tiene un reconocimiento de los objetos dentro del fondo, pero existe un problema relacionado con el tiempo de respuesta, entre mayor es la cantidad de fotogramas también es mayor el tiempo de computo, lo que significa que si se implementaría este sistema en un robot móvil real no podría esquivar los obstáculos debido a los tiempos de respuesta.

Como resultado se tiene que la cantidad de óptima de FPS para detectar 4 objetos en la imagen de fondo, es un valor entre 10-15 fotogramas por segundo, con este valor se reconocieron todos los objetos y se tiene una respuesta de tiempo entre 500-600 ms..

Para el cálculo y resolución de trayectoria el factor más importante en el sistema, es el número de nodos que el algoritmo RRT debe crear para construir la solución. Los resultados obtenidos fueron probados con un valor

de 15 fps, y realizando variaciones del número de nodos generados como se muestra en la Tabla 3:

Tabla 3. Tabla de pruebas de número de nodos para el algoritmo RRT

Número de nodos	Colisión con obstáculos	Llegada a la meta	Tiempo de respuesta en segundos
50	Si	No	1
100	Si	No	1
200	Si	No	3
300	No	No	4
400	No	Si	4
500	No	Si	5
600	No	Si	5
700	No	Si	8
800	No	Si	8

De los resultados obtenidos se observa, que si la cantidad de nodos es menor que 400 surgen dos problemas, el primero se relaciona a la suficiencia del sistema para esquivar los obstáculos y el segundo a capacidad del sistema para llegar a la meta, esto se debe a que si no existe el número suficiente de nodos el sistema no es capaz de calcular y construir un camino que llegue a la meta y a la vez esquive los obstáculos. Por otra, parte si el número de nodos es mayor a 600 se obtiene una trayectoria uniforme con un camino recto y definido, pero esto también implica un mayor tiempo de respuesta por lo que sería un inconveniente al momento de la implementación en sistemas físicos como robots diferenciales de movimiento. Se establece que el número óptimo de nodos para resolver la trayectoria se encuentra entre 500-600 nodos con un tiempo de resolución de 5 s.

Los resultados finales mostrados en las **Figura 32**4 y Figura 35,se obtuvieron con la configuración del sistema de visión artificial con 15 FPS y para el algoritmo RRT una cantidad de 600 nodos. El sistema fue probado utilizando una cámara web en tiempo real. La configuración de cámara se puede observar en la Figura 32 mientras que la selección de puntos de partida y llegada se muestra en la Figura 33.

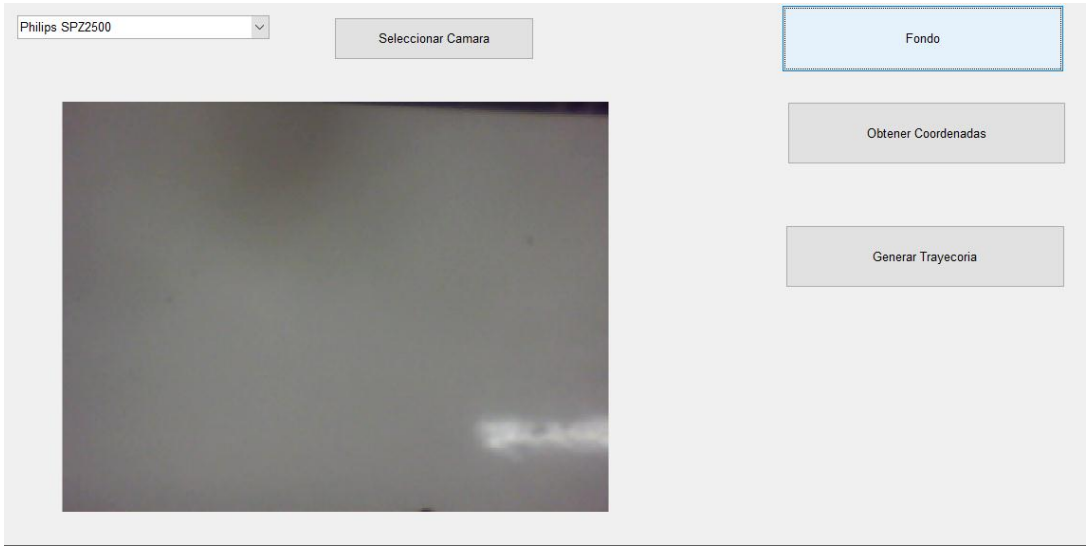


Figura 32. Imagen de fondo detectada por cámara

Figura 33. Selección del punto de partida y llegada para el cálculo de trayectoria

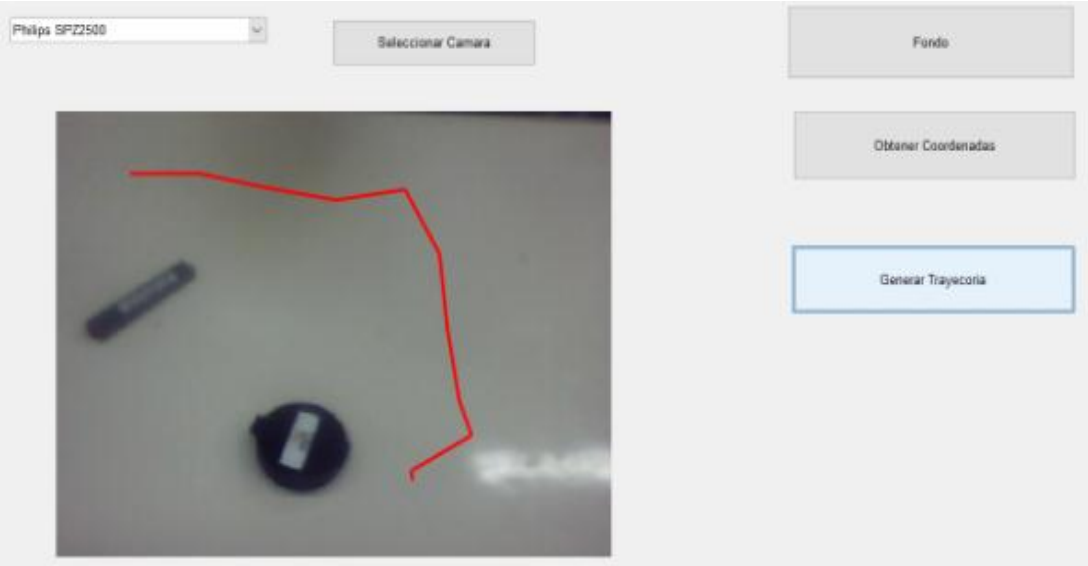


Figura 34. Trayectoria final calculada con dos obstáculos

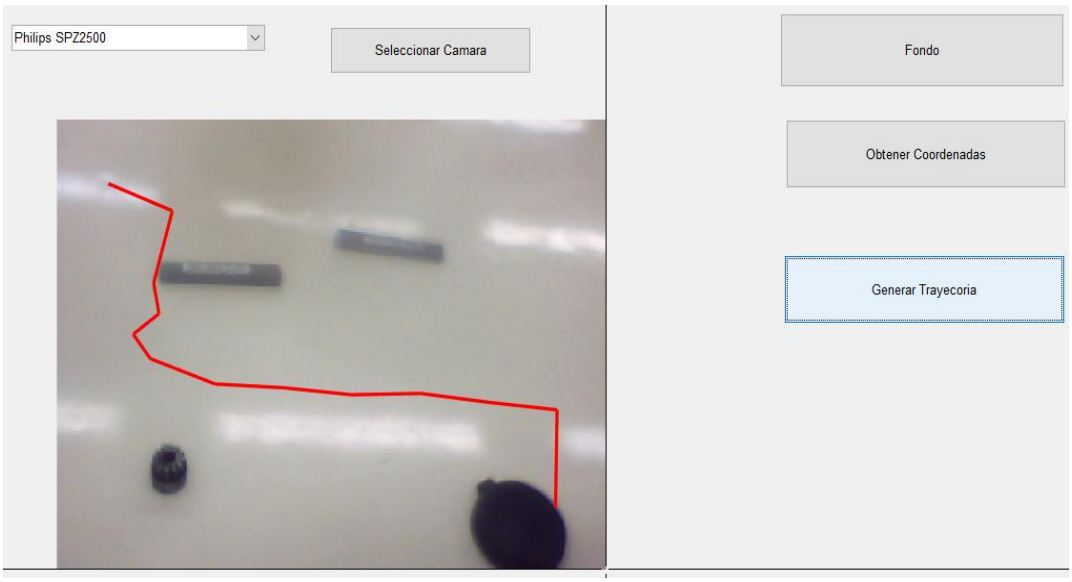


Figura 35 Trayectoria final calculada con cuatro obstáculos

4. CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Como resultado del presente proyecto se obtuvo un sistema de visión artificial capaz de diferenciar objetos dentro de una imagen de fondo previamente establecida. Y mediante el uso de un algoritmo RRT procesar la información de las imágenes para crear una trayectoria deseada para el sistema móvil.

- El algoritmo RRT desarrollado se aplicó a imágenes estáticas de resolución 320x240 pixeles previamente cargadas al programa teniendo como resultado una efectividad del 100%, siempre se resolvió la trayectoria. Posteriormente el programa fue adaptado para funcionar en tiempo real con el uso de la cámara web teniendo los resultados mostrados en la tabla 2 y tabla 3, definiendo como los factores de importancia la cantidad de fotogramas por segundo analizados y el número de nodos generados para la trayectoria.

- Los factores que causan mayores inconvenientes para el análisis son las distorsiones y ruidos presentes en las imágenes, producidos por un entorno no controlado, esto dificulta considerablemente el reconocimiento de objetos. En la fase de experimentación en tiempo real, la presencia de pequeñas sombras que modifican el color del objeto y la oclusión parcial de los objetos que modifican su forma, han demostrado la debilidad del sistema frente a estas variables.

- Si el número de fotogramas por segundo analizado es menor que 10 FPS el sistema no puede reconocer los objetos y si el número es mayor que 50 FPS, los tiempos de respuesta aumenta debido entre mayor sea el número de fotogramas, mayor será la cantidad de datos que debe procesar y analizar el sistema. Se estableció un valor de 15 FPS como óptimo para el presente proyecto.

- El número de nodos que se establece para el cálculo de trayectorias del algoritmo RRT determina la fiabilidad del resultado generado por el programa. Si el número de nodos es menor a 500 la trayectoria que se genera tiende a generar colisiones con los obstáculos o a no llegar al punto de destino. Si el número de nodos es mayor a 600 se mejora la resolución de la trayectoria pero se requiere más tiempo de resolución. Se estableció un valor de 600 nodos como óptimo para el presente proyecto.

RECOMENDACIONES

- Se recomienda utilizar la información mostrada en el presente proyecto para la implementación de nuevos sistemas de visión artificial o de resolución de trayectorias mediante algoritmos RRT's. Considerando que cada sistema tiene diferentes variables y condiciones por lo que este trabajo solo debe considerarse como una guía o referencia.

- Se recomienda implementar otras soluciones de visión artificial con sistemas dedicados o embebidos con el fin de mejorar los tiempos de respuesta del sistema final.

- Para solucionar la problemática dada por la oclusión parcial e incluso completa que pueden presentar algunos de los objetos en la imagen, se podría implementar un filtro de Kalman. Este filtro predice la posición del objeto basándose en los parámetros de su posición en la imagen anterior a la actual. Estos parámetros pueden incluir valores de velocidad, dirección y aceleración, que permiten estimar la posición aproximada del objeto aunque éste se encuentre oculto.

BIBLIOGRAFÍA

- Ana González Marcos, F. J. (2006). Técnicas y algoritmos básicos de visión artificial. Logroño, La Rioja: Universidad de la Rioja .
- Casado Fernández, C. (2010). Manual Básico de Matlab. Madrid, España: Servicios informáticos U.C.M.
- Domingo, M. (2004). Vision por Computador. Santiado de Chile: Universidad Católica de Chile.
- Gilat, A. (2006). Matlab, una introducción con ejemplos prácticos. Barcelona, España: Editorial Reverté, S.A.
- Krig, S. (2014). Computer vision metrics, survey, taxonomy and analysis. New York, Estados Unidos: Apress Open.
- López García , D. (2012). Nuevas aportaciones en algoritmos de planificación parala ejecución de maniobras en robots autónomos no la ejecución de maniobras en robots autónomos no. Huelva, España: Universidad de Huelva.
- Obinata, G., & Dutta, A. (2007). Vision System Applications. Vienna, Austria: I-Tech Education and Publishing.
- Pérez López, C. (2002). Matlab y sus aplicaciones en las ciencias y la ingeniería. Madrid, España: PEARSON EDUCACIÓN.
- Quintero Torres, R., & Quintero Bermúdez, R. (2012). Matlab para principiantes. Mexico DF: Universidad Nacional Autónoma de México.
- Solari, F., Chessa, M., & Sabatini, S. (2012). Machine vision applications and systems. Rijeka, Croacia: InTech.
- Sucar, E., & Gómez, G. (2010). Visión Computacional. Puebla, Mexico: Instituto nacional de astrofísica, óptica y electrónica.