

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA



TITULOXXX

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE:
INGENIERO MECATRÓNICO

PAREDES MERINO, JORGE SALVADOR

PROMOCIÓN 2010-II

LIMA - PERÚ

2011

Dedicatoria

Dedico el presente trabajo a mi familia por su apoyo incondicional, lo cual me permitió culminar mis estudios así como también un agradecimiento especial a los docentes de mi facultad por la orientación brindada a lo largo de mis 5 años de estudio. Finalmente agradecer a mis compañeros por las gratas experiencias compartidas en esta importante etapa de mi vida.

Agradecimientos

Quería agradecer de manera especial al Ing. José F. Oliden Martínez por la ayuda y asesoramiento brindado para la obtención de los robots móviles Moway, los cuales permitieron implementar los algoritmos desarrollados físicamente en la presente tesis. A su vez dichos robots quedan en manos de INIFIM para el uso posterior de algún proyecto que algún alumno desarrolle en un futuro.

TABLA DE CONTENIDOS

PRÓLOGO	1
CAPÍTULO 1.	
INTRODUCCIÓN	3
1.1. Antecedentes	3
1.2. Motivación de la Investigación	7
1.3. Objetivos	7
1.4. Alcances	8
1.5. Limitaciones	8
CAPÍTULO 2.	
PLANIFICACIÓN DE TRAYECTORIA	9
2.1. Definiciones básicas	9
2.1.1. Tiempo	9
2.1.2. Modelo	10
2.1.3. Estado	11
2.1.4. Acciones	11
2.1.5. Estado inicial y final	11
2.1.6. Configuración inicial y final	12
2.1.7. Entorno	12
2.1.8. Criterio	12
2.1.9. Plan	14
2.2. Tipos de Algoritmos de Path Planning	15
2.2.1. Cell decomposition	15
2.2.2. Potential fields	16
2.2.3. Roadmaps	19
2.3. Algoritmo de path planning RRT	19
2.4. Metodología de Trabajo	20
CAPÍTULO 3.	
ALGORITMO RRT	21
3.1. Elementos	22
3.2. Pseudocódigo	23
3.3. Algoritmos evolucionados de RRT	25
3.3.1. RRT Bidireccional Básica	25

3.3.2. RRT Ext-Ext	26
3.3.3. RRT Ext-Con	28
3.4. Sistemas no holónomos	33
 CAPÍTULO 4.	
SIMULACIÓN DE RESULTADOS Y PERFORMANCE DE EVALUACIÓN	
4.1. Simulación del algoritmo RRT en sistemas holónomos	34
4.1.1. Algoritmo RRT	34
4.1.2. Algoritmo RRT bidireccional	35
4.1.3. Algoritmo RRT Ext-Ext	36
4.1.4. Algoritmo RRT Ext-Con	37
4.2. Simulación del algoritmo RRT en sistemas no holónomos	41
4.2.1. Modelamiento Cinemático del Sistema	41
 CAPÍTULO 5.	
IMPLEMENTACIÓN DE PLATAFORMA E INTERFAZ	
5.1. Consideraciones	48
5.2. Funcionamiento de la interfaz	49
5.3. Características de la interfaz	50
5.4. Componentes de la Interfaz de Usuario	52
5.4.1. Sección de Comunicación Inalámbrica	52
5.4.2. Sección de Visión Artificial	56
 CAPÍTULO 6.	
Aportes para Aplicaciones Académicas	
6.1. Plataforma de pruebas	65
6.2. Ambiente desarrollado en C#	66
6.3. Simulación 3D	66
 CONCLUSIONES	
 BIBLIOGRAFÍA	
 APÉNDICE A.	
Motion Planning	
A.1. Reseña histórica de robots móviles	73
 APÉNDICE B.	
Robot Moway	
B.1. Arquitectura	75
B.1.1. Procesadores	76
B.1.2. Sistema de Navegación	76
B.1.3. Sensores e Indicadores	77
B.1.4. Sistema de Potencia	77
	78

B.1.5. Conector de Expansión	78
B.2. Moway GUI	79
B.3. Interfaz C#	80

APÉNDICE C.

Visión Artificial **82**

C.1. Data de la Imagen	83
C.1.1. Representación de Imágenes Digitales	83
C.2. Histograma de una imagen	84
C.3. Sistema HSV	85
C.4. Operaciones morfológicas en imágenes	88
C.4.1. Dilatación	89
C.4.2. Erosión	90
C.4.3. Apertura	91
C.4.4. Cierre	92

LISTA DE FIGURAS

1.1. Robot Shakey(1968), figura tomada de [1].	4
1.2. Robots reconfigurables.	7
2.1. Cubo Rubik.	10
2.2. Moviendo piano usando robots móviles.	10
2.3. Muestreo poligonal de la configuración de espacio. Figura tomada de [2].	16
2.4. Calculo de la ruta. Figura tomada de [2].	16
2.5. Aplicación de campos potenciales en Path Planning. Figura tomada de [3]	17
2.6. Vista del planeador. Figura tomada de [3].	18
2.7. Encontrando la ruta usando gradiente. Figura tomada de [3].	18
2.8. Ejemplo de mínimo local. Figura tomada de [2].	18
3.1. Naturaleza del Algoritmo RRT.	21
3.2. Árbol de exploración.	24
3.3. Seleccionando un punto aleatorio.	24
3.4. Calculando el punto el elemento más cercano del árbol al punto aleatorio.	24
3.5. Adicionando nuevo elemento al árbol explorador.	24
3.6. Conección de árboles exploradores.	30
3.7. Un árbol genera un nuevo elemento (círculo gris oscuro).	31
3.8. El nuevo elemento del árbol se convierte en el objetivo (q_{target}) del otro árbol.	31
3.9. Calculando el elemento más cercano (q_{near}) a q_{target}	31
3.10. Cálculando nuevo elemento (q_{new}) del árbol de q_{init}	32
3.11. Adición consecutiva de elementos hasta no colisión.	32
3.12. Conexión final del crecimiento.	32
3.13. Tracking del camino de conexión.	32
4.1. Algoritmo RRT explorando en configuración libre de obstáculos.	35
4.2. Algoritmo RRT explorando a traves de obstáculos.	35
4.3. RRT bidireccional básico en espacio libre.	36
4.4. A la izquierda algoritmo RRT-Ext-Ext, a la derecha el algoritmo RRT básico bidireccional. Comparación entre ambas variaciones de RRT.	37

4.5.	Árboles exploradores en RRT-Ext-Con	38
4.6.	Tracking de árboles que forman la ruta entre los puntos deseados.	38
4.7.	Ruta entre los puntos.	39
4.8.	Configuración inicial y final del robot Moway así como su entorno con obstáculos.	39
4.9.	Árboles exploradores entre las configuraciones deseadas.	40
4.10.	Tracking de la ruta entre las configuraciones inicial y final.	40
4.11.	Secuencia de movimiento entre las configuraciones inicial y final en el entorno.	41
4.12.	Ruta entre los puntos.	41
4.13.	Ruta entre las configuraciones inicial y final en el entorno dado.	43
4.14.	Ruta a seguir por el robot Moway.	43
4.15.	Tracking del movimiento del robot Moway desde la configuración inicial a la final.	44
5.1.	Plano de base de la plataforma. Unidades milímetros.	45
5.2.	Plano de columna de soporte. Unidades milímetros.	46
5.3.	Plano de soporte de cámara. Unidades milímetros.	46
5.4.	Planta de Prueba	47
5.5.	Controlador retroalimentado de Trayectoria	49
5.6.	Controlador retroalimentado de Trayectoria	50
5.7.	Sistema de referencia para la interfaz.	51
5.8.	Secuencia de manipulación de la interfaz.	51
5.9.	Secuencia de la máquina de estados	52
5.10.	Rapidez de la rueda del robot Moway respecto al Duty cycle del motor DC que genera el movimiento rotacional.	54
5.11.	Comportamiento de la rapidez respecto al duty cycle del robot Moway	54
5.12.	Estructura del paquete total enviado a través del protocolo de la RF del robot Moway	55
5.13.	CANALES DE RF MOWAY PARA EVITAR COLISIONES CON OTRAS ONDAS	56
5.14.	Función ProcessFrame	57
5.15.	Rutina Calculo_vector perteneciente a ProcessFrame	58
5.16.	Filtro de obstáculos verdes, codificación en lenguaje C#.	59
5.17.	Captura del entorno por la interfaz.	59
5.18.	Histograma de la imagen capturada por la cámara	60
5.19.	Filtrado de robots moway en sistema de color HSV, sin aplicar operaciones morfológicas.	61
5.20.	Filtrado de robot Moway rojo en sistema HSV, aplicando ciertas operaciones morfológicas.	61
5.21.	Uso de operaciones morfológicas en el procesamiento de imágenes para la obtención de posición y orientación del robot Moway.	62
5.22.	Dimensiones del área de trabajo.	63
5.23.	Orientación del Robot Moway.	63

6.1. Vista de perfil	66
6.2. Vista superior	67
6.3. Vista inferior	67
6.4. Vista isométrica	68
B.1. Robots Moway. Tomado de [4]	75
B.2. Partes del robot Moway. Tomado de [4]	76
B.3. Drive System: electrónica y mecánica. Tomado de [4]	77
B.4. Sensores y grupo de indicadores. Tomado de [4]	77
B.5. Fuente de sistema de alimentación. Tomado de [4]	78
B.6. Módulo RF. Tomado de [4]	78
B.7. RF-USB. Tomado de [4]	78
B.8. Interfaz gráfica del robot Moway.	79
B.9. Sección de Control de RF en la Interfaz gráfica Moway GUI.	79
B.10. Interface Moway en C#	81
C.1. Para la computadora, lo que capta la cámara es una matriz de números. Tomado de [5].	83
C.2. De izquierda a derecha: a colores en RGB, en escala de grises e imagen binaria (con un threshold de 28).	84
C.3. Imagen sub-expuesta.	84
C.4. Imagen propiamente expuesta.	85
C.5. Imagen sobre-expuesta.	85
C.6. Imagen transformada y mostrada en el espacio de color HSV, tomada de [6].	86
C.7. Espacio de color HSV como una rueda de color.	87
C.8. Cono de colores del espacio HSV.	87
C.9. Algunos ejemplos de elementos morfológicos estructurales. El píxel centro de cada elemento estructural esta sombreado. Figura tomada de [6]	89
C.10. Aplicación de dilatación.	89
C.11. Aplicación de dilatación.	90
C.12. Aplicación de erosión.	90
C.13. Aplicación de dilatación.	91
C.14. Aplicación de dilatación.	91
C.15. Aplicación de dilatación.	92

LISTA DE CUADROS

3.1. Elementos de una RRT	22
5.1. Parámetros principales del módulo BZI-RF2GH4, datos tomados de [7].	53
5.2. Asignación de números para las órdenes de movimiento	55
5.3. Rango de colores en interés en el sistema HSV.	58

PRÓLOGO

Hoy en día, los robots son parte de nuestra vida. Los podemos encontrar como juguetes, móviles exploradores, brazos, humanoides, entre otros. Pero algo en común que requieren los robots es interrelacionarse con su entorno, el cual limita su accionar y le obliga a adecuarse a tales condiciones. En el caso de los móviles lo que se requiere primordialmente es trasladarlos de un punto específico a otro requerido para posteriormente ejecutar alguna tarea. El problema de controlar un robot móvil y la planificación de su trayectoria esta dado por el hecho de que su ambiente de trabajo está condicionado a cambios, sea con alta o baja tendencia. Por tal motivo se han desarrollado a lo largo de los últimos 50 años diversos algoritmos de Path Planning que permiten interactuar apropiadamente al robot con su medio. Por ejemplo, en el caso de robots manipuladores, sus articulaciones no deben colisionar con objeto alguno. Pero el cálculo de dichas trayectorias(de robots móviles y manipuladores) involucra factores tales como la geometría, cinemática, dinámica, entre otros que son los que limitan al robot. En la presente tesis se propone el uso de técnicas estocásticas en lo referente al planeamiento de trayectorias para robots móviles en , ambientes no dinámicos, es decir, se considerará que el entorno no presenta obstáculos en movimiento. Para éste estudio, se ha organizado el trabajo en siete capítulos que serán expuestos de la siguiente forma: En el capítulo I, se realiza una introducción a esta tesis, con los antecedentes , el motivo de la investigación, los objetivos, alcances y limitaciones. En el capítulo II, se presenta una revisión de la literatura sobre el planeamiento

de trayectorias. Además se presentan definiciones básicas que han de permitir un mejor entendimiento, así como también una descripción de los principales algoritmos de planamiento de trayectoria. En el capítulo III, se explica lo referente al algoritmo utilizado en la presente tesis desde sus elementos, el pseudocódigo y los algoritmos evolucionados a partir de este. En el capítulo IV, se presentan los resultados de las simulaciones realizadas así como la performance de evaluación. En el capítulo V, se explicará lo referente a la implementación de la plataforma y la interfaz. En el capítulo VI, se presentan los principales aportes de esta tesis para el desarrollo de aplicaciones académicas relacionadas al planeamiento de trayectorias. Finalmente se presentan las conclusiones y recomendaciones para futuras investigaciones. El apéndice A presenta temas relacionas a la robótica móvil. En el apéndice B se aborda las principales características del robot Moway y finalmente el apéndice C describe los recursos utilizados del campo de Visión Artificial a la presente tesis.

CAPÍTULO 1

INTRODUCCIÓN

1.1. Antecedentes

Entre 1966 y 1972 se desarrolló el robot Shakey (ver Figura 1.1), desarrollado por el Centro de Inteligencia Artificial del SRI¹, el cual fue realmente fundamental no solo para la robótica móvil sino para el desarrollo de la “Inteligencia Artificial”, ya que fue el primer robot móvil en visualizar su ambiente y poder interpretarlo. El robot tenía una cámara de televisión, un telémetro, bumpers y un sistema de video inalámbrico. Este trabajo sirvió de inspiración para muchos investigadores, en consecuencia, tanto la investigación y la variedad de diseños de robots móviles creció vertiginosamente (para mayor detalle ver Apéndice A).

¹“Stanford Research Institute”, es el nombre de fundación(1946) de este centro que pertenecía a la Universidad de Standford. Posteriormente en 1970 se separó de dicha casa de estudio, siendo autónomo y se le llama desde entonces “SRI International”.

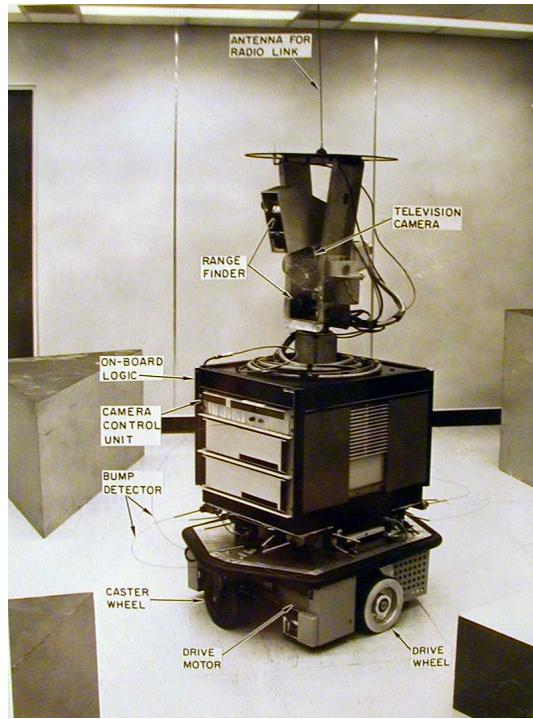


Figura 1.1: Robot Shakey(1968), figura tomada de [1].

Siendo “[Motion Planning](#)²”, desde los años 70 una emergente y crucial área productiva de investigación en la robótica lo explica [Jean-Claude Latombe](#)³ en [8]. El problema básico de la planificación de movimiento es: “Encontrar un camino libre de colisiones para un robot (rígido o articulado) dentro de un espacio con obstáculos rígidos y estáticos”. Este problema que es puramente geométrico puede lucir geométricamente simple; salvo para robots con pocos grados de libertad, es completamente difícil. La planificación de movimiento se convirtió en un tópico de investigación durante la primera década, entre los principales trabajos se tiene el de Nilsson que introdujo el método de visibilidad de grafo (combinado con A*)⁴) para encontrar el caminos más corto para un robot representado por un punto con obstáculos poligonales convirtiéndose en una técnica enormemente popular.

²Planificación de Movimiento, término referido generalmente a los movimientos de un robot en un mundo 2D o 3D con obstáculos

³Renombrado investigador mundial en Motion Planning y es el autor más citado en el campo. Es profesor en la escuela de Ingeniería de la Universidad de Stanford.

⁴A Star, es un algoritmo de Path Planning.

A inicios de los años 80, Lozano-Pérez introdujo el concepto de la “Configuración del Espacio del Robot”, el cual impactó en la Planificación del Movimiento más que alguna otra idea previa. El robot es representado como un punto llamado configuración, en un parámetro que contiene información de los grados de libertad del robot, la configuración del espacio. Los obstáculos son regiones prohibidas, siendo el complemento de éstos el espacio libre y la unión de éstos la configuración del espacio. El [Path Planning](#)⁵ de un robot con dimensiones es reducido al problema de planeamiento de un punto en un espacio que tiene muchas dimensiones tanto como grados de libertad tiene el robot. Posteriormente varias extensiones del problema básico han sido estudiados:

- Obstáculos en movimiento
- Restricciones cinemáticas y dinámicas
- Optimización de trayectorias
- Coordinación de múltiples robots

Asimismo las herramientas clásicas de geometría diferencial fueron usadas para estudiar las múltiples estructuras de una configuración de espacio, junto con sus más específicas propiedades de topología, de geometría y de álgebra. A mediados de los años 80, los planificadores de trayectoria más sofisticados lograron con dificultad calcular rutas de libre colisión para objetos planos trasladando y rotando en espacios de trabajos de 2 dimensiones.

Los conceptos físicos, tales como la fuerza y fricción, fueron elegantemente mapeados en esta representación. La mayoría de estos resultados tomaron un rol crucial en el entendimiento de los problemas de la Planificación de Movimientos, especialmente en sistemas [no holónomos](#)⁶ y de planeamiento óptimo. Es impor-

⁵Planificador de trayectoria, término referido al cálculo de la trayectoria. Esta incluido en Motion Planning.

⁶referido a sistemas que presenta restricciones cinemáticas y/o dinámicas que hacen que los grados de libertad sean dependientes

tante mencionar también que en esta década el problema de la Planificación de Trayectorias atrajo el interés de la comunidad de Ciencias de la Computación, obteniéndose algoritmos que crecían exponencialmente con el número de grados de libertad, la mayoría de estos no fueron implementados, pero ayudaron a calibrar la complejidad del Planeador de Trayectorias y el entendimiento de su naturaleza. Con respecto a las técnicas populares de los 80's, básicamente son 2:

- Aproximación de Descomposición de Celdas, donde el espacio libre es representado por una colección de celdas.
- Campo potencial, basado en atracción y repulsión.

Ambos enfoques trabajan bien en espacios de 2 y 3 dimensiones, pero para mayores grados de libertad no tuvieron el éxito esperado, también a partir de la década de los 90's se empezó a tomar con mayor interés los sistemas no holónomos, es por tal razón que nuevos algoritmos de planificación de tipo aleatorio se desarrollaron tales como PRM y RRT. Desde aquel entonces increíbles progresos han sido desarrollados, lográndose planeadores que resuelven problemas complejos de tipo práctico, algunos con robots de muchos grados de libertad en entornos complejos. Así como el problema básico se han encontrado soluciones usadas en la práctica, pero que aún requieren mayor investigación. Por ejemplo su costo computacional es aún alto, por ende se siguen desarrollando y optimizando dichas técnicas.

Tal como lo indica Jean-Claude Latombe, los cambios en la tecnología, así como nuevas aplicaciones, aumentarán nuevos problemas a investigar. Por ejemplo, los módulos de robots reconfigurables (ver Figura 1.1, tomadas de Biorobotics Laboratory BioRob) debido a sus prestaciones están siendo desarrollados con mayor interés a partir del año 2000. Éstos necesitarán nuevos tipos de planeadores para calcular sus movimientos a reconfigurar.

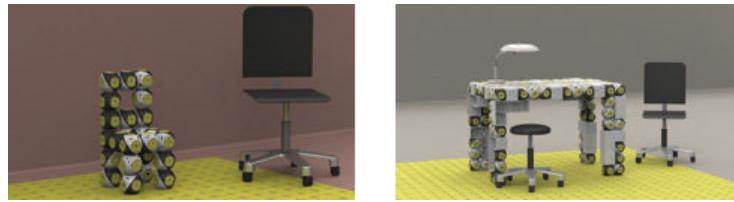


Figura 1.2: Robots reconfigurables.

1.2. Motivación de la Investigación

Varios tipos de máquinas o dispositivos requieren interacción con su entorno, si bien es cierto con los diversos tipos de sensores se puede obtener data del entorno pero esto no es suficiente, ya que se requiere de un algoritmo que interprete dicha data para ejecutar ejecutar acciones pertinentes. El movimiento es una forma de interacción con el medio, ya que en los diversos entornos se presentan obstáculos, restricciones propias del sistema que dificultan en cierto modo el movimiento deseado. El campo de estudio es llamado “Motion Planning”, y lo que se va a enfocar en esta tesis es una parte de ese campo, la planificación de trayectorias en robots móviles.

Lo que motivo la investigación en el presente tema es que las aplicaciones derivadas de su estudio son amplias como por ejemplo en el área de robótica, de manufactura, medicina y animaciones. La contribución al desarrollar este tipo de algoritmos es que se tendría una herramienta de planificación que permita interactuar con nuestro entorno.

1.3. Objetivos

- Desarrollar un algoritmo de planificación de trayectorias de tipo estocástico para robots móviles que les permita navegar en sus respectivos entornos.
- Desarrollar un algoritmo que no este supeditado a los grados de libertad del sistema.

- Generar experiencia académica en planeación de movimiento para posteriores aplicaciones.

1.4. Alcances

El desarrollar algoritmos de Path Planning permite poder realizar aplicaciones futuras no solo enfocadas a robótica móvil o manipuladores sino también a nuevos campos emergentes tales como:

- Diseño para manufactura
- Animación gráfica
- Software de videojuegos
- Cirugía robótica
- Biología computacional

1.5. Limitaciones

El algoritmo heurístico de Path Planning a desarrollar no puede ser aplicado en ambientes o entornos muy dinámicos ya que el costo computacional no lo permite, esto se verá a detalle con las simulaciones a realizar. La presente tesis desarrolla una implementación a pequeña escala con un robot móvil, lo cual servirá para validar el algoritmo en un entorno real.

CAPÍTULO 2

PLANIFICACIÓN DE TRAYECTORIA

La Planificación de Trayectoria ó “Path Planning” se refiere al planeamiento de la trayectoria a seguir para llevar a un sistema desde una configuración inicial a una final o deseada, en el presente caso, dicho sistema es un robot móvil que debe desplazarse por una trayectoria evitando los obstáculos de sus entorno hasta llegar a la posición deseada. Por ende un planeador de trayectoria en una primera instancia reconoce o bosqueja una aproximación del entorno, esto debido a que en algunos sistemas el entorno es conocido a priori; en otros casos el robot va conociendo su entorno de a poco utilizando sensores.

2.1. Definiciones básicas

Los siguientes conceptos son definiciones básicas que se dan en Path Planning y que ayudarán a un mejor entendimiento en la presente tesis, [9] y [10]:

2.1.1. Tiempo

Todos los problemas de planificación implican una secuencia de decisiones que se deben aplicar en el tiempo. El tiempo puede ser modelado explícitamente, como en un problema, tales como conducir un coche tan rápido como sea posible a través de una carrera de obstáculos. Por otra parte, el tiempo puede ser implícito, simplemente refleja el hecho de que las acciones deben seguir en la sucesión, como

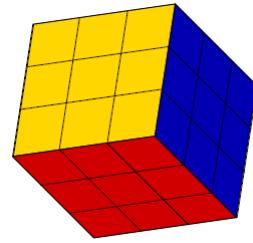


Figura 2.1: Cubo Rubik.

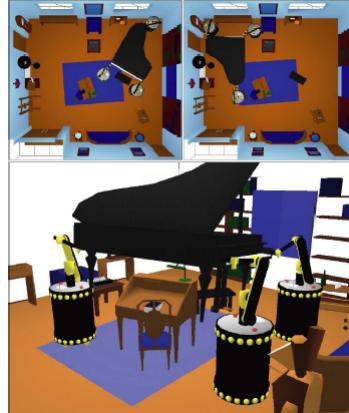


Figura 2.2: Moviendo piano usando robots móviles.

en el caso de resolver el cubo de Rubik (ver Figura 2.1, tomada de [9]). El tiempo en particular no es importante, pero la secuencia adecuada debe mantenerse. Otro ejemplo de tiempo implícito es una solución al problema del Mover el Piano (ver Figura 2.2, tomada de [9]), la solución a mover el piano se puede convertir en una animación con el tiempo, pero la velocidad en particular no se especifica en el plan. Al igual que en el caso de los espacios del estado, el tiempo puede ser discreto o continuo.

2.1.2. Modelo

Un modelo es la representación de un sistema y es usado con la finalidad de responder preguntas mediante análisis y simulación. El modelo que se elige depende del tipo de preguntas que se desee responder, y como tal, existe más de un tipo de modelo para un mismo sistema, con diferentes niveles de fidelidad dependiendo del fenómeno de interés.

2.1.3. Estado

El estado de un sistema es la colección de variables que caracteriza completamente el movimiento de un sistema con el propósito de predecir el movimiento futuro. Al conjunto de todos los estados posibles se le denomina espacio de estados. Un estado puede representarla posición y orientación del robot. Un tema recurrente es que un estado es generalmente representado implícitamente por un algoritmo de planificación.

2.1.4. Acciones

Un plan genera acciones que manipulan el estado. Las acciones de los términos y los operadores son comunes en la inteligencia artificial, en la teoría del control y la robótica, los términos relacionados son los insumos y los controles. En algún lugar de la formulación de la planificación, se debe especificar cómo los cambios de estado cuando las acciones se aplican. Esto puede ser expresado como una función con valores de estado para el caso de tiempo discreto o como una ecuación diferencial ordinaria de tiempo continuo. Para la mayoría de los problemas de movimiento de la planificación, la referencia explícita a tiempo se evita especificar una ruta directa a través de un espacio de estado continuo. Caminos podrían obtenerse como la integral de las ecuaciones diferenciales, pero esto no es necesario. Para algunos problemas, las acciones podrían ser elegidas por la naturaleza, que interfieren con el resultado y se encuentran bajo el control del tomador de decisiones. Esto permite a la incertidumbre en la previsibilidad que se introduce en el problema de planificación.

2.1.5. Estado inicial y final

Un problema de planificación implica usualmente involucra iniciar en algún estado inicial y tratar de llegar a un estado final determinado o cualquier estado de un conjunto de estados meta. Las acciones se seleccionan de manera que trata

de hacer que esto suceda.

2.1.6. Configuración inicial y final

El término configuración para robótica móvil hace referencia a la posición y orientación de un robot.

2.1.7. Entorno

Es el lugar que comprende tanto a los obstáculos como espacios por donde el robot puede desplazarse, para el Path Planning el entorno es el dato principal sobre el cual se realizarán los algoritmos que calculen la trayectoria a seguir por el móvil para alcanzar la posición deseada, ya que en una primera instancia se reconoce o bosqueja una aproximación del entorno. En cuanto a la información referida al entorno se podría clasificar en:

- Entorno conocido, tanto el entorno como la configuración y orientación inicial del robot son conocidas.
- Entorno desconocido, es decir el robot no conoce su entorno a priori como el caso anterior, pero provee sensores que le permiten bosquejar un entorno progresivamente para posteriormente encontrar la ruta deseada.

2.1.8. Criterio

Esto codifica el resultado deseado de un planeador en términos de su espacio estado y acciones ejecutadas. Hay generalmente 2 tipos diferentes de planeamientos basados en el tipo de criterio:

1. **Viabilidad.**- Encontrar un plan que permita llegar al estado deseado, independientemente de su eficiencia.
2. **Optimalidad.**- Encontrar un plan viable que optimice el desempeño en algunos, además de llegar a un estado deseado.

Para la mayoría de problemas la viabilidad es ya suficientemente difícil, lograr la optimización es considerablemente difícil para la mayoría de situaciones. Por lo tanto, gran parte de la atención se centra en la búsqueda de soluciones viables a los problemas, en lugar de soluciones óptimas. La mayoría de la literatura en robótica, teoría de control, y áreas relacionadas se centra en la optimización, pero esto no es necesariamente importante en muchos problemas de interés. En muchas aplicaciones, es difícil formular, incluso un criterio adecuado de optimización. Incluso si un criterio deseable puede formularse, puede ser imposible obtener un algoritmo práctico que compute óptimos planeadores. En tales casos, las soluciones viables son sin duda preferibles a no tener soluciones. Afortunadamente, para muchos algoritmos las soluciones producidas no están tan lejos de las óptimas en la práctica. Esto reduce parte de la motivación para encontrar soluciones óptimas. Para los problemas que involucran incertidumbre probabilística, sin embargo, la optimización surge con mayor frecuencia. Las probabilidades son a menudo utilizadas para obtener el mejor rendimiento en términos de costes previstos. Viabilidad se asocia a menudo con la realización (el desempeño) de un análisis del peor caso de incertidumbres.

Este criterio es tomado no solo en cuenta en lo que concierne, por ejemplo lo que denomina David Lammers en la "Era del error-Tolerancia computacional", [10], en el cual explica que la era perfeccionista del ordenador esta llegando a su fin. En el Simposio Internacional de Electrónica de Baja Potencia y Diseño, según los expertos se refiere a consumo de energía están impulsando la informática hacia una filosofía de diseño en el que los errores están permitidos y son ignorados, o se corregidos sólo cuando sea necesario. Los resultados probabilísticos sustituirán a la forma determinista de procesamiento de datos que ha prevalecido durante el último medio siglo.

Naresh Shanbhag, profesor en el departamento de Ingeniería Eléctrica y Computación de la Universidad de Illions, dice: "Si la aplicación es tal que pe-

queños errores pueden ser tolerados, nosotros permitiríamos que sucedan", agrega "Dependiendo de la aplicación, nosotros mantenemos frecuencias de error bajo un umbral, usando algoritmos o técnicas de circuito". Para muchas aplicaciones tales como procesamiento gráfico o inferencias gráficas desde una gran cantidad de data, los errores en un número razonable no impactan terriblemente en la calidad del resultado. Después de todo, nuestros ojos no pueden notar la presencia de un pixel errado dentro de la mayoría de imágenes.

Otro ejemplo puede darse en los sistemas de control, en los cuales ya hay incertidumbre a las señales a evaluar y buscar una precisión total para nuestra salida deseada, la cual de por si también presenta incertidumbre.

En cuanto a resultados aplicar el criterio expuesto ha logrado en los procesadores un menor consumo de energía; en el procesamiento gráfico y sistemas de control, mayor rapidez.

2.1.9. Plan

En general, un plan establece una estrategia específica o el comportamiento de un tomador de decisiones. Un plan sólo puede especificar una secuencia de acciones a tomar, sin embargo, podría ser más complicado. Si no es posible predecir los estados futuros, a continuación, el plan puede especificar las acciones en función del estado. En este caso, con independencia de los estados futuros, la acción correspondiente se determinará. Utilizando la terminología de otros campos, lo que permite comentarios o planes de reactivos. Incluso podría darse el caso de que el Estado no puede ser medido. En este caso, la acción adecuada debe determinarse a partir de toda la información disponible hasta el momento actual. Esto generalmente se conoce como un estado de información, en la que las acciones de un plan están condicionadas.

2.2. Tipos de Algoritmos de Path Planning

Principalmente desde los años 70 hasta ahora se han desarrollado una gran variedad de algoritmos de Path Planning que acorde a su naturaleza pueden clasificarse en:

2.2.1. Cell decomposition

Hasta hace poco, el método más común en path planning fue basado en la construcción de una descomposición de celdas (la más común es la trapezoidal,[2], ver Figuras 2.3 y 2.4) del espacio libre del robot. Una celda es una región del espacio libre de forma sencilla de tal manera que un camino puede ser construido fácilmente entre 2 configuraciones dentro de la celda. El espacio libre es una colección de celdas, el path planning puede ser reducido a una búsqueda de la representación gráfica la relación adyacente entre celdas. Cell decomposition puede ser exacta o aproximada, la exacta tiene a ser compleja para su implementación e ineficiente por tal razón la aproximada tiene mayor acogida, aunque hay parámetros que tienen que ser regulados como cuando el robot pasa a través de regiones de libre configuración cuyo tamaño es menor que la resolución de la celda, así nos explica Sean Quinlan[11].

Sin embargo cuando la dimensión del espacio libre es alta o cuando la complejidad de la escena es mayor, la representación de las celdas tiende a crecer exponencialmente y el método deja de ser práctico.

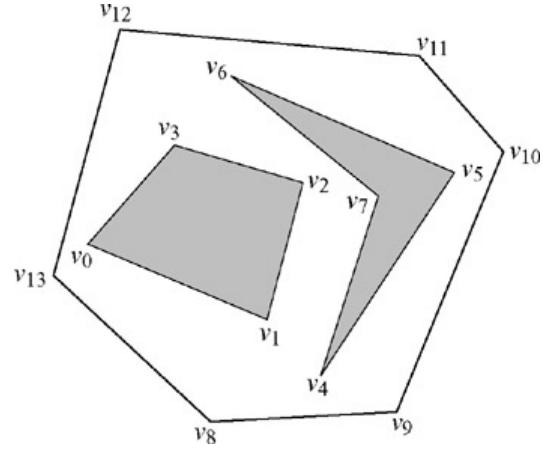


Figura 2.3: Muestreo poligonal de la configuración de espacio. Figura tomada de [2].

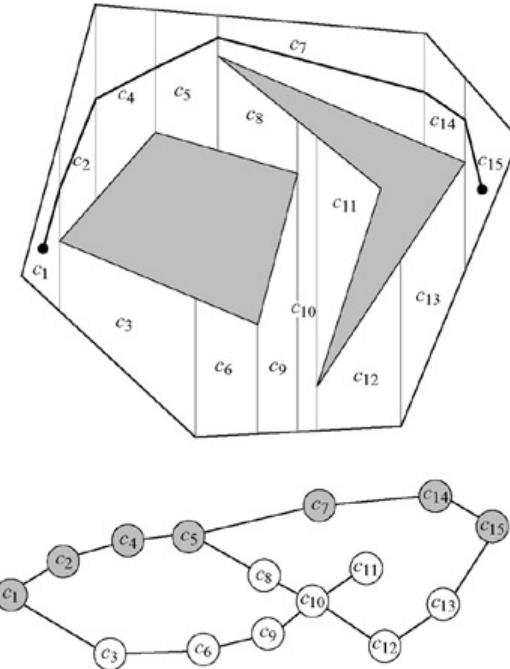


Figura 2.4: Calculo de la ruta. Figura tomada de [2].

2.2.2. Potential fields

El método de los campos potenciales artificiales tiene como principio atraer al robot a la posición deseada y de repelerlo de los obstáculos del entorno de navegación. La gran ventaja de este método es que puede ser implementado para operar en tiempo real y puede ser incorporado directamente en sistema de control

del robot. Para muchas situaciones triviales, este método es muy satisfactorio pero la mayor dificultad se presenta cuando las funciones candidatas afrontan los llamados mínimos locales (ver Figura 3.1) que atascan al robot, de los cuales no es tan fácil escapar, así nos explica Sean Quinlan,[11].

Con respecto a este punto se han desarrollado “Local minima-free potential functions”(también llamadas Funciones de Navegación), sin embargo, los costos de dichas funciones son altos en configuraciones espaciales de grandes dimensiones y no han obtenido grandes resultados en robots de muchos grados de libertad. Otra alternativa propuesta era almacenar las probabilidades de las vecindades de las configuraciones sin caer en mínimos locales, con ello se obtuvieron buenos resultados hasta robots de 6 grados de libertad, pero conforme aumentaban los grados de libertad esta técnica se volvía impráctica como lo explica Latombe,[12].

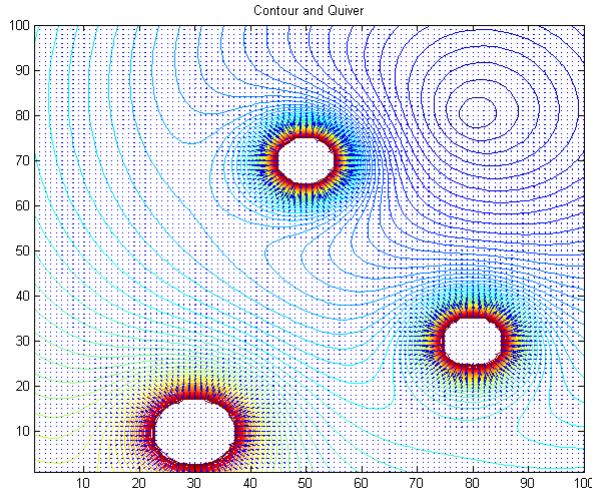


Figura 2.5: Aplicación de campos potenciales en Path Planning. Figura tomada de [3]

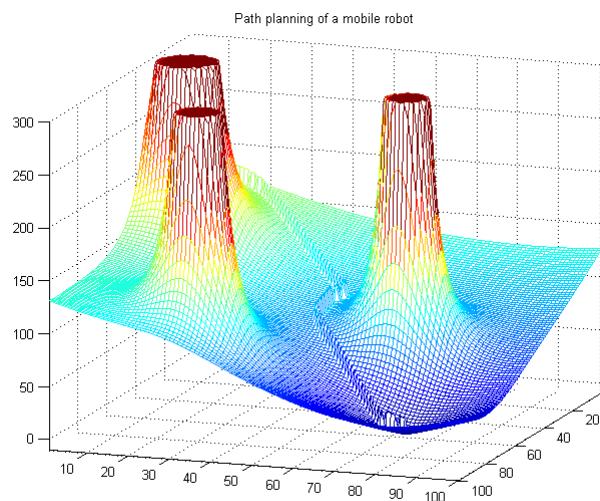


Figura 2.6: Vista del planeador. Figura tomada de [3].

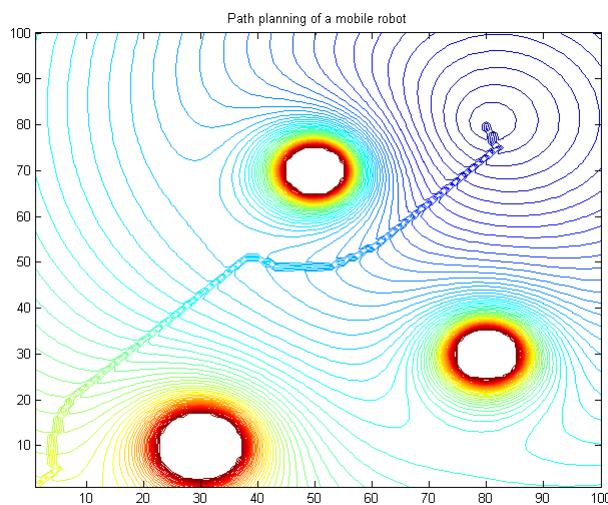


Figura 2.7: Encontrando la ruta usando gradiente. Figura tomada de [3].

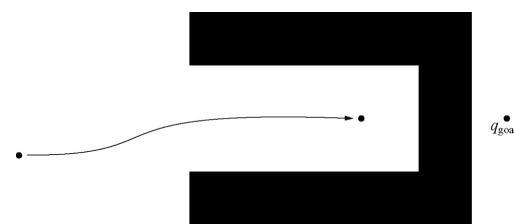


Figura 2.8: Ejemplo de mínimo local. Figura tomada de [2].

2.2.3. Roadmaps

Se enfoca el espacio libre del robot como la colección de caminos libres de colisión conectados a lo largo de dicho espacio. Con dicho conjunto, se construye la ruta desde el punto inicial al punto deseado que está en alguna parte del roadmap. Para un ambiente estático el roadmap es construido una vez para resolver múltiples problemas de planeamiento. Las muchas variaciones de roadmap se dan principalmente por el enfoque diverso en el método de construcción de dicho roadmap, estas variaciones incluyen [11]:

- Visibilidad de grafos
- Diagramas de Voronoi
- Roadmaps aleatorios

Los 3 primeros funcionan para robots de 2-3 grados de libertad pero no resultan muy adecuados para mayor grados de libertad tal como lo explica Latombe[12], que propone el uso de técnicas aleatorias, siendo los procesos estocásticos experimentalmente los que han tenido buenos resultados en robots complejos de muchos grados de libertad.

Se construye un grafo o “roadmap” (hoja de ruta) que representa la configuración libre de obstáculos del entorno; con el grafo ya definido se procede a encontrar la ruta que une la configuración inicial con la deseada, la cual debe ser modificada para cumplir con las restricciones no holonómicas del robot y la naturaleza del ambiente (dinámica o estática). La desventaja que posee es un alto costo computacional con respecto a otros métodos.

2.3. Algoritmo de path planning RRT

En la presente tesis se desarrollará el algoritmo RRT (Rapidly Random Exploring Trees[13]), que es pertenece al grupo de Roadmaps ya que explora el

entorno del robot para encontrar la ruta deseada. A su vez es un algoritmo de tipo estocástico.

2.4. Metodología de Trabajo

Básicamente consta de 3 parte(que son explicadas a detalle en CAPITULO IMPLEMENTACION):

1. Reconocimiento de Entorno, que será a priori para el robot, es decir se conocerán los obstáculos y configuraciones libres, así como la configuración inicial del robot.
2. Ejecutar el Planeador de trayectorias, cuyos datos son el entorno, la configuración inicial del robot y la configuración deseada.
3. Supervisar que el robot siga la trayectoria de referencia mediante un controlador que tiene un sensor que calcula la posición y orientación del robot.

CAPÍTULO 3

ALGORITMO RRT

El algoritmo RRT es una estructura de datos que está diseñado para explorar de manera eficiente espacios no convexos de grandes dimensiones (ver Figura 3.1, tomada de [2]). El algoritmo se construye de forma incremental, reduciendo rápidamente la distancia de espera de un punto elegido al azar en el árbol. RRT es especialmente adecuado para problemas de planificación de ruta de acceso que implican los obstáculos y restricciones diferenciales. RRT puede ser considerado como una técnica para generar las trayectorias a lazo abierto para sistemas no lineales con restricciones de estado.

Hay que notar que de por sí solo, RRT es incapaz de resolver un problema de path planning. En consecuencia, puede considerarse como un componente ha ser incorporado en el desarrollo de una variedad de algoritmos de planificación diferentes.



Figura 3.1: Naturaleza del Algoritmo RRT.

3.1. Elementos

El algoritmo RRT original se basa en la construcción de un árbol de configuraciones que crece buscando a partir de un punto origen. Para entender el algoritmo se usará la nomenclatura expresada en el cuadro 3.1:

Elemento	Descripción
C	es el conjunto de todas las configuraciones posibles del entorno, es decir, los obstáculos y las configuraciones libres.
C_{free}	Subconjunto de C , configuraciones libres de obstáculos existentes.
R	Indicador de ponderación de proximidad al punto deseado. La distancia euclídea es la más utilizada.
q_{init}	Configuración inicial
q_{fin}	Configuración que se desea alcanzar
q_{rand}	Configuración aleatoria dentro del espacio C_{free}
q_{near}	Es la configuración más próxima a q_{rand} , en el sentido denido por R , de entre las existentes en un árbol. Se evalúa con el indicador de ponderación.
q_{new}	Configuración a añadir al árbol.
e	Longitud de segmento de crecimiento. Geométricamente, es la distancia entre un punto del árbol y el siguiente con el que está conectado.
$Arbol$	Estructura de datos.

Cuadro 3.1: Elementos de una RRT

Algorithm 3.1 Algoritmo RRT

```

Función: RRT( $q_{ini}$ ,  $K_{max}$ ,  $e$ )
1.  $Arbol[0] = q_{init};$ 
2. for  $K = 1$  to  $K_{max}$ 
3.  $q_{rand} = \text{Configuración_Aleatoria}();$ 
4.  $q_{near} = \text{Elemento_más_cercano}(Arbol, q_{rand})$ 
5.  $q_{new} = \text{Nuevo_elemento}(q_{rand}, q_{near}, e)$ 
6. end for
7. Devuelve  $Arbol;$ 

Función: Nuevo_elemento( $q_{rand}$ ,  $q_{near}$ ,  $e$ )
1.  $u_{q_{near}-q_{rand}} = (q_{rand} - q_{near}) / \|q_{rand} - q_{near}\|;$ 
2.  $q_{new} = q_{near} + e.u_{q_{near}-q_{rand}};$ 
3. Devuelve  $q_{new};$ 

```

3.2. Pseudocódigo

El algoritmo viene expresado por:

El pseudocódigo de RRT es explicado como sigue:

1. Se inicializa la estructura de datos llamada $Arbol$ con la configuración inicial q_{init} (ver Figura 3.2)
2. Se inicia un bucle con un máximo número de iteraciones K_{max}
3. Se asigna a q_{rand} una configuración aleatoria del espacio C_{free} (ver Figura 3.3)
4. Se asigna a q_{near} el elemento más cercano de $Arbol$ a q_{rand} (ver Figura 3.4)
5. Se asigna a q_{new} el nuevo elemento de $Arbol$, este elemento se calcula como la suma vectorial del producto del vector unitario (desde q_{near} a q_{rand}) con la longitud de crecimiento definida (e) y q_{near} (ver Figura 3.5).
6. Se verifica la condición del bucle.
7. Devolución de la estructura $Arbol$

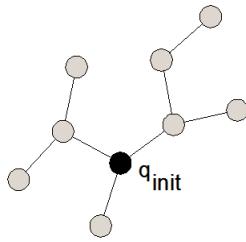


Figura 3.2: Árbol de exploración.

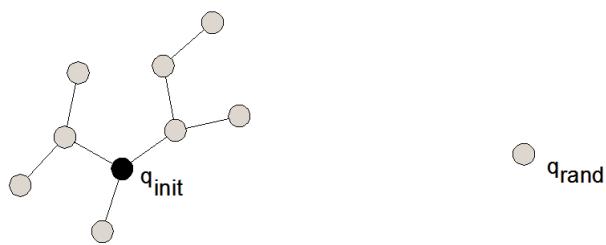


Figura 3.3: Seleccionando un punto aleatorio.

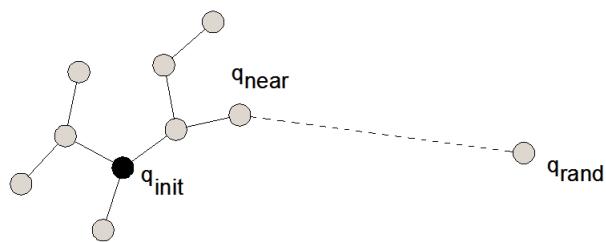


Figura 3.4: Calculando el punto el elemento más cercano del árbol al punto aleatorio.

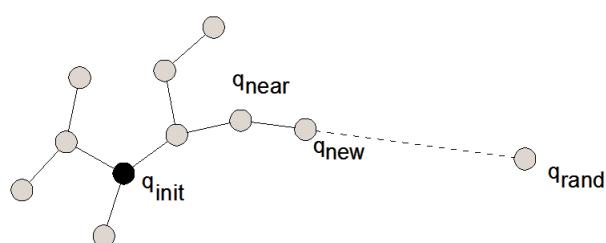


Figura 3.5: Adicionando nuevo elemento al árbol explorador.

Algorithm 3.2 Algoritmo RRT Bidireccional Básica

```

Función: RRT_BB( $q_{ini}$ ,  $q_{goal}$ ,  $K_{max}$ , e)
1.  $Arbol\_A[0] = q_{init}$ ;  $Arbol\_B[0] = q_{goal}$ ;  $K = 1$ ;
2. while ( $K < K_{max}$ || state='Alcanzado')
3.    $q_{rand} = \text{Configuración\_Aleatoria}()$ ;
4.   if ( Extiende( $Arbol\_A$ ,  $q_{rand}$ ) ≠ 'rechazado' )
5.     a=Extiende( $Arbol\_A$ ,  $q_{rand}$ ); b=Extiende( $Arbol\_B$ ,  $q_{rand}$ );
6.     if (a=='alcanzado' y b=='alcanzado')
7.       Devuelve Ruta;
8.     end if;
9.   end if;
10.  Intercambiar ( $Arbol\_A$ ,  $Arbol\_B$ );
11.   $K = K + 1$ ;
12. end while;

Función: Extiende( $Arbol$ ,  $q_{rand}$ )
1.  $q_{near} = \text{Vecino\_mas\_proximo}(Arbol, q_{rand})$ ;
2.  $q_{new} = \text{Nuevo\_elemento}(q_{rand}, q_{near})$ ;
3. if(  $q_{new} \in \text{Configuracion\_libre}$ )
4.    $Arbol = [Arbol; q_{new}]$  //se añade nuevo vértice
5.   if (  $\text{abs}(q_{new}-q_{rand}) < \text{tolerancia}$  ) Devuelve 'alcanzado';
6.   else Devuelve 'avanzado';
7.   endif
8. else Devuelve 'rechazado';
9. endif

Función: Nuevo_elemento( $q_{rand}$ ,  $q_{near}$ , e)
1.  $u_{q_{near}-q_{rand}} = (q_{rand} - q_{near}) / \|q_{rand} - q_{near}\|$ ;
2.  $q_{new} = q_{near} + e.u_{q_{near}-q_{rand}}$ ;
3. Devuelve  $q_{new}$ ;

```

3.3. Algoritmos evolucionados de RRT

Estos algoritmos permiten que el RRT pueda encontrar la ruta entre 2 configuraciones. Entre las principales destacan:

3.3.1. RRT Bidireccional Básica

Consiste en conectar los árboles exploradores que surgen desde el punto inicial y final. El algoritmo esta dado por:

El pseudocódigo es explicado como sigue:

1. Se inicializan los árboles exploradores con las configuraciones inicial (q_{init}) y final (q_{goal}), es decir con 2 estructuras de datos tipo *Arbol* (ver Figura 3.2)
2. Se inicia un bucle
3. Se asigna a q_{rand} una configuración aleatoria del espacio C_{free} (ver Figura 3.3)
4. Si el nuevo elemento del *Arbol_A* con respecto a q_{rand} está dentro de configuración C_{free} se procede al paso 5, caso contrario al paso 10
5. Se calculan las nuevas configuraciones de los árboles *Arbol_A* y *Arbol_B* con respecto a q_{rand} si es que las hay mediante la función Extiende.
6. Si en ambos árboles se obtiene ‘alcanzado’ ir al paso 7, caso contrario ir al paso 9.
7. Si las nuevas configuraciones de los 2 árboles alcanzan al punto q_{rand} , se procede a calcular y devolver la ruta contenida en la estructura *Arbol*, así como salir del bucle.
8. Se verifica la condición del bucle, si se termina se va al paso 7, caso contrario al paso 3.
9. Se intercambian los árboles.
10. Se aumenta la cuenta de la variable K .
11. Se verifica la condición del bucle.

3.3.2. RRT Ext-Ext

Es una mejora del algoritmo RRT Bidireccinal básico, el algoritmo viene expresado como:

Algorithm 3.3 Algoritmo RRT Ext-Ext

```

Función: RRT_Ext_Ext(qini, qgoal, Kmax, e)
1. Arbol_A[0] = qinit; Arbol_B[0] = qgoal; K = 1;
2. while (K < Kmax || state='Alcanzado')
3.     qrand = Configuración_Aleatoria();
4.     if ( Extiende(Arbol_A, qrand)≠'rechazado')
5.         a=Extiende(Arbol_A, qrand);
6.         qnew =last_element(Arbol_A);
7.         b=Extiende(Arbol_B, qnew);
8.         if (a=='alcanzado' y b=='alcanzado')
9.             Devuelve Ruta;
10.        end if;
11.    end if;
10.    Intercambiar (Arbol_A, Arbol_B);
11.    K = K + 1;
12. end while;

Función: Extiende(Arbol, qrand)
1. qnear= Vecino_mas_proximo(Arbol, qrand);
2. qnew= Nuevo_elemento(qrand, qnear);
3. if( qnew ∈ Configuracion_libre)
4.     Arbol=[Arbol; qnew] //se añade nuevo vértice
5.     if ( abs(qnew=qrand)< tolerancia) Devuelve 'alcanzado';
6.     else Devuelve 'avanzado';
7.     endif
8. else Devuelve 'rechazado';
9. endif

Función: Nuevo_elemento(qrand, qnear, e)
1. uqnear-qrand = (qrand - qnear) / ||qrand - qnear|| ;
2. qnew = qnear + e.uqnear-qrand;
3. Devuelve qnew;

```

El pseudocódigo es explicado como sigue:

1. Se inicializan los árboles exploradores con las configuraciones inicial (q_{init}) y final (q_{goal}), es decir con 2 estructuras de datos tipo *Arbol* (ver Figura 3.6)
2. Se tiene bucle con un máximo número de iteraciones K_{max} .
3. Se asigna a q_{rand} una configuración aleatoria del espacio C_{free} (ver Figura 3.3).
4. Se calcula el nuevo elemento del *Arbol_A*, si este elemento (q_{new}) no pertenece a la región de los obstáculos, se procede al paso 5, caso contrario al paso 2.
5. Se calcula el nuevo elemento del *Arbol_B*, teniendo como objetivo el q_{new} de *Arbol_A* del paso 4.
6. Se verifica la condición de '**alcanzado**' para ambos árboles, si se cumple se devuelve la ruta y se finaliza el bucle, caso contrario se procede al paso 7.
7. Se intercambian los árboles.
8. Se aumenta la cuenta de la variable K .
9. Se va al paso 2.

3.3.3. RRT Ext-Con

Tiene el siguiente principio:

Algorithm 3.4 Algoritmo RRT Ext-Con

```

Función: RRT_Ext_Con( $q_{ini}$ ,  $q_{goal}$ ,  $K_{max}$ ,  $e$ )
1.  $Arbol\_A[0] = q_{init}$ ;  $Arbol\_B[0] = q_{goal}$ ;  $K = 1$ ;
2. for  $K = 1$  to  $K_{max}$ 
3.    $q_{rand} = \text{Configuración\_Aleatoria}()$ ;
4.   if ( $\text{Extiende}(Arbol\_A, q_{rand}) \neq \text{'rechazado'}$ )
5.      $a = \text{Extiende}(Arbol\_A, q_{rand})$ ;
6.      $q_{new} = \text{last\_element}(Arbol\_A)$ ;
7.      $b = \text{Conecta}(Arbol\_B, q_{new})$ ;
8.     if ( $a == \text{'alcanzado'}$  y  $b == \text{'alcanzado'}$ )
9.       Devuelve Ruta;
10.    end if;
11.   end if;
10.  Intercambiar ( $Arbol\_A$ ,  $Arbol\_B$ );
11.   $K = K + 1$ ;
12. endfor;

Función:  $\text{Extiende}(Arbol, q_{rand})$ 
1.  $q_{near} = \text{Vecino\_mas\_proximo}(Arbol, q_{rand})$ ;
2.  $q_{new} = \text{Nuevo\_elemento}(q_{rand}, q_{near})$ ;
3. if ( $q_{new} \in \text{Configuracion\_libre}$ )
4.    $Arbol = [Arbol; q_{new}]$  //se añade nuevo vértice
5.   if ( $\text{abs}(q_{new} - q_{rand}) < \text{tolerancia}$ ) Devuelve 'alcanzado';
6.   else Devuelve 'avanzado';
7.   endif
8. else Devuelve 'rechazado';
9. endif

Función:  $\text{Conecta}(Arbol, q)$ 
1. do { $S = \text{Extiende}(Arbol, q)$ ; }
2. while ( $S = \text{'avanzado'}$ );
3. Devuelve  $S$ ;

Función:  $\text{Nuevo\_elemento}(q_{rand}, q_{near}, e)$ 
1.  $u_{q_{near}-q_{rand}} = (q_{rand} - q_{near}) / \|q_{rand} - q_{near}\|$ ;
2.  $q_{new} = q_{near} + e.u_{q_{near}-q_{rand}}$ ;
3. Devuelve  $q_{new}$ ;

```

El pseudocódigo es explicado como sigue:

1. Se inicializan los árboles exploradores con las configuraciones inicial (q_{init}) y final (q_{goal}), es decir con 2 estructuras de datos tipo $Arbol$ (ver Figura 3.6)

2. Se tiene un bucle con un máximo número de iteraciones K_{max} .
3. Se asigna a q_{rand} una configuración aleatoria del espacio C_{free} (ver Figuras 3.7 y 3.8).
4. Se calcula el nuevo elemento del $Arbol_A$, si este elemento (q_{new}) no pertenece a la región de los obstáculos, se procede al paso 5, caso contrario al paso 2 (ver Figura 3.9 y 3.10).
5. Se calcula el nuevo elemento o nuevos elementos del $Arbol_B$ (ver función **Conecta** y Figuras 3.10, 3.11 y 3.12), teniendo como objetivo el q_{new} de $Arbol_A$ del paso 4. Se verifica la condición de '**alcanzado**', si se cumple se devuelve la ruta (ver Figura 3.13) y se sale del bucle, caso contrario se procede al paso 6.
6. Se intercambian los árboles.
7. Se aumenta la cuenta de la variable K .
8. Se va al paso 2.

Los pasos son:



Figura 3.6: Conección de árboles exploradores.

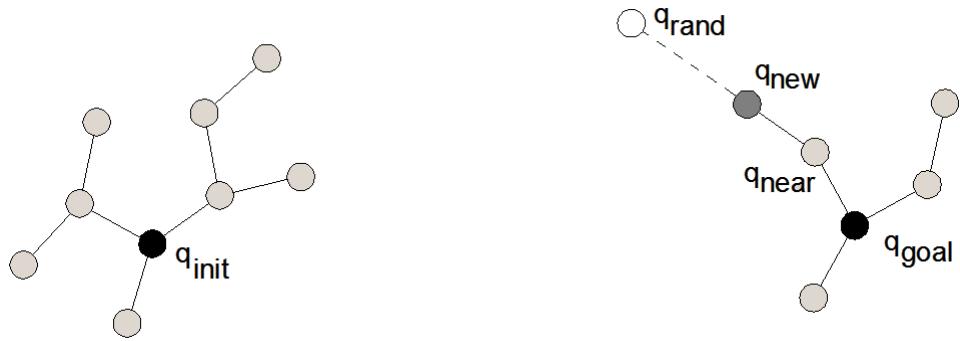


Figura 3.7: Un árbol genera un nuevo elemento (círculo gris oscuro).

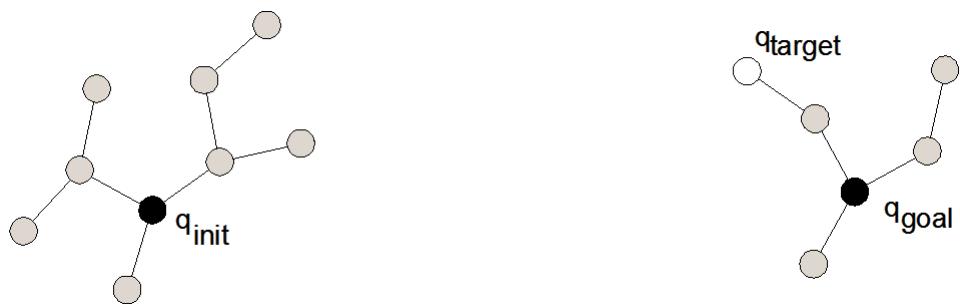


Figura 3.8: El nuevo elemento del árbol se convierte en el objetivo (q_{target}) del otro árbol.

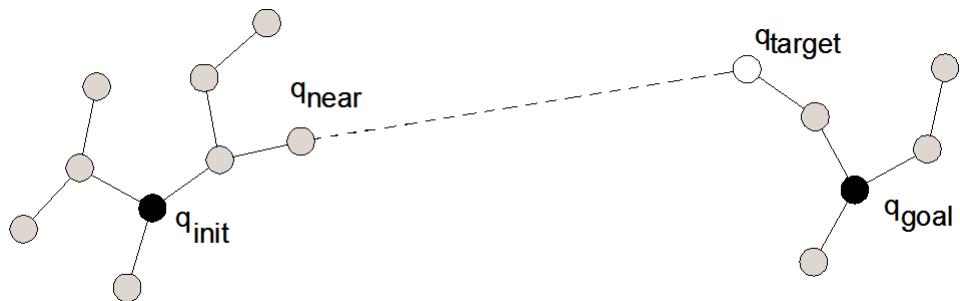


Figura 3.9: Calculando el elemento más cercano (q_{near}) a q_{target} .

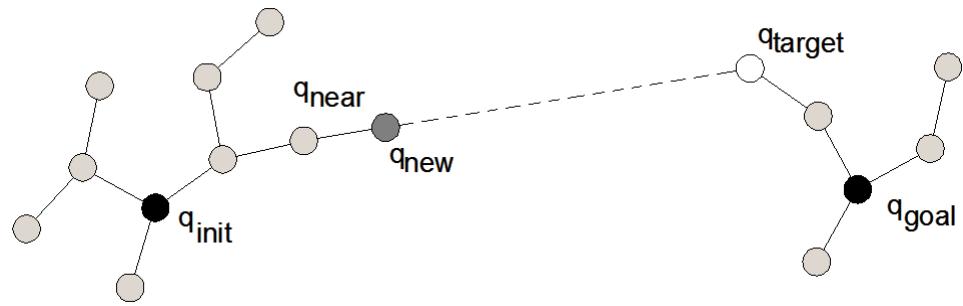


Figura 3.10: Cálculo de nuevo elemento (q_{new}) del árbol de q_{init} .

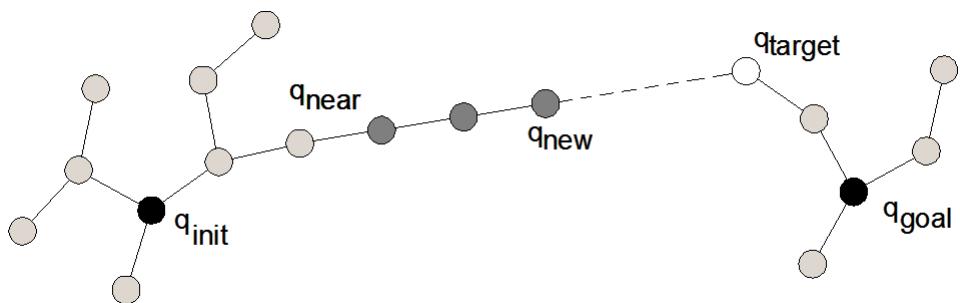


Figura 3.11: Adición consecutiva de elementos hasta no colisión.

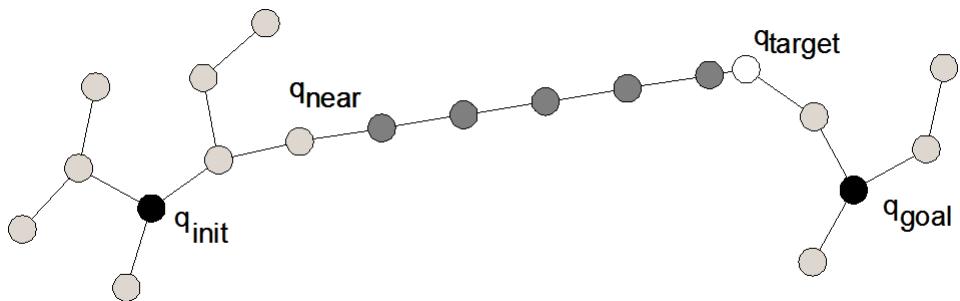


Figura 3.12: Conexión final del crecimiento.

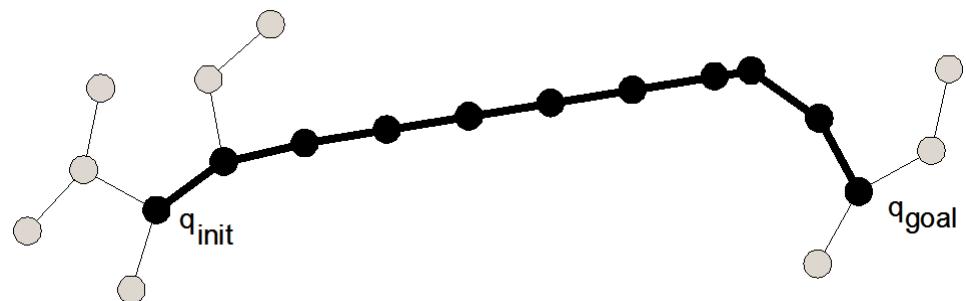


Figura 3.13: Tracking del camino de conexión.

3.4. Sistemas no holónomos

El algoritmo RRT se puede adaptar para sistemas que presenten..., tal es el caso de sistemas que son representados de la siguiente manera:

$$\dot{x} = f(x, u)$$

Algorithm 3.5 Algoritmo RRT para sistemas no holónomos.

```

Función: RRT_non_hol(qini, Kmax, Δt)
1. Arbol = qinit; K = 1;
2. for 1 to Kmax
3.   qrand = Configuración_Aleatoria();
4.   a=Extiende(Arbol, qrand);
5.   if (a=='alcanzado')
6.     Devuelve Ruta;
7.   endif;
8.   endif;
9. K = K + 1;
10. endfor;

Función: Extiende(Arbol, qrand)
1. qnear= Vecino_mas_proximo(Arbol, qrand);
2. u=seleccionar_entrada(qrand, qnear);
3. qnew= Nuevo_elemento(qnear, u,Δt);
4. if( qnew ∈ Configuracion_libre)
5.   Arbol=[Arbol; qnew] //se añade nuevo vértice
6.   if ( abs(qnew-qrand)< tolerancia) Devuelve 'alcanzado';
7.   else Devuelve 'avanzado';
8.   endif
9. else Devuelve 'rechazado';
10. endif

Función: Nuevo_elemento(qnear, u,Δt);
1. qnew=qnear+dot{x}(u)*Δt;
2. Devuelve qnew;
```

Para el cálculo del nuevo elemento del árbol es recomendable utilizar el método de Runge-Kutta 4 para obtener mayor presición en sistemas de mayor número de grados de libertad, en el presente algoritmo se ha utilizado la aproximación de tangente.

CAPÍTULO 4

SIMULACIÓN DE RESULTADOS Y PERFORMANCE DE EVALUACIÓN

Para las simulaciones realizadas para el algoritmo de Path Planning se ha utilizado el software Matlab, aunque puede utilizarse también el opensoftware Octave ya que se han codificado las funciones sin utilizar toolbox especiales. Con respecto al equipo de procesamiento se ha utilizado una Corel 5 con un clock de 2.4 Ghz.

4.1. Simulación del algoritmo RRT en sistemas holónomos

4.1.1. Algoritmo RRT

Tal como se había explicado en el capítulo anterior el algoritmo RRT es un algoritmo de exploración de las configuraciones libres (zonas de color blanco, ver Figura 4.1 y Figura 4.2). Para el caso de la Figura 4.1 se tiene un espacio de 40mx40m con una longitud de segmento de crecimiento de 5cm y se puede apreciar que al aumentar el número de iteraciones el árbol explorador con origen en (0,0) tiende a crecer a lo largo del espacio libre, visualizándose así la naturaleza exploradora del algoritmo. Este mismo comportamiento se puede apreciar en la Figura 4.2, a través de un espacio que presenta obstáculos (representados por polígonos de color negro).

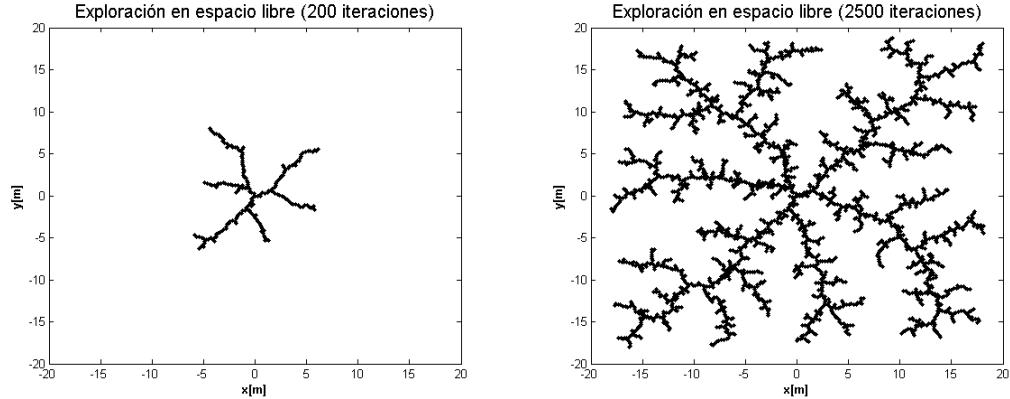


Figura 4.1: Algoritmo RRT explorando en configuración libre de obstáculos.

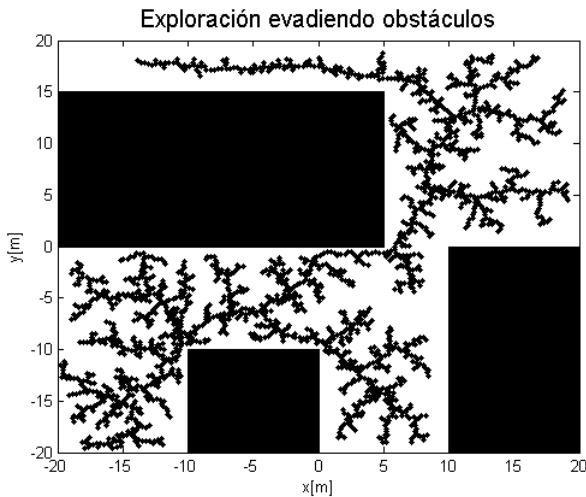


Figura 4.2: Algoritmo RRT explorando a traves de obstáculos.

4.1.2. Algoritmo RRT bidireccional

En este algoritmo la idea es encontrar un punto común de crecimiento entre los 2 árboles exploradores (ver Figura 4.3), siendo los orígenes de dichos árboles los puntos A y B. Aparentemente se puede apreciar muchos posibles puntos comunes de crecimiento, lo que permite intuir que el procedimiento puede mejorarse ya que el criterio de parada es que ambos árboles colisionen en el preciso instante que se expanden hacia el mismo punto.

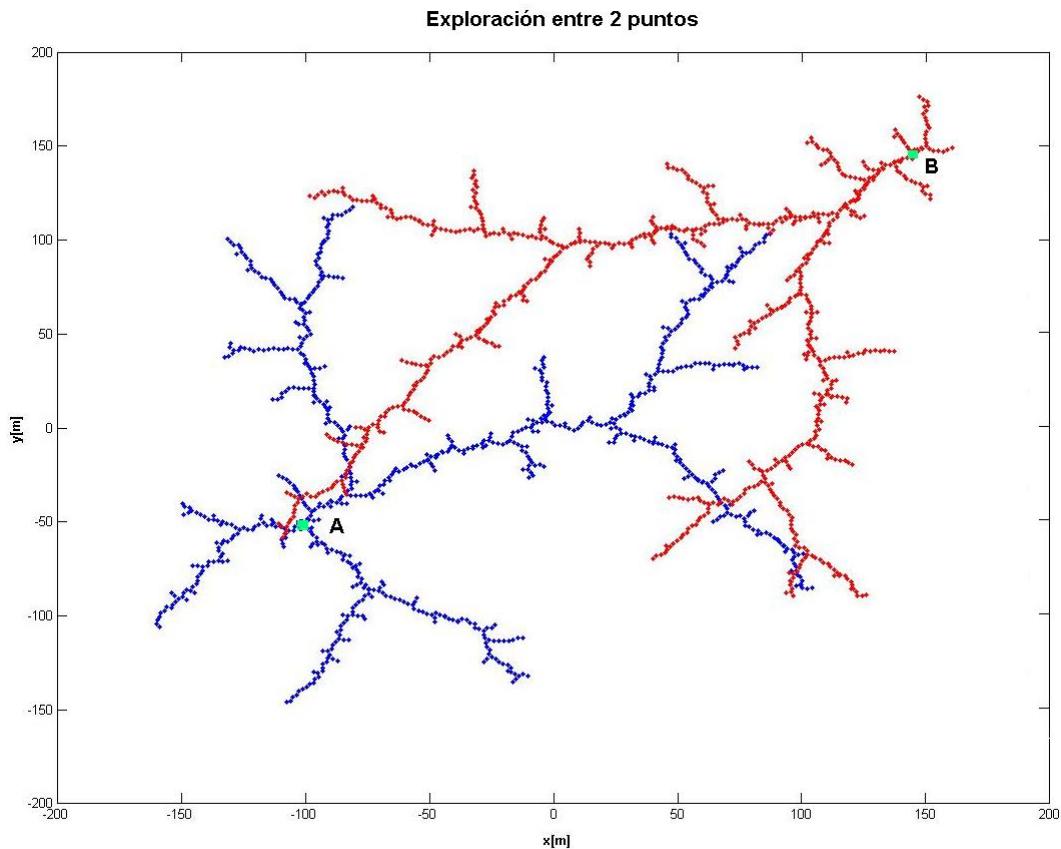


Figura 4.3: RRT bidireccional básico en espacio libre.

4.1.3. Algoritmo RRT Ext-Ext

Es una mejora del algoritmo RRT bidireccional, explicado en el capítulo 3. En la Figura 4.4, el algoritmo RRT-Ext-Ext genera 385 iteraciones, en cambio el otro 781 iteraciones. A su vez el camino obtenido con el RRT-Ext-Ext tiende a ser una línea, como debería ser.

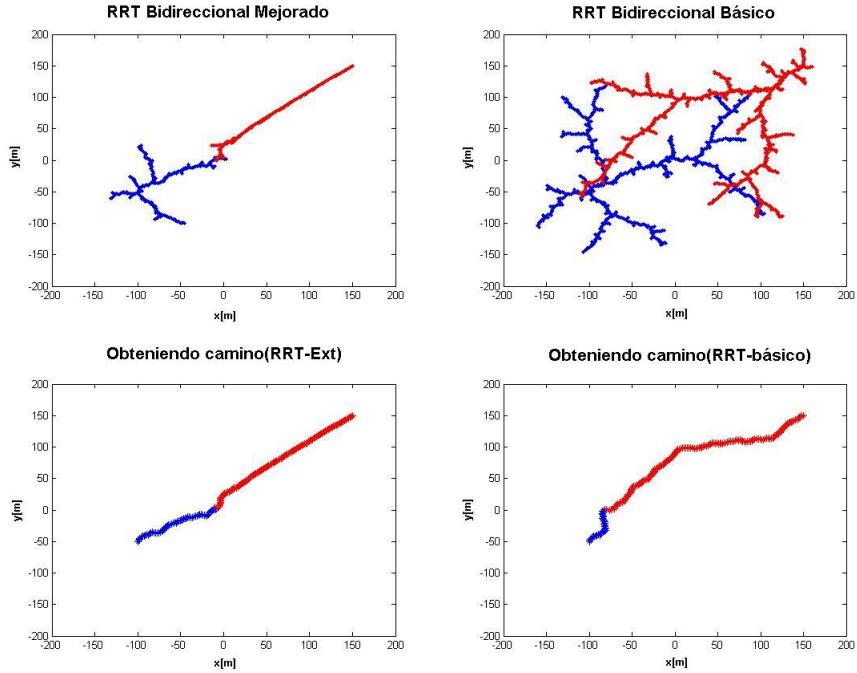


Figura 4.4: A la izquierda algoritmo RRT-Ext-Ext, a la derecha el algoritmo RRT básico bidireccional. Comparación entre ambas variaciones de RRT.

4.1.4. Algoritmo RRT Ext-Con

Es también otro mejora del algoritmo RRT bidireccional, el crecimiento del árbol es variable ya que la expansión del segmento de crecimiento se da consecutivamente hasta ser interrumpida por una configuración prohibida(obstáculo), ver Figuras 4.5, 4.6 y 4.7.

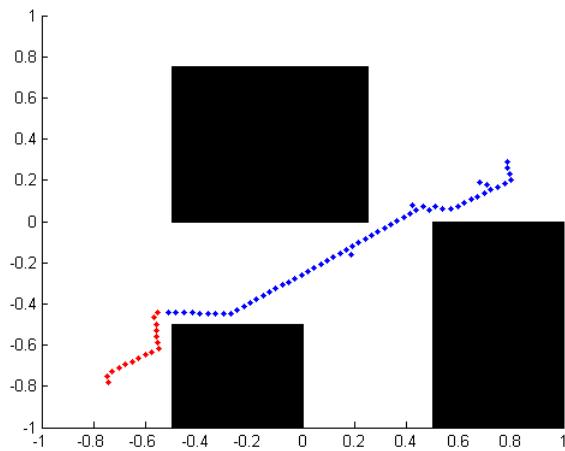


Figura 4.5: Árboles exploradores en RRT-Ext-Con

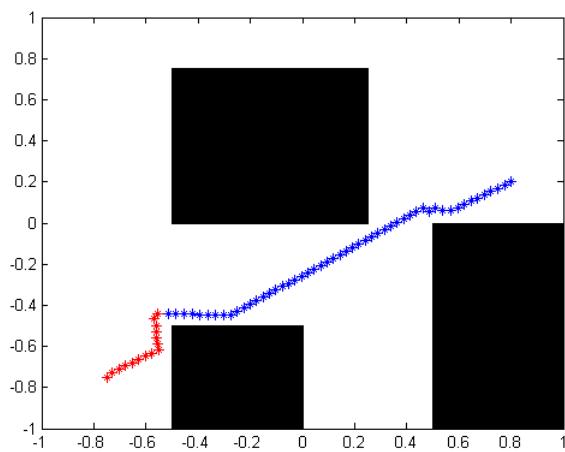


Figura 4.6: Tracking de árboles que forman la ruta entre los puntos deseados.

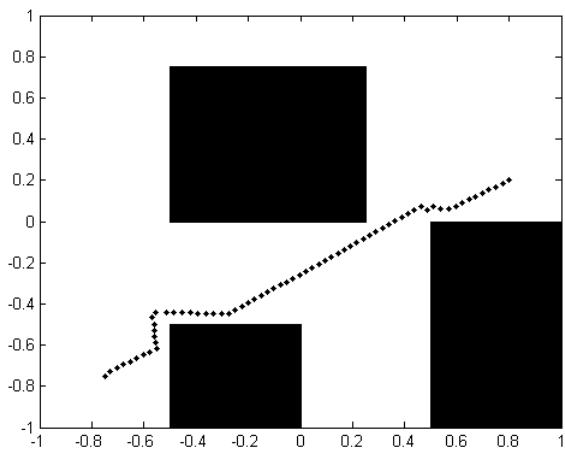


Figura 4.7: Ruta entre los puntos.

Se presenta una simulación considerando las dimensiones del robot moway con un rectángulo, es decir ya no se considera al robot como sólo un punto, sino que también se toma en cuenta sus dimensiones (ver Figuras 4.8, 4.9, 4.10 y 4.11).

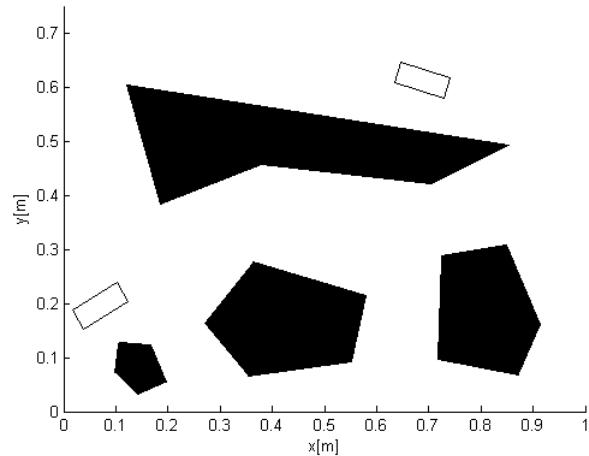


Figura 4.8: Configuración inicial y final del robot Moway así como su entorno con obstáculos.

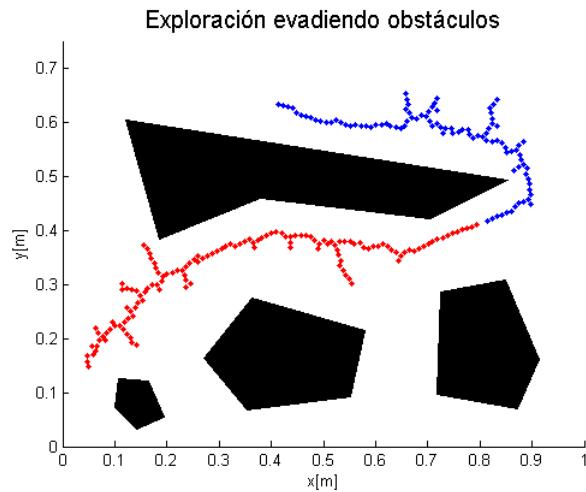


Figura 4.9: Árboles exploradores entre las configuraciones deseadas.

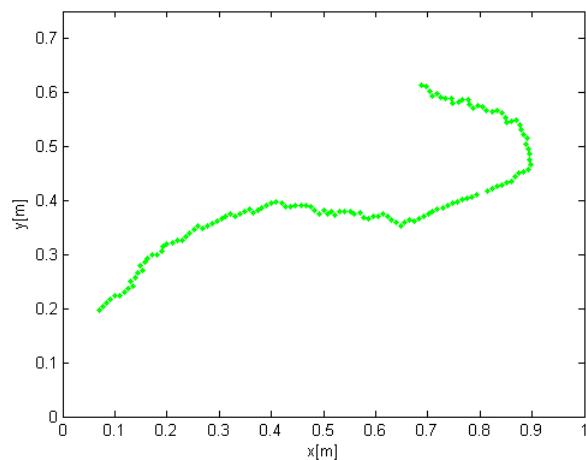


Figura 4.10: Tracking de la ruta entre las configuraciones inicial y final.

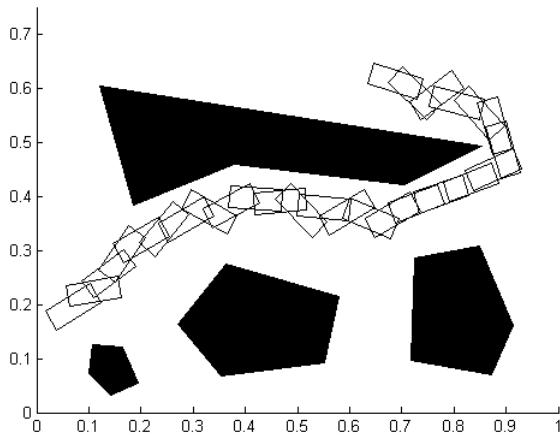


Figura 4.11: Secuencia de movimiento entre las configuraciones inicial y final en el entorno.

4.2. Simulación del algoritmo RRT en sistemas no holónomos

En este caso, se toma en consideración las restricciones que presentan los sistemas, como por ejemplo un vehículo de configuración ackerman no puede realizar una maniobra con radio de giro cero, en cambio una plataforma omnidireccional si puede realizarla. Por ello se hace necesario el modelar el sistema para poder determinar a priori que restricciones ha de presentar el sistema.

4.2.1. Modelamiento Cinemático del Sistema

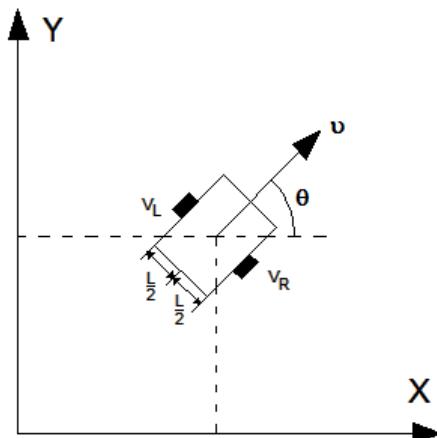


Figura 4.12: Ruta entre los puntos.

La cinemática del robot Moway esta expresada por:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (4.1)$$

Pero para el presente caso se ha de expresar en función de las velocidades de cada rueda:

$$v = \frac{V_R + V_L}{L} \quad (4.2)$$

$$\omega = \frac{V_R - V_L}{2} \quad (4.3)$$

Reemplazando 4.2 y 4.3 en 4.1, se tiene:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos(\theta)}{2} & \frac{\cos(\theta)}{2} \\ \frac{\sin(\theta)}{2} & \frac{\sin(\theta)}{2} \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \cdot \begin{bmatrix} V_R \\ V_L \end{bmatrix} \quad (4.4)$$

Simulando el sistema para que parta de la configuración 1 a la configuración 2, tenemos:

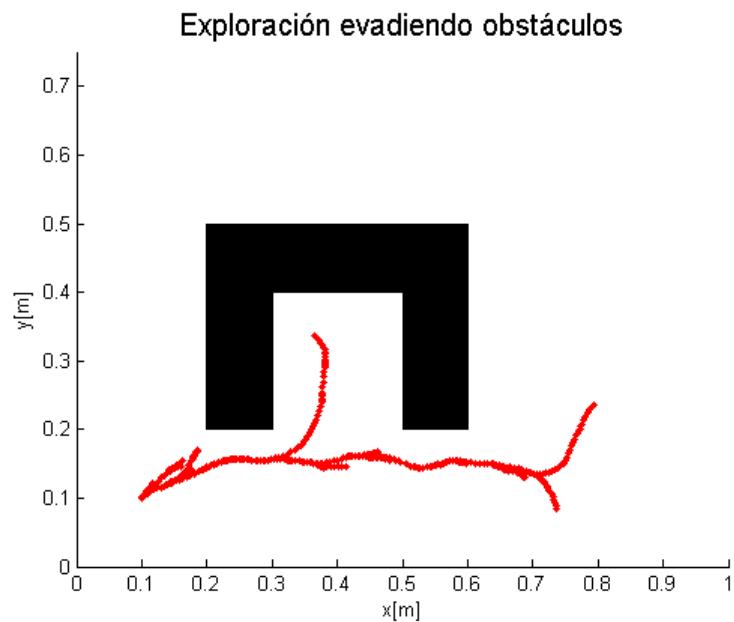


Figura 4.13: Ruta entre las configuraciones inicial y final en el entorno dado.

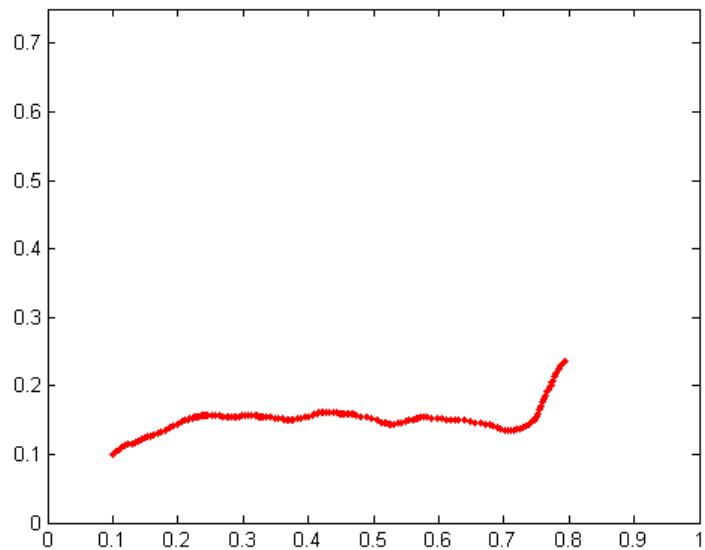


Figura 4.14: Ruta a seguir por el robot Moway.

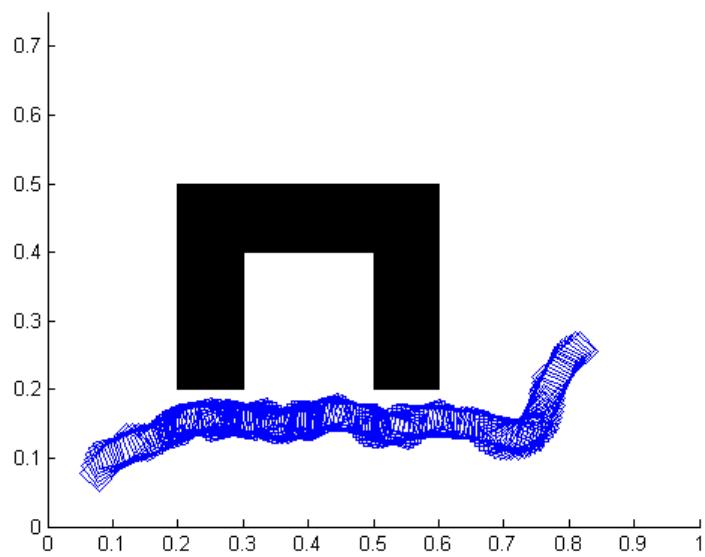


Figura 4.15: Tracking del movimiento del robot Moway desde la configuración inicial a la final.

CAPÍTULO 5

IMPLEMENTACIÓN DE PLATAFORMA E INTERFAZ

Para la implementación física del planeador de trayectorias se ha desarrollado una interfaz de usuario, mediante la cual se ha de ejecutar tareas a especificar por un algoritmo dado. Asimismo, han realizado las pruebas físicas sobre la siguiente plataforma de prueba que se detalla a continuación (ver figuras 5.1, 5.2 y 5.3):

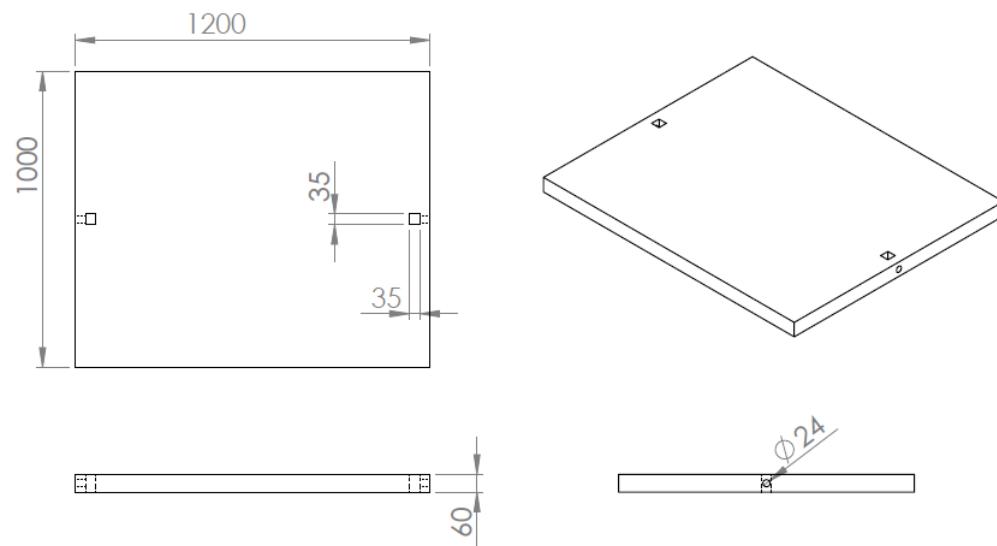


Figura 5.1: Plano de base de la plataforma. Unidades milímetros.

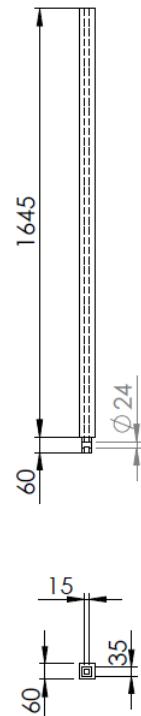


Figura 5.2: Plano de columna de soporte. Unidades milímetros.

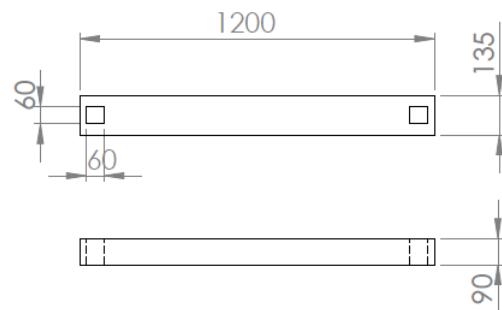


Figura 5.3: Plano de soporte de cámara. Unidades milímetros.

La plataforma descrita forma parte de una pequeña planta, cuya conformación está dada por (ver Figura 5.4):

- Robot Moway
- Cámara Hallion
- Plataforma

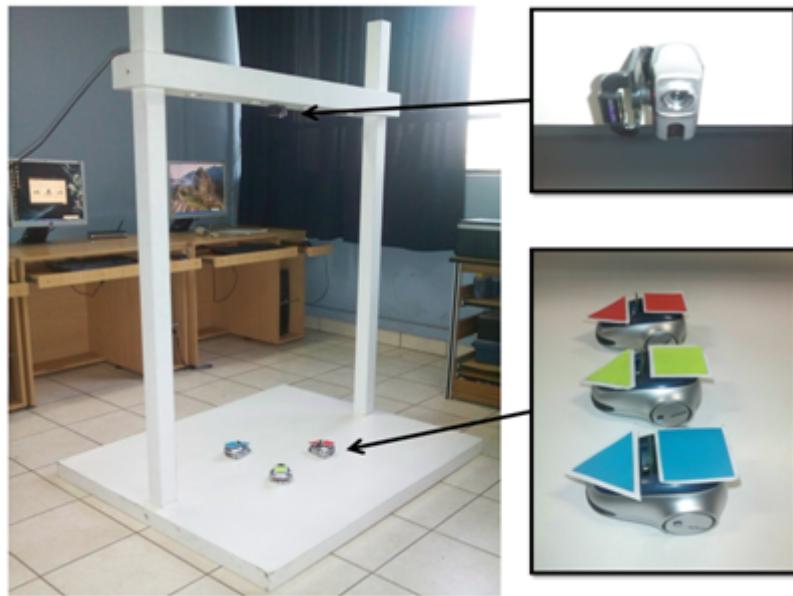


Figura 5.4: Planta de Prueba

En las pruebas se ha trabajado con el robot móvil Moway (ver apéndice B) de la compañía MiniRobots¹ por las prestaciones que tiene. La Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería a través de INIFIM² cuenta con 3 robots Moway de iguales características, por tal razón se ha considerado colocarles un color característico para poder diferenciarlos durante el trabajo con los mismos.

Por tal razón surge la necesidad de crear una interfaz que permita controlar a dichos robots para así poder gestionarlos con flujos de datos de otros programas o algoritmos que se desarrolle. Si bien es cierto que el robot Moway cuenta con un software propio para adiestramiento y manipulación inalámbrica (mediante tecnología de radiofrecuencia), dicho software no permite trasladar data o interactuar con otros programas así como tampoco permite enviar órdenes de movimiento a varios robots Moway. En este capítulo se explicará cómo se realizó dicha interfaz que tiene como principales características los siguientes puntos:

¹Es la única empresa fabricante del robot Moway con sede central en Barakaldo-España. La dirección web de su producto es www.moway-robot.com

²Instituto Nacional de Investigación de la Facultad de Ingeniería Mecánica.

- Comunicación inalámbrica con varios robots Moway
- Posibilidad de flujo de data con otros programas
- Reconocimiento de robots mediante visión artificial
- Indicadores de posición y orientación para 3 robots
- Planificador de trayectorias

Para lograr lo mencionado anteriormente se ha utilizado una cámara como sensor, cuya data es procesada por la interfaz para efectos de cálculo de trayectorias. La interfaz puede controlar no sólo un robot moway, ésta ha sido diseñada para reconocer hasta 3 robots, sin embargo, esto puede escalarse aumentando el número de pantallas y configurando los filtros.

5.1. Consideraciones

Las simulaciones del planeador de trayectoria fueron realizadas en el software Matlab, para el desarrollo del sistema de visión y la interfaz de comunicación con el robot Moway se empleó la plataforma de Visual C#³(debido a que los controladores del robot moway están diseñados en ese lenguaje) y las librerías de visión Emgu[14] (wrapper⁴ de OpenCV⁵ para C#); la integración entre Matlab y Visual C# se dio mediante el uso de dll⁶ s; sin embargo, no se descarta una posterior codificación completa del algoritmo de Path Planning en C#.

³Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET, similar al de Java aunque incluye mejoras derivadas de otros lenguajes (entre ellos Delphi).

⁴Envoltura, para este caso permite en C# accesar a las funciones contenidas en la librería OpenCV.

⁵Open Source Computer Vision: es una librería con funciones para el desarrollo de visión computacional, escrita inicialmente en C y posteriormente en C++

⁶Dynamic Library Link (biblioteca de enlace dinámico), es la extensión con el que se identifican a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo. Esta denominación es exclusiva a los sistemas operativos Windows.

5.2. Funcionamiento de la interfaz

Para el presente caso el procedimiento a seguir es:

1. Obtener la data del entorno mediante la cámara, así como la posición inicial del robot Moway.
2. Calcular la trayectoria a seguir por el Algoritmo de Path Planning, teniendo en cuenta la información recogida por la cámara (entorno del robot y configuración inicial) así como la configuración deseada.
3. Inicio del controlador de trayectoria, el cual transmite las órdenes necesarias para que el robot realice la trayectoria calculada (ver Figura 5.5).
4. Calcular y enviar las órdenes de movimiento a través de la RF-USB desde la PC al robot.
5. Obtener la posición y orientación del móvil, esta data es enviada al controlador de trayectoria descrito en el paso 3 para corroborar si se sigue la ruta calculada por el planificador de trayectorias.

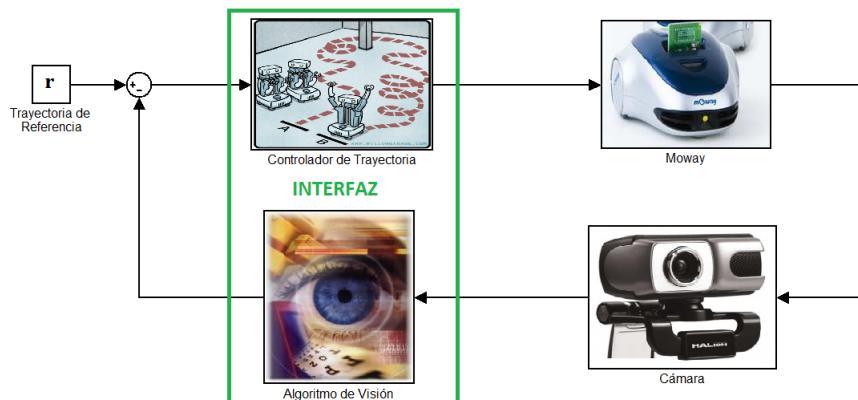


Figura 5.5: Controlador retroalimentado de Trayectoria

5.3. Características de la interfaz

La presente interfaz agrega visión computacional a la interfaz ejemplo de Moway, así como también un planeador de trayectorias que es el objetivo de la presente tesis (ver figura 5.6).

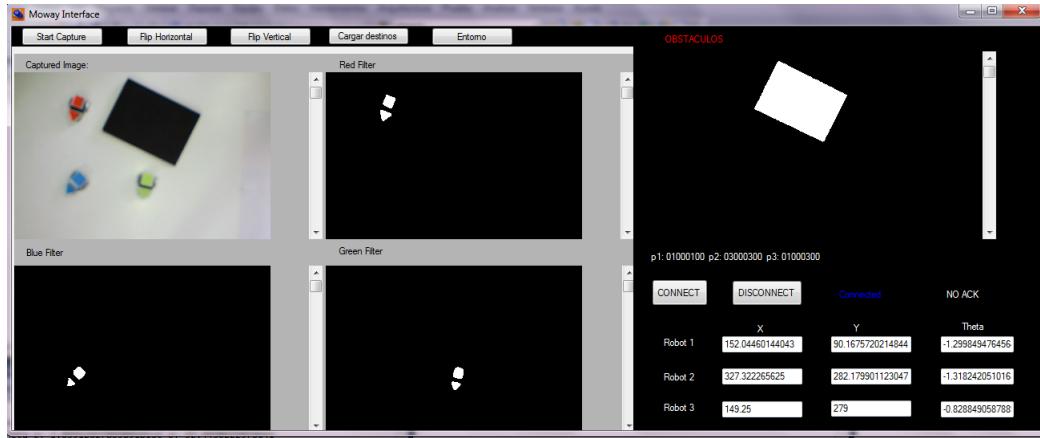


Figura 5.6: Controlador retroalimentado de Trayectoria

La interfaz de usuario desarrollada presenta los siguientes elementos visuales:

- 5 pantallas de procesamiento de imágenes: 1 de lo captado por la cámara, 3 de reconocimiento de los móviles y 1 del entorno.
- Botón de **Flip Vertical**, que permite voltear la imagen verticalmente.
- Botón de **Flip Horizontal**, que permite voltear la imagen horizontalmente.
- Botón **Start Capture**, se inicia la captura y procesamiento de los frames (cuadros ó fotogramas) captados por la cámara.
- Botón **Cargar destinos**, la trayectoria calculada es cargada a la interfaz que tiene como objetivo que el robot la realice.
- Botón **Entorno**, captura los obstáculos presentes en el entorno.

- Indicadores de posición y orientación de los robots, siendo la posición expresada en pixeles y la orientación de los robots en radianes (ver Figura 5.7).

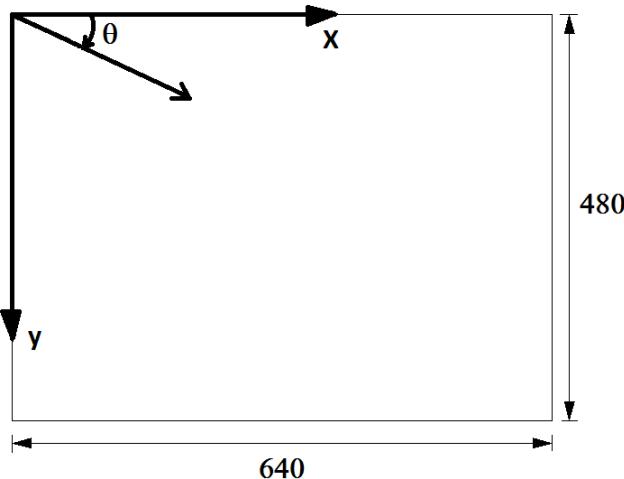


Figura 5.7: Sistema de referencia para la interfaz.

Secuencia de la Interfaz

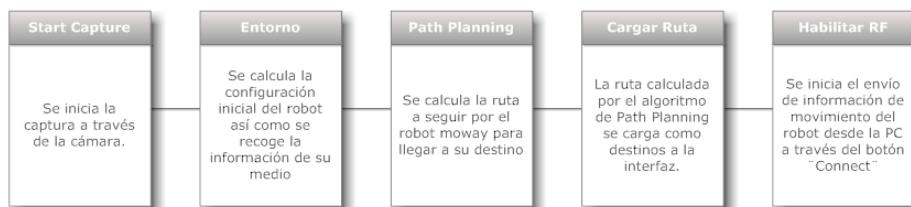


Figura 5.8: Secuencia de manipulación de la interfaz.

Lo expresado anteriormente ha sido codificado en un máquina de estados (ver Figura 5.9), la cual es ejecutada por la interfaz mediante un timer configurado cada 10 ms. Los estados de dicha máquina de estados están descrita por:

Máquina de Estados

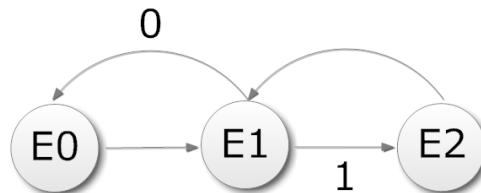


Figura 5.9: Secuencia de la máquina de estados

- **E0:** Estado inicial
- **E1:** Estado de visión. En este estado se ejecuta la función ProcessFrame, la cual calcula la posición y orientación de los robots Moway así como también muestra en las pantallas el procesamiento realizado por la interfaz para entender el funcionamiento de la parte de visión artificial.
- **E2:** Control y envío RF. Se carga la trayectoria de referencia, se calcula el movimiento del robot y se envía dicha orden mediante inalámbricamente.

Para que se pase del estado E1 al estado E2 se requiere que flagstarcamera este en uno lógico, es decir, que la cámara este capturando el entorno. En caso que no haya conexión de la cámara se retornará al estado E0.

5.4. Componentes de la Interfaz de Usuario

5.4.1. Sección de Comunicación Inalámbrica

En esta sección se da la comunicación inalámbrica entre la PC y el robot Moway utilizando el módulo BZI-RF2GH4 de radiofrecuencia cuyas características técnicas se pueden apreciar en el cuadro 5.1.

Parámetro	Valor	Unidad
Tensión mínima de alimentación	1.9	V
Tensión máxima de alimentación	3.6	V
Potencia máxima de salida	0	dBm
Velocidad máxima de transmisión	2000	Kbps
Corriente en modo transmisión @ 0dbm potencia de salida	11.3	mA
Corriente en modo recepción @ 2000kbps	12.3	mA
Corriente en modo Power Down	900	nA
Frecuencia máxima del bus SPI	8	Mhz
Rango de temperatura	-40 a +85	°C

Cuadro 5.1: Parámetros principales del módulo BZI-RF2GH4, datos tomados de [7].

Las características principales del módulo BZI-RF2GH4 son:

- Bajo consumo.
- Frecuencia de trabajo de 2.4GHz
- Potencia de emisión entre -18 y 0 dBm
- Velocidad de transmisión entre 1 y 2 Mbps
- 128 canales de transmisión seleccionables por el bus SPI.

Con el objetivo de facilitar el manejo del módulo, BIZINTEK INNOVAS.L. ha desarrollado unas librerías tanto en ensamblador como en C que simplifican y acortan el tiempo de desarrollo de aplicaciones inalámbricas con estos módulos. A su vez para desarrollar una aplicación mediante una interfaz desde la PC se utilizan las dlls lib_nrf241u1_RF y lib_prog_mow2, las cuales están compiladas para C#. Desde dicha interfaz se ha de recibir la ruta calculada por un controlador de trayectoria para y se han de enviar órdenes de movimiento para realizar dicha trayectoria, a su vez se esta considerando retroalimentar la posición y orientación del robot Moway (ver Figura 5.5).

Para la presente aplicación se han considerado las siguientes órdenes de movimiento para el robot moway (considerando que siempre avanza):

- Adelante
- Giro a la derecha
- Giro a la izquierda

Estas órdenes de movimientos se han definido debido a la restricción física que presente el robot en cuya hoja técnica se describe el comportamiento en lo que respecta a la rapidez de sus ruedas en función al ciclo de trabajo(duty cycle) del motor DC que genera el movimiento rotacional. Ello se representa a través de la siguiente gráfica:

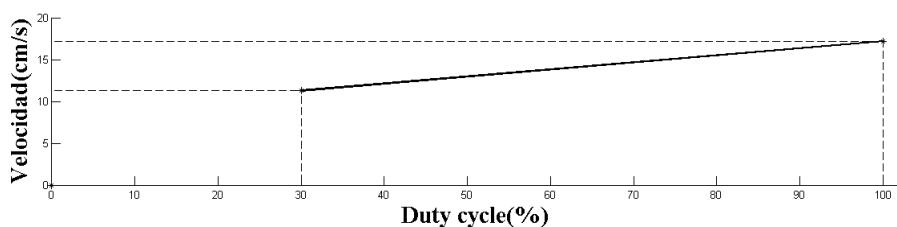


Figura 5.10: Rapidez de la rueda del robot Moway respecto al Duty cycle del motor DC que genera el movimiento rotacional.

De la Figura 5.10 se puede apreciar que no contamos con el manejo completo de los duty cycle de las ruedas debido a que la rapidez de la rueda no está definida en el intervalo 1-29. A su vez, la rapidez al 30 % de duty cycle con el 100 % son próximos, por tal razón en vista de que no hay gran diferencia entre dichas rapideces se ha considerado trabajar con una representación equivalente (ver Figura 5.11), en ella se ha decidido trabajar con la rapidez máxima y la cero.

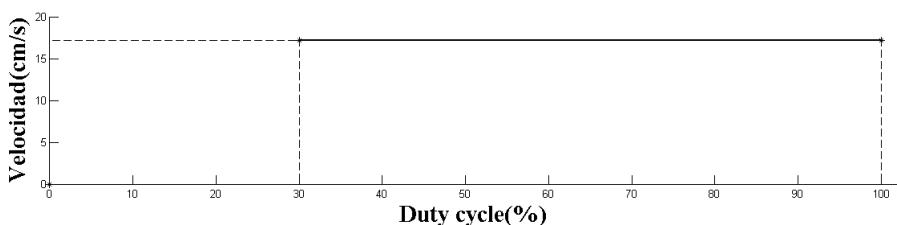


Figura 5.11: Comportamiento de la rapidez respecto al duty cycle del robot Moway

Orden de movimiento	Número
Giro a la derecha	1
Giro a la izquierda	2
Adelante	3
Alto	4

Cuadro 5.2: Asignación de números para las órdenes de movimiento

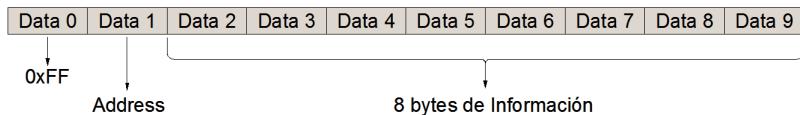


Figura 5.12: Estructura del paquete total enviado a través del protocolo de la RF del robot Moway

Una vez aclarado el criterio tomado para los tipos de movimientos a utilizar en el robot, se procederá a explicar como se ha de enviar las órdenes de movimiento para que el robot moway las cumpla. Como se había mencionado anteriormente se ha tomado 4 movimientos principales del robot asignándoles números, ver el cuadro 5.2.

Con respecto al protocolo de comunicación que usa el robot Moway, se designa a todo el paquete de información transmitido por la radiofrecuencia (entiéndase por cabeceras y la información propiamente dicha) como “Data”. La estructura de “Data” se muestra a continuación:

Como se puede apreciar de la Figura 5.12, el paquete “Data” es en si una estructura de 10 bytes, siendo los 2 primeros de configuración y teniendo 8 bytes para envío de información desde la PC al robot Moway en cuestión. El primer byte de “Data”(**Data 0**) está seteado a una constante en hexadecimal 0xFF que es 255. El segundo byte de “Data”(**Data 1**) contiene el Address o dirección del dispositivo(robot Moway o PC) que recibirá la información. Finalmente desde **Data 2 a Data 9** disponemos de estos bytes para la información que deseamos transmitir entre 2 puntos (sea PC-Moway ó Moway-Moway).

Si bien es cierto que se puede configurar el envío y recepción para la comu-

nicación, el éxito de esta depende de un protocolo de comunicación el cual esta contenido en las librerías desarrolladas por Moway. Por ejemplo se describe que al enviar data de un punto a otro hay un tiempo de espera maximo de 16ms, tiempo en cual se espera recibir un flag o indicador del éxito de la comunicación caso contrario se puede retransmitir la información. A su vez es importante indicar que para el desarrollo del presente proyecto se ha considerado el problema de la interferencia de otras ondas inalámbricas como las de WIFI, las cuales entorpecen el accionar de las RF de los robots Moway, si éstas son bastantes y si su intensidad es alta, para ello una recomendación del fabricante es el cambio de canal para evitar interferencias. Ello se puede apreciar en la siguiente figura:

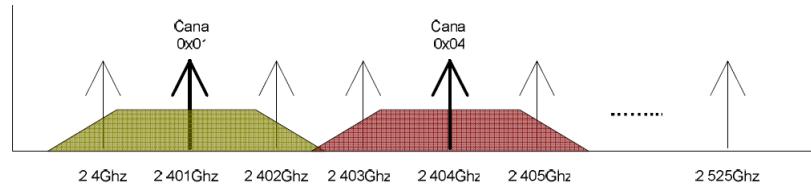


Figura 5.13: CANALES DE RF MOWAY PARA EVITAR COLISIONES CON OTRAS ONDAS

5.4.2. Sección de Visión Artificial

La visión artificial, también conocida como visión por computador, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un dispositivo como un computador para que interprete una escena o las características de una imagen (en este caso sería de los frames captados por la cámara). Para el presente caso la data requerida es:

- El entorno del robot: obstáculos y espacios libres del entorno, esta información tendrá el formato de una matriz binaria, la cual es el dato del mapa para el desarrollo del Path Planning.
- Posición y orientación del robot Moway: se necesita de algún sensor que verifique si el robot sigue la trayectoria calculada para corregir su movimiento.

Con ello se tendría un sistema retroalimentado.

Como se había mencionado anteriormente en la interfaz una función denominada ProcessFrame realiza tanto el cálculo de la posición y orientación de los robots Moway así como el reconocimiento de las características del entorno. Ahora se procederá a explicar lo referente dicha función (ver Figura 5.14 y 5.15):

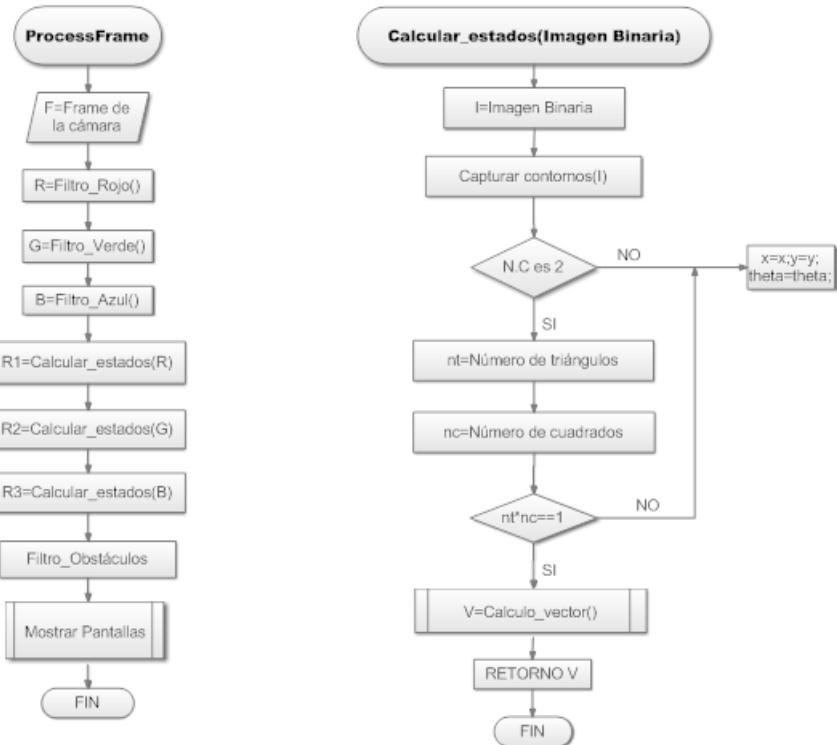


Figura 5.14: Función ProcessFrame

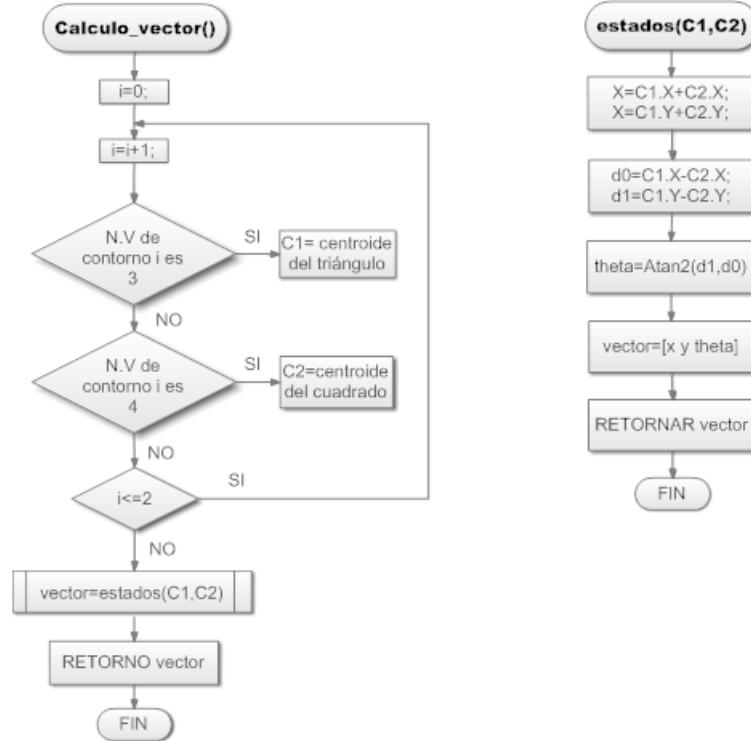
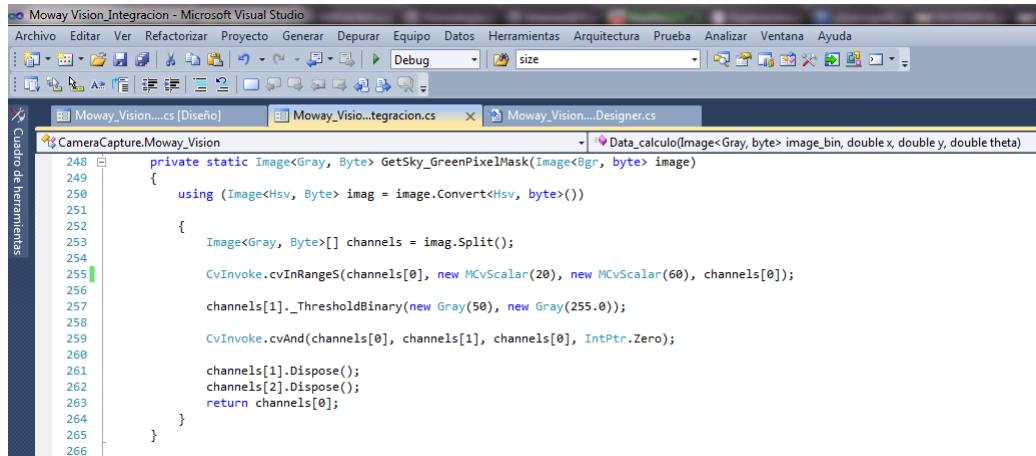


Figura 5.15: Rutina Calculo_vector perteneciente a ProcessFrame

En primer lugar se toma el frame de la cámara, el cual pasa a ser filtrado por 3 filtros de reconocimiento de los robots moway y 1 filtro para el entorno. Los filtros de reconocimiento de los Moway toman como entrada el frame de la cámara y entregan una imagen binaria, la cual se da como resultado tras filtrar un color en interés. En el presente caso son los colores rojo, verde y azul comprendidos en los siguientes rangos (ver tabla 5.3 y Figura 5.16) del sistema HSV (ver apéndice C):

	Robot 1 (Rojo)	Robot 2 (Verde)	Robot 3 (Azul)
H (Hue)	$H < 20; H > 160$	$20 < H < 60$	$100 < H < 120$
S (Saturation)	$S > 10$	$S > 50$	$S > 90$

Cuadro 5.3: Rango de colores en interés en el sistema HSV.



```

Moway Vision_Integracion - Microsoft Visual Studio
Archivo Editar Ver Refactorizar Proyecto Generar Depurar Equipo Datos Herramientas Arquitectura Prueba Analizar Ventana Ayuda
Debug size
Moway_Vision...cs [Diseño] Moway_Vision...tegracion.cs Moway_Vision...Designer.cs
Cuadro de herramientas
248     private static Image<Gray, Byte> GetSky_GreenPixelMask(Image<Bgr, byte> image)
249     {
250         using (Image<Hsv, Byte> imag = image.Convert<Hsv, byte>())
251         {
252             Image<Gray, Byte>[] channels = imag.Split();
253             CvInvoke.cvInRangeS(channels[0], new MCvScalar(20), new MCvScalar(60), channels[0]);
254             channels[1]._ThresholdBinary(new Gray(50), new Gray(255.0));
255             CvInvoke.cvAnd(channels[0], channels[1], channels[0], IntPtr.Zero);
256             channels[1].Dispose();
257             channels[2].Dispose();
258             return channels[0];
259         }
260     }
261 }
262
263
264
265
266

```

Figura 5.16: Filtro de obstáculos verdes, codificación en lenguaje C#.

Los filtros de colores se evalúan en el sistema HSV debido a que este sistema no es tan susceptible al brillo como en el caso del sistema de color RGB (ver Figura 5.17) . Esto queda corroborado con el histograma de dicha imagen (ver Figura 5.18), en dicho histograma se aprecia picos en la zona de luminosidad.

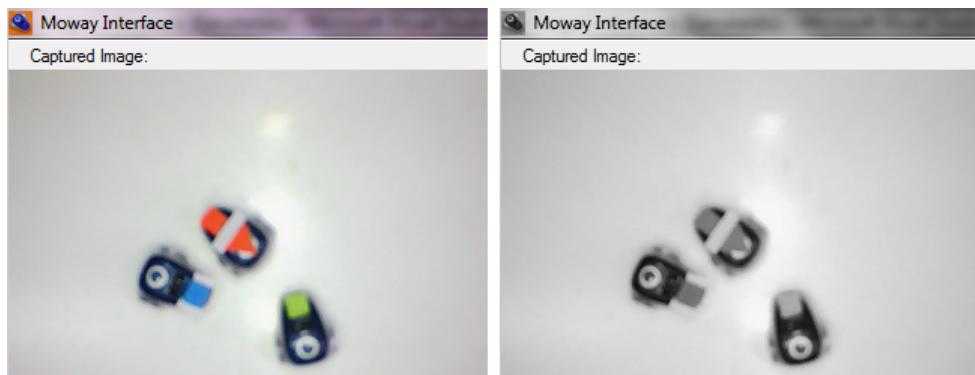


Figura 5.17: Captura del entorno por la interfaz.

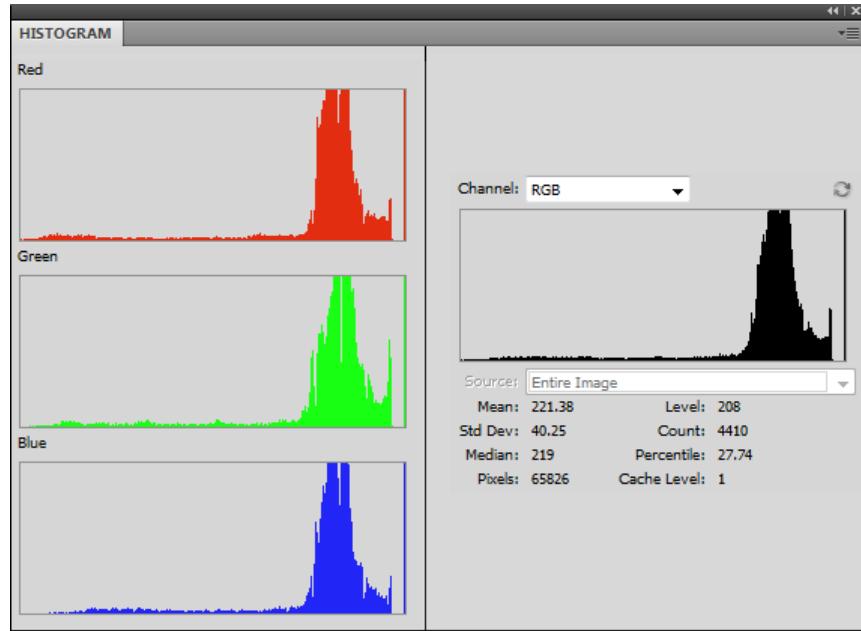


Figura 5.18: Histograma de la imagen capturada por la cámara

El procedimiento de este filtrado se describe como:

- Capturar la data de la cámara
- Convertir la imagen del Sistema RGB al sistema HSV
- Establecer los rangos en los canales H y S.
- Binarizar con los umbrales mencionados el color en interés

Si bien se aplican los filtros definidos por los rangos de los colores en interés del sistema HSV, los resultados presentan ciertas impurezas como podemos apreciar en la Figura 5.19.

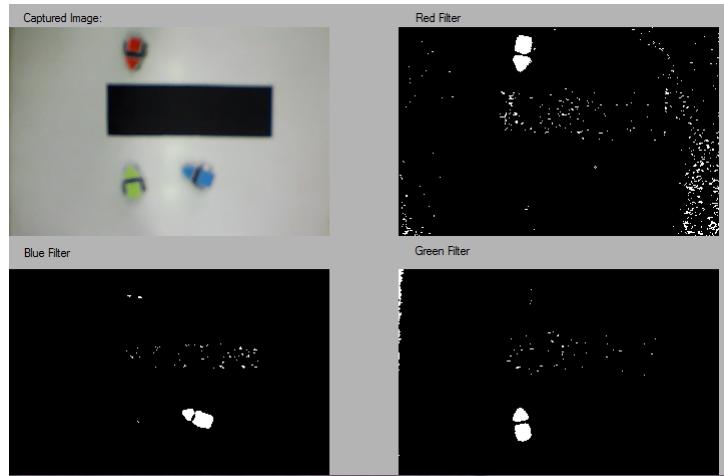


Figura 5.19: Filtrado de robots moway en sistema de color HSV, sin aplicar operaciones morfológicas.

Por ello se hace necesario utilizar algún otro criterio que en el presente caso se da mediante el uso de las operaciones morfológicas (ver apéndice C) sobre las imágenes binarias que resultan de aplicar el filtro de colores. Los resultados tras aplicar las operaciones morfológicas sobre las imágenes se pueden apreciar en las Figuras 5.20 y 5.21.



Figura 5.20: Filtrado de robot Moway rojo en sistema HSV, aplicando ciertas operaciones morfológicas.

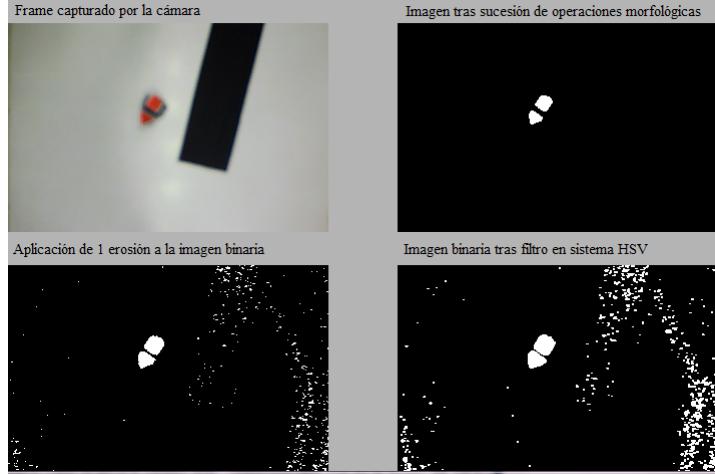


Figura 5.21: Uso de operaciones morfológicas en el procesamiento de imágenes para la obtención de posición y orientación del robot Moway.

Con las imágenes binarias sin impurezas se procede a calcular los estados (posición y orientación) de los moway, para ello el procedimiento es el siguiente:

- Capturar los contornos, si el número de estos es 2 se procede a calcular el número de triángulos y cuadrados presentes en la imagen binaria, para ello se considera que dichos objetos deben presentar un área mayor a 250 píxeles.
- Si el número de triángulos y cuadrados es 1 se procede a calcular los centroides de dichos objetos y posteriormente la posición y orientación del robot móvil.

A. Entorno del Robot El área de trabajo considerada para esta aplicación es de 1m de largo por 0.75m de ancho(ver Figura 5.22), siendo la superficie de color blanca. Los obstáculos a considerar son de color negro, por ende es necesario filtrar ese color.

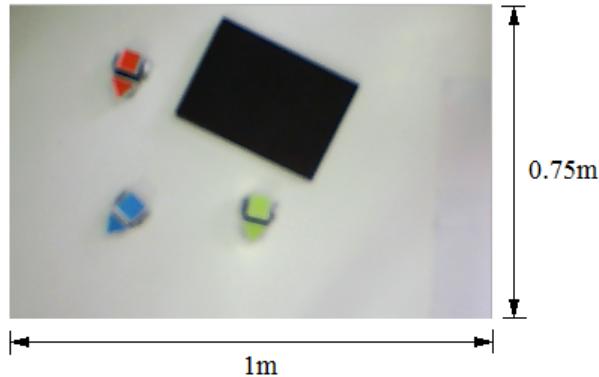


Figura 5.22: Dimensiones del área de trabajo.

B. Posición y Orientación Moway Ambos estados se calculan en base a los centros del triángulo y cuadrado denotados por C1 y C2 respectivamente (ver Figura 5.23). La orientación del robot móvil se halla en base a un vector definido por C1 y C2; en cuanto a la posición se halla con la semisuma de C1 y C2. Ello se ejecuta en la subrutina **estados** (ver Figura 5.15) perteneciente a la función ProcessFrame (ver Figura 5.14)

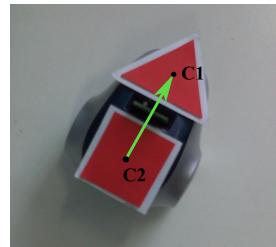


Figura 5.23: Orientación del Robot Moway.

Para el cálculo de los centroides del triángulo y cuadrado se tomará a partir de la imagen binaria obtenida tras el filtro de color en el sistema HSV y tras la aplicación de las operaciones morfológicas, luego se procede a calcular los estados mediante la función **Calcular_estados** (ver Figura 5.14 y 5.15) cuyo argumento es una **imagen binaria**. Esta función captura los contornos presentes en la imagen binaria. Si el número de dichos contornos es mayor a 2, los estados se actualizarían con valores erróneos; caso contrario, cuando se detectan 2 contornos

se procede a la actualización de los estados. La actualización de los estados se calcula en base a los 2 contornos detectados, realizándose una detección del número de triángulos y cuadrados. Si el número de ambos es 1 se procede al cálculo de los centroides, siendo los centroides los que permiten el cálculo de la posición y orientación del robot moway (ver Figura 5.15).

CAPÍTULO 6

Aportes para Aplicaciones Académicas

En este capítulo se presentan los principales aportes de esta tesis para el desarrollo de aplicaciones relacionadas al tema de la investigación. El sistema dado su concepción, podrá ser reutilizado en proyectos futuros que demanden soluciones planificación de trayectorias.

6.1. Plataforma de pruebas

El tener una plataforma con una interfaz de pruebas permite centrarse en el desarrollo de algoritmos. La plataforma con la que se han hecho las pruebas físicas queda para los alumnos de la facultad, los cuales ya no tendrán que enfocarse tanto en la parte de hardware sino concentrarse netamente en el desarrollo de algoritmos.

A su vez la interfaz funciona como sensor ya que calcula la posiciones y orientaciones de los robots moway, es decir, nos permite tener un sistema a lazo cerrado, lo cual es importante para poder también probar algoritmos no solo de generación sino también de control de trayectorias.

Esta plataforma puede interrelacionarse con otros programas, es decir los algoritmos planificadores de trayectoria pueden codificarse en otros lenguajes ya que mediante flujo de data se puede cargar la ruta a la interfaz. En el caso de algoritmos de control de trayectoria estos deben codificarse en C# en la misma

interfaz para optimizar los tiempos de ejecución.

6.2. Ambiente desarrollado en C#

CSharp o C# es un lenguaje orientado a objetos, como el C/C++ con una sintaxis muy similar a éstos, lo cual le hace un lenguaje muy versátil. A su vez muchas aplicaciones se desarrollan en este lenguaje que es joven y popular y sobretodo anda en constante evolución.

6.3. Simulación 3D

La aplicación de la interfaz fue mostrada a la empresa MiniRobots, cuya acogida permitió obtener los archivos CAD del robot moway para realizar las simulaciones en 3D. Para ello se ha firmado un acuerdo de confidencialidad con la compañía de MiniRobots, lo que permite manipular dichos archivos bajo fines académicos beneficiando así a los estudiantes de la UNI. Esto permite simular de manera más real al robot moway en un entorno, apreciándose así los movimientos del robot y su comportamiento.

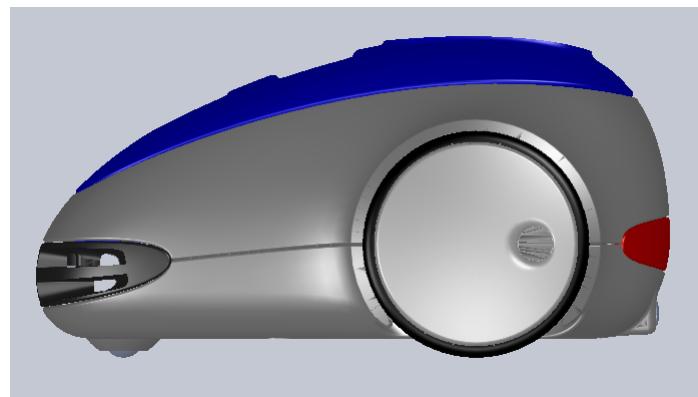


Figura 6.1: Vista de perfil

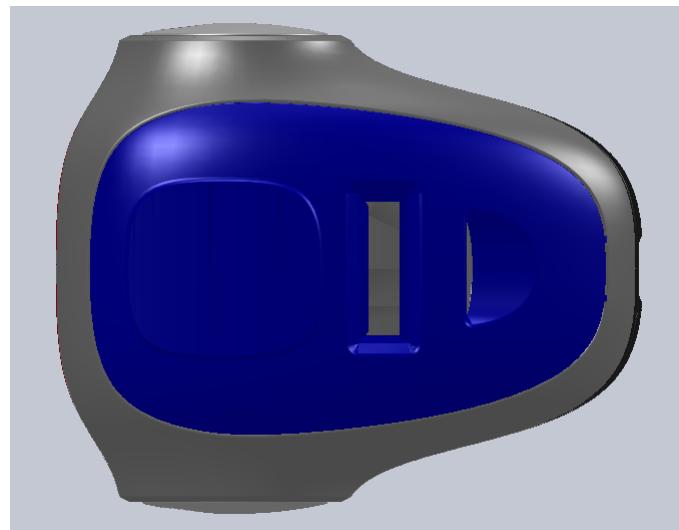


Figura 6.2: Vista superior

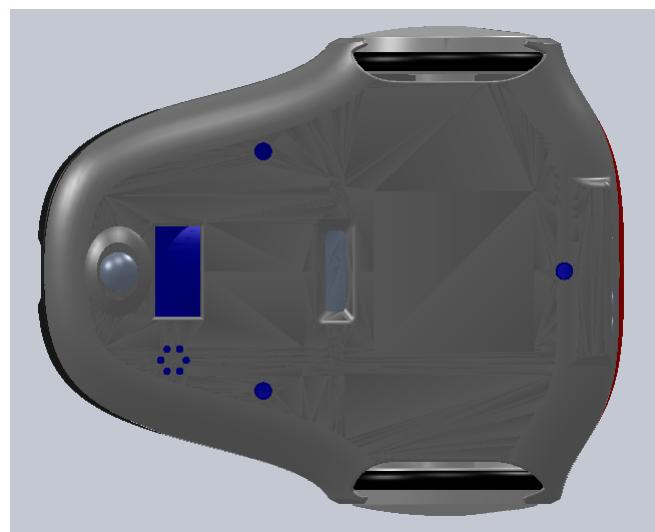


Figura 6.3: Vista inferior



Figura 6.4: Vista isométrica

CONCLUSIONES

- El desarrollo de algoritmos planeadores de trayectoria permite tener una visión mayor en cuanto a desarrollar herramientas para encontrar secuencia de movimientos que permitan realizar las tareas en los robots.
- El tener una plataforma de pruebas permitirá probar y desarrollar una gran variedad de algoritmos, ya que la implementación ha sido desarrollada.
- En las Figuras 4.13, 4.14 y 4.15 se pueden apreciar que la trayectoria generada para el sistema son curvas suaves con respecto a las mostradas anteriormente, en las cuales no se consideraba el modelamiento del sistema. El algoritmo RRT tiene como ventaja el poder extenderse en sistemas del tipo no holónomo, que permite resolver cuestiones propias que caracterizan al sistema, es decir, en el planeamiento de la trayectoria se considera el modelamiento del sistema que permite obtener caminos o rutas que el sistema puede realizar (ver capítulo 4).
- El algoritmo RRT por su costo computacional se ha corroborado que es para desarrollo offline, que puede servir para generar trayectorias de referencia.
- La optimización computacional de los algoritmos es un factor importante ya que permite acelerar el procesamiento de un algoritmo, esto queda demostrado por ejemplo en lo que respecta al uso del algoritmo de KNN. En

la simulación respectiva (ver capítulo 5) se puede corroborar que utilizando este algoritmo la eficiencia de un algoritmo básico mejora notablemente.

BIBLIOGRAFÍA

- [1] Mark W. Greenia. *History of Computing: An Encyclopedia of the People and Machines that Made Computer History*. Lexikon Services Publishing, March 2001.
- [2] Seth Hutchinson George Kantor Wolfram Burgard Lydia Kavraki Sebastian Thrun Howie Choset, Kevin Lynch. *Principles of Robot Motion Theory, Algorithms, and Implementation*. The MIT Press, 2005.
- [3] José Mireles Jr. Kinematics of mobile robots. Technical report, University of Texas Arlington, 2004.
- [4] Moway. *MOWAY USER MANUAL*, June 2010.
- [5] Gary Bradski and Adrian Kaehler Beijing. *Learning OpenCV*. O'Reilly Media, first edition, September 2008.
- [6] Toby Breckon Chris Solomon. *Fundamentals of Digital Image Processing*. Wiley-Blackwell, 2011.
- [7] BIZINTEK INNOVA S.L. *Manual Módulo BZI-RF2GH4*, Febrero 2007.
- [8] Jean-Claude Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. In *International Journal of Robotics Research*, volume 18, pages 1119–1128, Stanford, CA 94305, USA, July 1999. Stanford University.

- [9] Steven M. LaValle. *PLANNING ALGORITHMS*. Cambridge University Press, 2006.
- [10] David Lammers. The era of error-tolerant computing, November 2010.
- [11] Sean Quinlan. *Real Time Modification of Collision-Free Paths*. PhD thesis, Stanford University, December 1994.
- [12] Jean-Claude Latombe Lydia E. Kavraki, Petr Svestka and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE Transactions on Robotics and Automation*, pages 566–580. Robotics Laboratory, Department of Computer Science, Standford University and Department of Computer Science, Utrecht University., 1996.
- [13] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Department of Computer Science; Iowa State University, 1998.
- [14] Canming Huang. www.emgu.com.
- [15] R. Silva Ortigoza; J. R. García Sánchez; V. R. Barrientos Sotelo; M. A. Molina Vilchis; V. M. Hernández Guzmán y G. Silva Ortigoza. Una panorámica de los robots móviles. In *TELEMATIQUE*, volume 6. Revista Electrónica de Estudios Telemáticas, 2007.
- [16] Moway. *Manual librería Base*, Abril 2008.

APÉNDICE A

Motion Planning

A.1. Reseña histórica de robots móviles

Una breve reseña extraída de [15] nos puede dar un panorama de la evolución de la robótica móvil desde los años 70:

“A principios de la década del setenta, el robot Newt [Hollis, 1977] fue desarrollado por Hollis. El robot Hilare [Giralt, 1979] desarrollado en el LAAS en Francia. En el Jet Propulsion Laboratory (JPL) se desarrolló el Lunar rover [Thompson, 1977], diseñado particularmente para la exploración planetaria. A finales de esa década, Moravec desarrolló el robot Stanford cart [Movarec, 1979], capaz de seguir una trayectoria delimitada por una línea establecida en una superficie, en el SAIL. En 1983, el robot Raibert [Raibert, 1986], fue desarrollado en el MIT, un robot de una sola pata diseñado para estudiar la estabilidad de éstos sistemas. A principios de la década del noventa, Vos et al. desarrollaron un robot “uniciclo” [Vos et al., 1990] (una sola rueda, similar a la de una bicicleta) en el MIT. Años más tarde, en 1994, el Instituto de robótica CMU desarrolló el robot Dante II [Bares et al., 1999], un sistema de seis patas. En 1996 también en el CMU, se desarrolló el robot Gyrover [Brown et al., 1997], un mecanismo ausente de ruedas y patas basado en el funcionamiento del giroscopio. Ese mismo año se desarrolló en el MIT el Spring Flamingo [Pratt et al., 1998], un robot que emulaba el movimiento de un flamingo. Por su parte, la NASA en 1997 envío

a Marte un robot móvil teleoperado llamado Sojourner rover [Muirhead, 1997], dedicado a enviar fotografías del entorno de dicho planeta. Ese mismo año, la empresa japonesa HONDA, dio a conocer el robot P3 [Tanie, 2003], el primer humanoide capaz de imitar movimientos del cuerpo humano. Al siguiente año, se desarrolla en la universidad Waseda en Japón, el WABIAN R-III [Hashimoto, 1998], un robot humanoide. En 1999 en el CMU, Zeglin propuso un nuevo diseño de robot con una pata llamado Bow Leg Hopper [Zeglin, 1999], un diseño que permite almacenar la energía potencial de la pata. En 2006, Hollis et al. desarrollaron el robot Ballbot [Lauwers, 2006], un sistema holónomo cuyo movimiento es proporcionado por una sola esfera ubicada en la parte inferior de la estructura. Sin embargo, el estudio de este tipo de robots con una esfera, fue iniciado por Koshiyama y Yamafuji [Koshiyama et al., 1991] en 1991. Actualmente los robots teleoperados Spirit rover y Opportunity rover, [Aronson et al., 2001], se encuentran explorando la superficie del planeta Marte en busca de mantos acuíferos. Los robots aquí mencionados, son únicamente una porción de los tantos que se han diseñado, sin embargo, es posible notar que las aplicaciones de estos son vastas y que las mismas son ilimitadas debido al desarrollo cada vez más vertiginoso de la tecnología.”

APÉNDICE B

Robot Moway

El robot Moway (ver Figura B.1) es un robot programable diseñado principalmente para desarrollar aplicaciones prácticas de robótica móvil. Está equipado con una serie de sensores que le permiten moverse en un entorno real, así como también puede comunicarse inalámbricamente (mediante tecnología de radiofrecuencia) con la PC. Sus dimensiones son aproximadamente de 6cm x 4cm y con 3cm de alto, posee 2 ruedas motrices y una esfera metálica que se utiliza como rueda loca para el robot móvil.



Figura B.1: Robots Moway. Tomado de [4]

B.1. Arquitectura

El robot Moway esta conformado por los siguientes elementos (ver Figura B.2):

- Procesador
- Sistema de Navegación
- Sensores en indicadores de grupos
- Sistema de Potencia
- Conector de Expansión

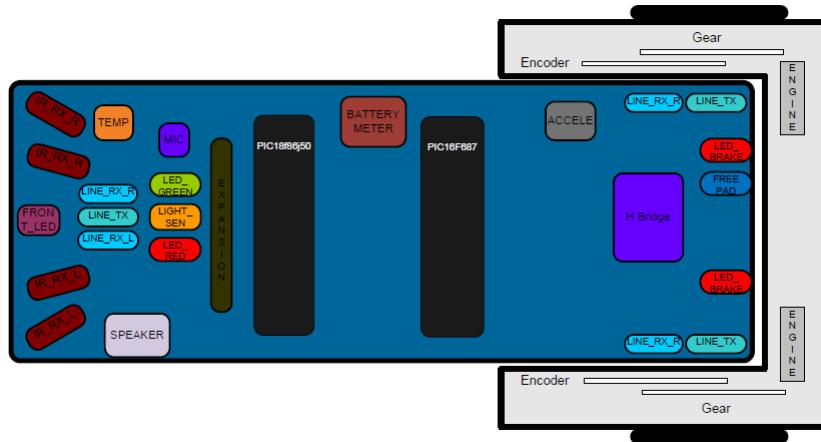


Figura B.2: Partes del robot Moway. Tomado de [4]

B.1.1. Procesadores

Los Moways son gobernados por el microcontrolador PIC18F87J50 de 4MHz, este procesador se encarga tanto de las comunicaciones, de procesar la data recibida por los sensores y el que gobierna el sistema de navegación del móvil.

B.1.2. Sistema de Navegación

Está dado por el PIC16F887 que mediante un puente H varia la velocidad de las ruedas a través de los motores DC que posee el robot Moway. Este microcontrolador es el esclavo del microcontrolador principal PIC18F87J50.

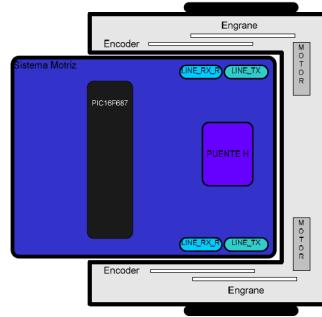


Figura B.3: Drive System: electrónica y mecánica. Tomado de [4]

B.1.3. Sensores e Indicadores

Permiten al Moway relacionarse con su entorno y dicha información se puede enviar inalámbricamente a la PC.

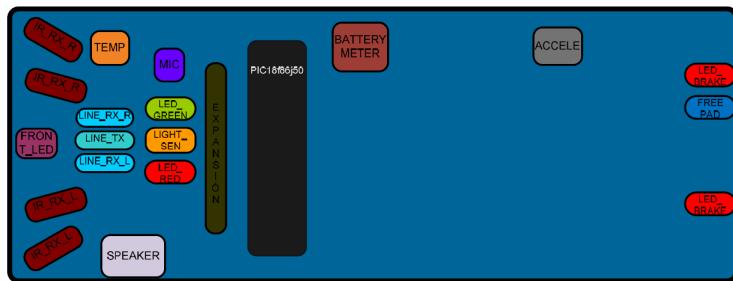


Figura B.4: Sensores y grupo de indicadores. Tomado de [4]

B.1.4. Sistema de Potencia

El robot tiene una batería LiPo recargable a través del puerto USB del Moway desde la computadora.



Figura B.5: Fuente de sistema de alimentación. Tomado de [4]

B.1.5. Conector de Expansión

Es un módulo RF(ver Figura B.6) que permite interacción entre el Moway con la PC a través de un RF-USB(ver Figura B.7), la cual es en sí un modulo de RF con una extensión de RS232 a USB.



Figura B.6: Módulo RF. Tomado de [4]



Figura B.7: RF-USB. Tomado de [4]

B.2. Moway GUI

Es una interfaz (ver Figura B.8) que permite interactuar con el robot, pudiendo programarlo en assembler, C18 o mediante bloques en un diagrama de flujo. En los casos mencionados se debe referenciar las librerías del robot moway para accesar tanto a la data como periféricos de robot.

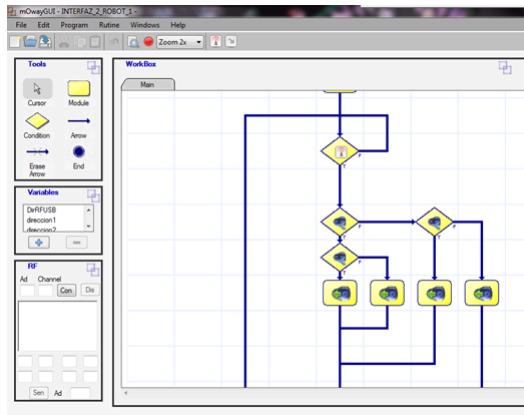


Figura B.8: Interfaz gráfica del robot Moway.

Moway GUI tiene una sección para realizar pruebas con la radiofrecuencia BZI-RF2GH4[7], la cual permite la comunicación inalámbrica entre el robot Moway y la PC.

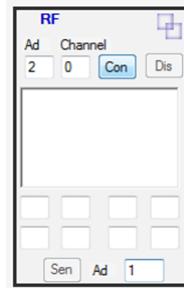


Figura B.9: Sección de Control de RF en la Interfaz gráfica Moway GUI.

Pero esta sección (ver Figura B.9) permite mandar data en un paquete conformado por 8 bytes pero que tienen que ser seteados manualmente en la interfaz de Moway GUI, por esta razón solo se utilizará la aplicación Moway GUI para

programar a los Moway y realizar pequeñas pruebas de comunicación pero no será la interfaz del robot en la presente tesis.

B.3. Interfaz C#

Es un proyecto ejemplo de Moway escrito en C#(ver Figura B.10), el cual posee las funcionalidades de Moway GUI debido a que incluye las 3 dlls que permiten controlan al robot:

- lib_nrf24lu1_RF, desarrolla el protocolo de comunicación inalámbrico que utilizan los robots Moway, es decir define por ejemplo el tiempo de confirmación del envío de data así como el empaquetamiento de la información y cabeceras de la data a enviar.
- lib_prog_mow2, permite grabar sobre el robot moway.
- LibUsbDotNet, es una librería USB escrita en .NET C# para drivers WinUsb, libusb-win32 y Linux libusb v1.x. Todas las funcionalidades básicas del dispositivo USB se pueden realizar a través de clases comunes de dispositivo, lo que le permite interactuar con el sistema operativo.

Esta interfaz es la base de la que se ha desarrollado en la presente tesis, la cual agrega visión computacional y control de trayectoria del robot Moway. Para entender las funciones de la interfaz del fabricante, se recomienda revisar [16].

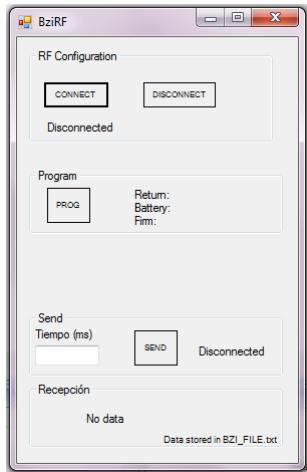


Figura B.10: Interface Moway en C#

APÉNDICE C

Visión Artificial

Visión Artificial es la transformación de data desde una cámara fotográfica o de video en algún otro tipo de representación para el logro de algún objetivo en particular. El ser humano realiza dicha transformación dividiendo la señal de visión en varios canales que proveen diferentes tipos de información al cerebro. El cerebro tiene un sistema de atención que identifica, en una tarea de modo dependiente, partes importantes de una imagen a examinar mientras sorpresivamente examina otras áreas. Hay una masiva retroalimentación en el flujo visual que es aún no entendido por completo, así como también hay una amplia asociación de entradas de varios sentidos que permiten al cerebro bosquejar situaciones aprendidas por la experiencia. En los sistemas de visión artificial, sin embargo, una computadora recibe una matriz de números (por ejemplo lo mostrado en la Figura C.1) desde la cámara donde por defecto no hay un reconocimiento de patrones, no hay un control automático de focus y apertura, no hay asociaciones cruzadas con los años de experiencia. Es por tal razón que a partir de la información se aplican diversos algoritmos para lograr el objetivo en particular. En este caso el objetivo será encontrar la posición y orientación del robot Moway en su entorno.

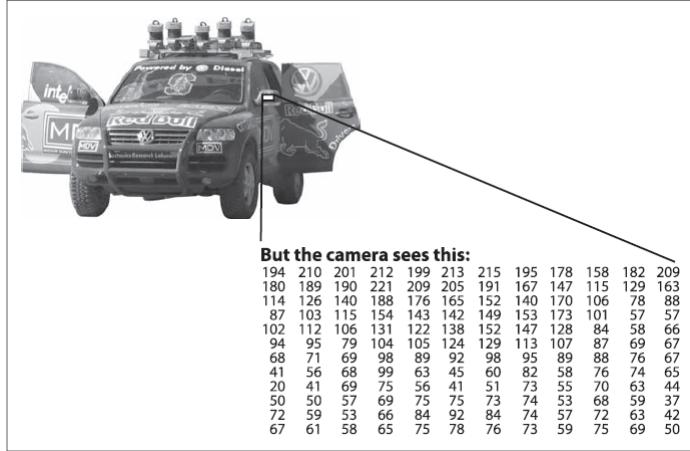


Figura C.1: Para la computadora, lo que capta la cámara es una matriz de números. Tomado de [5].

C.1. Data de la Imagen

Lo captado por la cámara de video es una secuencia de frames (fotogramas). Cada frame es una matriz, o un array bidimensional de números siendo cada celda un píxel¹. Un píxel contiene la información de la combinación de cada uno de los canales del sistema de colores que se utiliza, por ejemplo en el sistema RGB cada canal toma valores entre el rango de 0 a 255.

C.1.1. Representación de Imágenes Digitales

1. Imagen Binaria, 1 píxel es representado por 1 bit: 1 ó 0.
2. Imagen en escala de grises, 1 píxel es representado por 1 byte: 0-255.
3. Imagen en color, por ejemplo en el sistema RGB cada píxel consta de 3 valores: Rojo, Verde, Azul. Con un byte para cada canal se puede obtener 16,7 millones posibles colores.

Los 3 representaciones mencionadas se aprecian en la Figura C.2.

¹La palabra píxel es una abreviación de ‘picture element’, es la unidad básica de información en una imagen.

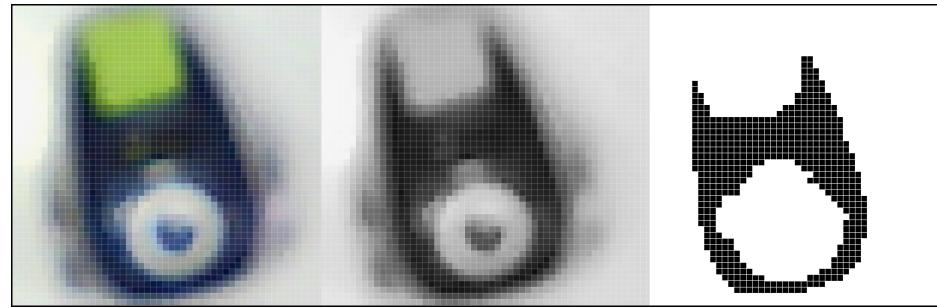


Figura C.2: De izquierda a derecha: a colores en RGB, en escala de grises e imagen binaria (con un threshold de 28).

C.2. Histograma de una imagen

Un histograma es un gráfico que muestra la distribución de los valores de intensidad de cada píxel en una imagen, donde la abcisa horizontal indica la intensidad y la abcisa vertical indica la cantidad de píxeles con dicha intensidad. La intensidad podría expresarse en escala de grises teniendo 256 elementos cuyos valores van desde 0 a 255 ó también podría normalizarse dicha frecuencia, pudiéndose tratar el histograma como una función de densidad de la probabilidad que define la vecindad del valor de un píxel dentro de la imagen. Una inspección visual del histograma permite revelar los contrastes básicos presentes en la imagen. En el ámbito de la fotografía se le utiliza como herramienta para mejorar la exposición de las fotos, corrigiendo los colores con técnicas de ecualización.

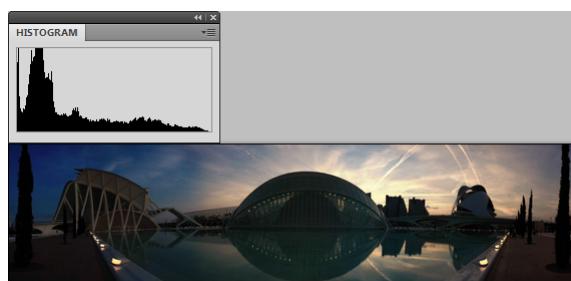


Figura C.3: Imagen sub-expuesta.

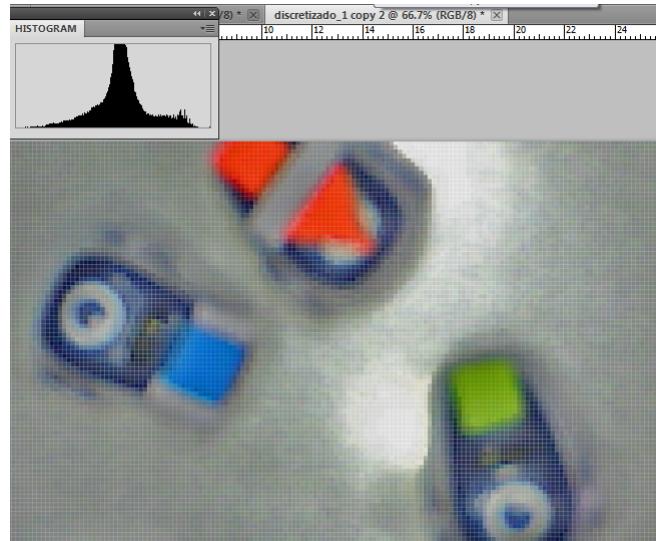


Figura C.4: Imagen propiamente expuesta.

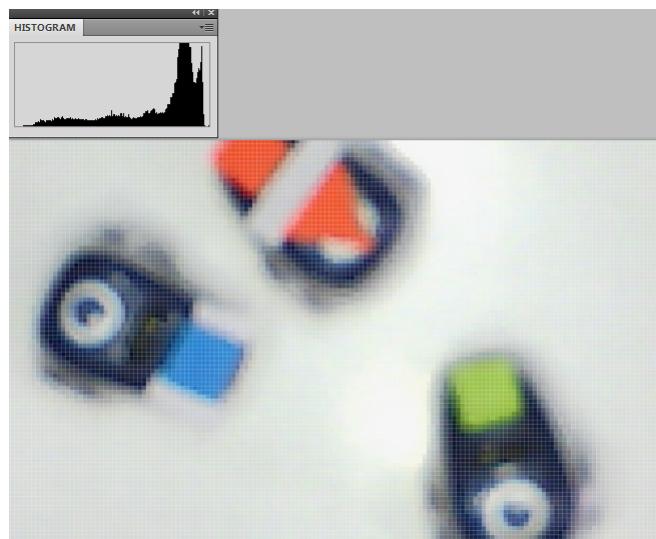


Figura C.5: Imagen sobre-expuesta.

C.3. Sistema HSV

El modelo HSV (conocido también como HSB) fue creado en 1978 por Alvy Ray Smith. Se trata de una transformación no lineal del espacio de color RGB, y se puede usar en progresiones de color. El formato HSV no es muy sensible al cambio de iluminación, es decir nos conviene en ambientes donde la iluminación es muy inestable (ver Figura C.6).

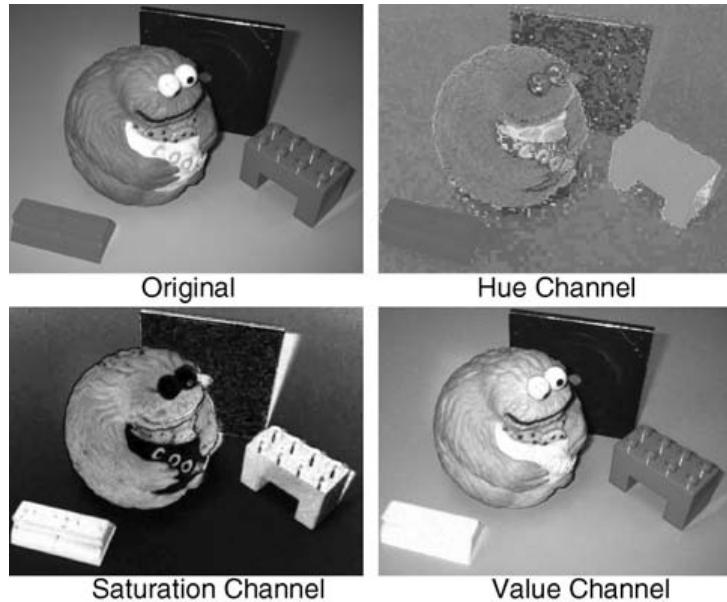


Figura C.6: Imagen transformada y mostrada en el espacio de color HSV, tomada de [6].

Los 3 canales de este formato de color son:

- **H:** Hue (Tonalidad), se representa como un grado de ángulo cuyos valores posibles van de 0 a 360° (aunque para algunas aplicaciones se normalizan del 0 al 100 %). Cada valor corresponde a un color. Ejemplos: 0 es rojo, 60 es amarillo y 120 es verde.
- **S:** Saturation (Saturación), se representa como la distancia al eje de brillo negro-blanco. Los valores posibles van del 0 al 100 %. A este parámetro también se le suele llamar "pureza" por la analogía con la pureza de excitación y la pureza colorimétrica de la colorimetría. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará. Por eso es útil definir la insaturación como la inversa cualitativa de la saturación.
- **V:** Value (Valor), el brillo del color. Representa la altura en el eje blanco-negro. Los valores posibles van del 0 al 100 %. 0 siempre es negro. Depend-

diendo de la saturación, 100 podría ser blanco o un color más o menos saturado.

- **B:** Brightness (Brillo).

Algunas formas gráficas de la representación de este sistema (ver Figura C.7 y C.8):

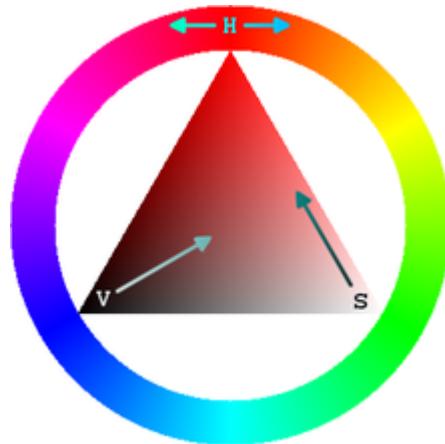


Figura C.7: Espacio de color HSV como una rueda de color.

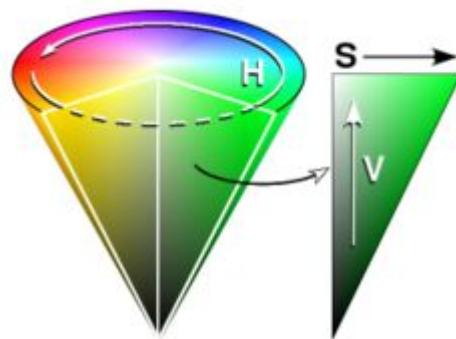


Figura C.8: Cono de colores del espacio HSV.

En EMGU, los rangos de valores de los canales del modelo HSV son:

- H:0-255
- S:0-255
- V:0-255

C.4. Operaciones morfológicas en imágenes

La palabra morfología se refiere al estudio de la forma o estructura de un objeto. En el procesamiento de imágenes se utiliza la morfología matemática para la identificación y extracción de descriptores significativos de las imágenes; tales como contornos, esqueletos, entre otros. Es decir se simplifican las imágenes y se conservan las principales características de forma de los objetos.

La morfología en imágenes abarca un poderoso e importante grupo de métodos que pueden ser tratados matemáticamente dentro del marco de la teoría de conjuntos, la teoría de retículos, la topología y las funciones aleatorias. Conceptos topológicos y geométricos de espacio continuo, tales como tamaño, forma, convexidad, conectividad y distancia geodésica, se pueden caracterizar por la morfología matemática en espacios continuos y discretos.

Las operaciones morfológicas pueden ser aplicadas a imágenes de todo tipo, pero su uso es extendido principalmente en imágenes binarias siendo las operaciones clave la dilatación y erosión ya que muchos procedimientos morfológicos sofisticados pueden ser reducidos a una secuencia de dilataciones y erosiones.

El resultado de una operación morfológica depende de 3 cosas: la imagen de entrada, el tipo de operación y el elemento estructural. El elemento estructural (ver Figura C.9, tomada de [6]) es quien define la forma y el tamaño de la vecindad del píxel que será analizado para posteriormente alterar su valor. Su tamaño es muy inferior al tamaño de la matriz original que define el conjunto a modificar. Esta compuesto de unos y ceros en imágenes binarias de forma y tamaño arbitrario en la cual las posiciones donde está el uno define la vecindad.

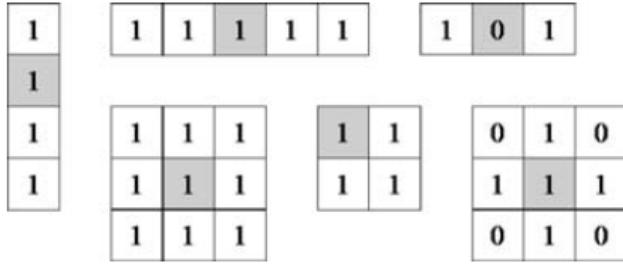


Figura C.9: Algunos ejemplos de elementos morfológicos estructurales. El píxel centro de cada elemento estructural esta sombreado. Figura tomada de [6]

C.4.1. Dilatación

$$\delta_C(A) = A \oplus C = \{x | (\hat{C})_x \subset A \neq 0\}$$

Se obtiene en base a la reflexión de C con respecto a su origen y un desplazamiento x.

La salida de la dilatación es el conjunto de puntos barridos por el centro del elemento estructural mientras algún punto de C coincide con alguno de A. Alternativamente (véase figuras C.10 y C.11), puede interpretarse la dilatación como el resultado de reemplazar cada píxel blanco de la imagen original por una réplica del elemento estructural. La dilatación añade todos los puntos del fondo que tocan el borde de un objeto, es decir, es extensiva.

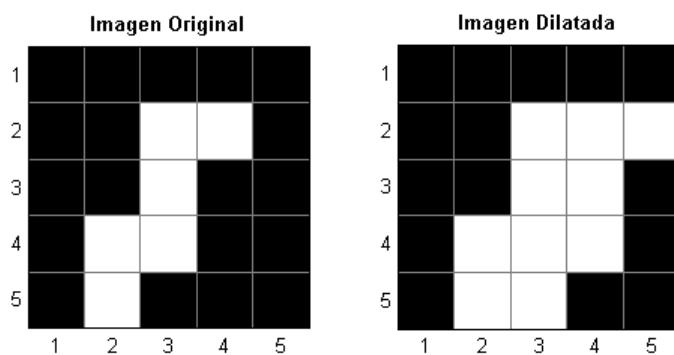


Figura C.10: Aplicación de dilatación.

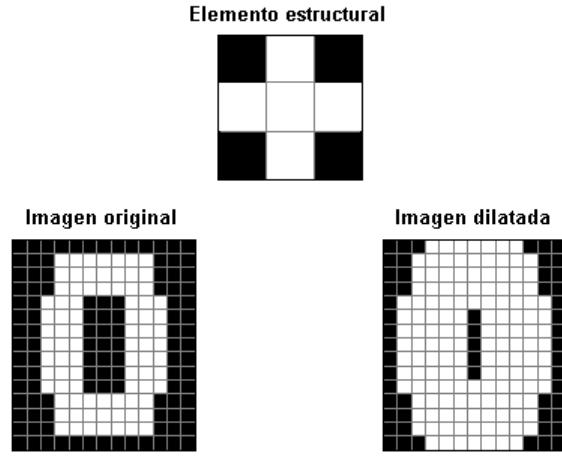


Figura C.11: Aplicación de dilatación.

C.4.2. Erosión

$$\varepsilon_C(A) = AC = \{x | (C_x \subset A)\}$$

La salida de la erosión es el conjunto de puntos barridos por el centro del elemento estructural mientras se cumpla que todos los puntos de C estaban contenidos en A. Elimina grupos de píxeles donde el elemento estructural no está contenido (ver figuras C.12 y C.13). La erosión reduce el tamaño del objeto, es decir, es antiextensiva. La aplicación más común de la erosión es la eliminación de detalles irrelevantes.

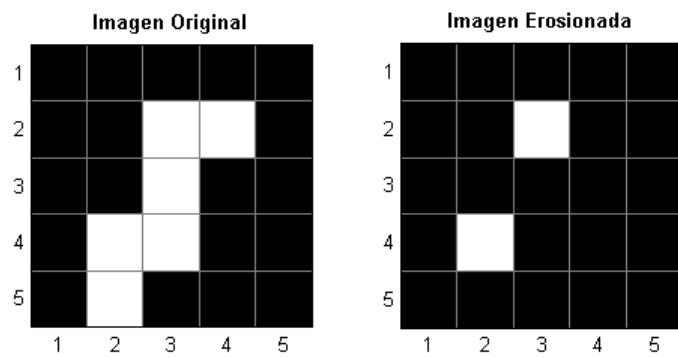


Figura C.12: Aplicación de erosión.

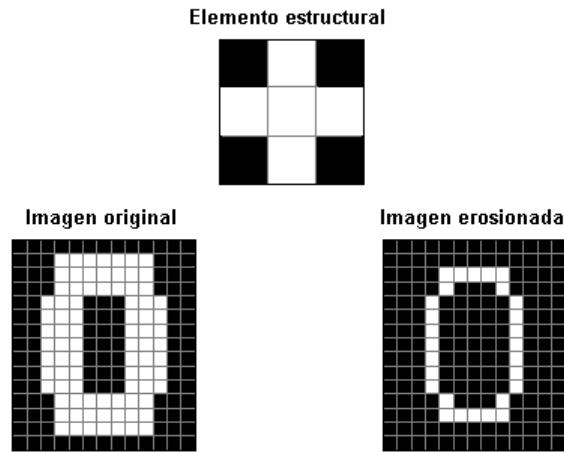


Figura C.13: Aplicación de dilatación.

C.4.3. Apertura

$$\gamma_C(A) = A \circ C = (AC) \oplus C = \delta_C(\varepsilon_C(A))$$

Es la composición de un operador de erosión y otro de dilatación con el mismo elemento estructurante (ver figura C.14).

Se obtiene desplazando el elemento estructural C por el interior del conjunto y eliminando las zonas por las que C no pueda "pasar". Entre sus efectos se tiende a alisar contornos (redondear esquinas que no contengan al elemento estructural) así como también a separar objetos en puntos estrechos.

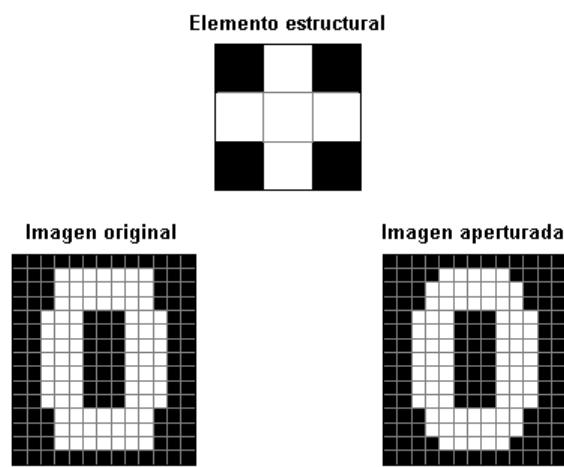


Figura C.14: Aplicación de dilatación.

C.4.4. Cierre

$$\varphi_C(A) = A \bullet C = (A \oplus C)C = \varepsilon_C(\delta_C(A))$$

Es la composición de un operador de dilatación seguido de otro de erosión con el mismo elemento estructural (ver figura C.15). El cierre tiende a alisar porciones del contorno, a fusionar grietas estrechas, llenar vacíos del contorno (agujeros) y conectar objetos vecinos.

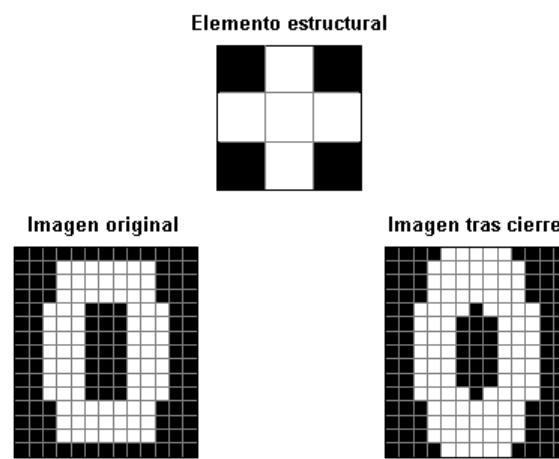


Figura C.15: Aplicación de dilatación.