DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: DESARROLLO DEL SISTEMA DE NAVEGACIÓN PARA ROBOT CUADRÚPEDO DE DIMENSIONES REDUCIDAS EN ENTORNOS NO DEFINIDOS CON PERSPECTIVA CENITAL

AUTORES: CABALLEROS SOLIZ, JESSICA ALEXANDRA
SEGARRA VÁSCONEZ, DAVID EDUARDO

DIRECTOR: DR. AGUILAR CASTILLO, WILBERT GEOVANNY

SANGOLQUÍ

2018

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación **"DESARROLLO DEL SISTEMA DE NAVEGACIÓN PARA ROBOT CUADRÚPEDO DE DIMENSIONES REDUCIDAS EN ENTORNOS NO DEFINIDOS CON PERSPECTIVA CENITAL"** fue realizado por la señorita **Caballeros Solíz, Jessica Alexandra** y el señor **Segarra Vásconez, David Eduardo** el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolqui, 20 de Agosto del 2018

Dr. Wilbert G. Aguilar
CI: 0703844696

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Nosotros, **Caballeros Solíz, Jessica Alexandra** y **Segarra Vásconez, David Eduardo,** declaramos que el contenido, ideas y criterios del trabajo de titulación: **"DESARROLLO DEL SISTEMA DE NAVEGACIÓN PARA ROBOT CUADRÚPEDO DE DIMENSIONES REDUCIDAS EN ENTORNOS NO DEFINIDOS CON PERSPECTIVA CENITAL"** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas utilizadas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolqui, 20 de Agosto del 2018

Caballeros Solíz, Jessica Alexandra
CI: 1720841137

Segarra Vásconez, David Eduardo
CI: 1722291463

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Nosotros, **Caballeros Solíz, Jessica Alexandra** y **Segarra Vásconez, David Eduardo,** autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"DESARROLLO DEL SISTEMA DE NAVEGACIÓN PARA ROBOT CUADRÚPEDO DE DIMENSIONES REDUCIDAS EN ENTORNOS NO DEFINIDOS CON PERSPECTIVA CENITAL"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolqui, 20 de Agosto del 2018

Caballeros Solíz, Jessica Alexandra
CI: 1720841137

Segarra Vásconez, David Eduardo
CI: 1722291463

DEDICATORIA

Dedico este proyecto a mi familia:

A mis padres, Humberto y Miriam, por el amor y paciencia que me brindan, mostrándome el camino hacia la superación, ustedes son el pilar fundamental en mi formación.

A mis hermanos, Cristian y Shosue, por apoyarme en cada paso que doy.

A mi querida amiga, Pauly, porque siempre tuviste las palabras perfectas, que calmen mi mente y alivien mi corazón, jamás encontraré un mejor lugar que estar entre tus brazos. Agradezco a Dios, por el tiempo que recorrimos juntas, a pesar de que no fue suficiente.

Esto es posible gracias a ustedes.

Jessica A. Caballeros S.

DEDICATORIA

A Dios, forjador de mi camino, quien siempre me acompaña.

A mi madre, Marjorie, quien, sin su sacrificio personal, paciencia, guía, apoyo y amor incondicional desde el momento en que nací, nunca habría llegado a ser la persona que soy ahora.

A mis queridos abuelitos, Aidita y Alejandro, padres para mi, grandes fuentes de sabiduría, afecto y cariño durante toda mi vida.

A mi tía Narci y prima Gaby, que han estado presentes y han hecho posibles momentos importantes en mi vida.

A mi familia, pilar fundamental de valores y enseñanzas.

A mis amigos cercanos, personas amables que he tenido el privilegio de conocer y se han convertido prácticamente en hermanos y compañeros de aventuras.

Este logro es también de ustedes.

David E. Segarra V.

AGRADECIMIENTOS

Agradezco a Dios, por darme la valentía y fuerza necesaria para afrontar cualquier adversidad.

A mis queridos padres, por darme tanto de todo y por darme todo de ustedes, gracias por creer y confiar en mí, porque ustedes me permiten soñar y volar lejos, y tener la confianza de poder volver a un lugar llamado hogar. A mi padre, por los consejos y cada palabra de sabiduría en todo el trascurso de mi vida. Gracias a mi madre, porque con su amor y comprensión, hoy culmino mi carrera profesional. A mi hermano pequeño, por todos los abrazos, peleas, cosquillas, pero sobre todo gracias por brindarme todo tu cariño y amor sincero, eres la persona más importante en mi vida. A mi hermano mayor, por ser mi ejemplo a seguir, solo deseo que te sientas orgulloso de este logro, gracias por el gran amor incondicional que me ofreces.

A una persona muy especial en mi vida, Cristian, ha sido parte de mi fortaleza e inspiración.

Expreso mi infinita gratitud a mi tutor de tesis, Wilbert, por el conocimiento compartido durante estos últimos años, por cada detalle y momento dedicado para la culminación de mi paso por la universidad.

Gracias a mi universidad, por haberme permitido fórmame en ella, y a todas las personas que fueron participes de este proceso, gracias a todos mis amigos, fueron ustedes los responsables de realizar su pequeño aporte, para cumplir uno de mis sueños.

A mi amigo David, por todo el tiempo de amistad, ha sido un privilegio poder compartir las aulas contigo, porque eres una persona extraordinaria, gracias por las palabras de aliento y me alegro mucho de poder lograr este triunfo a tu lado. Amigo, hoy, todo valió la pena. ¡Lo hicimos!

Jessica A. Caballeros S.

AGRADECIMIENTOS

Primero quiero agradecer a Dios sin quien nada es posible, por darme la fuerza necesaria y permitir encontrarme continuamente con salud y una buena vida para completar este capítulo.

Ha habido muchas personas que han caminado a mi lado a lo largo de mi Carrera Universitaria, me han guiado, apoyado, dado oportunidades, compartido conocimientos, y me han mostrado puertas que podrían ser útiles abrir. Me gustaría dar las gracias a cada uno de ellos.

A mi familia, debido a su gran cariño, virtudes y valores que me han brindado, en especial a mi madre por su gran apoyo, cuidado continuo, abundante paciencia y comprensión, por enseñarme a superar todo lo que me proponga. Sin ellos este proyecto y título no hubieran sido posibles.

A Jessi, persona con quien compartí el honor de trabajar en este proyecto y quien, en los últimos años, ha sido una gran compañera de estudios, responsable, dedicada, pero en especial una amiga con la que puedo confiar y contar en cualquier cosa y ella también conmigo.

A mis profesores los cuales han sido una fuente constante de conocimiento e inspiración, en especial a Wilbert por transmitir un espíritu de aventura en lo que respecta a la investigación, por su ánimo y ayuda a lo largo de este proyecto.

Deseo también agradecer a mis amigos quienes han estado en momentos de alegría y dificultad en los cuales me han animado y mostrado su afecto.

Por último, si bien no menos importante, gracias a la Universidad por acogerme y darme la formación, conocimientos necesarios, y el apoyo en experiencias académicas internacionales.

David E. Segarra V.

INDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En varios campos de la ciencia es necesario el uso de robots capaces de moverse de un punto a otro en zonas irregulares para el logro de sus objetivos, zonas en donde los robots articulados toman gran importancia, ya que estos presentan una ventaja considerable en su movilidad. Por esta razón el presente proyecto de investigación va dirigido al desarrollo del sistema de navegación mediante planificación de trayectoria para un robot cuadrúpedo en entornos no definidos con perspectiva cenital, esto se realiza a través del análisis del modelo matemático del diseño físico del robot cuadrúpedo nombrado SpiderBot, empleando cinemática directa con la ayuda del método de Denavit-Hartenberg, destacando sus propias características de movimiento en tres dimensiones para trasladarse desde una posición a otra , con el propósito de programar su algoritmo de desplazamiento, también se implementaron algoritmos de detección y evasión de objetos, usando particularidades de morfología y color, con el fin de poder determinar la orientación del robot y evitar colisiones entre SpiderBot y un obstáculo estático, por ultimo para la planificación de trayectoria se usa técnicas probabilísticas e información parcial sobre el medio espacio de configuración, haciendo uso del algoritmo RRT (*Rapidly Exploring Random Trees*). El prototipo implementado actualmente está totalmente terminado y funcional. Una vez finiquitada la etapa de experimentación se logró optimizar el algoritmo de planificación, manteniendo un tiempo relativamente bajo para concluir con la estimación de la ruta.

PALABRAS CLAVE:

- **ROBOT CUADRÚPEDO**
- **PLANIFICADOR DE RUTA**
- **EVASIÓN DE OBSTÁCULOS**

ABSTRACT

In several fields of science, it's necessary to use robots capable of moving from one point to another in irregular surfaces to achieve their objectives, surfaces where articulated robots take great importance, since they present a considerable advantage in mobility. For this reason the present research project is aimed to develop a navigation system through path planning for a quadruped robot in undefined environments with overhead perspective, this is done by the analysis of the mathematical model of the physical design of the robot named SpiderBot , using kinematics with the help of the Denavit-Hartenberg method, highlighting movement characteristics in three dimensions to travel from one position to another, with the purpose of programming its displacement algorithm, object detection and evasion algorithms were also implemented, using morphology and color characteristics, in order to determine the orientation of the robot and avoid collisions between SpiderBot and a static obstacle, finally, to do the path planning, probabilistic techniques and partial information about the configuration half space are used, using the RRT algorithm (Rapidly Exploring Random Trees). The prototype is currently implemented, entirely finished and functional. Once the experimentation stage was concluded, the planning algorithm was optimized, maintaining a relatively low time to complete the estimation of the route.

KEYWORDS:

- **QUADRUPED ROBOT**
- **PATH PLANING**
- **OBSTACLE AVOIDANCE**

CAPITULO I

INTRODUCCIÓN

1.1 Antecedentes

En la actualidad uno de los principales objetivos de la robótica es desarrollar métodos robustos de planificación autónoma de trayectorias (Menéndez, 2012). Muchas aplicaciones necesitan el uso de robots capaces de moverse para el logro de sus objetivos, siendo aquí donde los robots articulados toman importancia. Es complicado el uso de robots con ruedas cuando se trata de moverse en zonas irregulares y desarrollarse con autonomía. (Fernández & Barrientos, 2016).

Evitar obstáculos en entornos no definidos es uno de los problemas más importantes y de mayor interés en el campo de la robótica móvil. (Bertozzi, Broggi, & Fascioli, 1996). Desde que se crearon los robots industriales, quedó claro lo fácil que estos podrían colisionar con su entorno, o entre ellos. Desde entonces, la búsqueda de métodos anticolisión ha exigido la atención de los investigadores. En el primer caso se encuentran los métodos de planificación de rutas y, en el segundo caso, los métodos de evasión de obstáculos. Los métodos de planificación de rutas se encuentran en las rutas automáticas sin colisiones, mientras que los métodos de evasión de obstáculos consisten en eludir los obstáculos que surgen en el camino del robot, al mismo tiempo que logra alcanzar el objetivo, en el cual la toma de decisiones se da de acuerdo con la información capturada desde el entorno en el que el robot está (Alvarez, Antonio, Estrada, Fernández, & Torres, 2010).

El problema general de planificación de movimiento es tratar de encontrar una trayectoria libre de colisiones desde algún estado de partida a alguna región meta siendo ampliamente estudiado. En particular, las técnicas basadas en el muestreo han recibido mucha atención en los últimos 15 años. El RRT (Rapidly-exploring Random Tree) opera creando un árbol en el espacio, muestreando

iterativamente nuevos estados y luego dirigiendo el nodo existente que esta más cerca de cada nueva muestra hacia una nueva muestra formando un árbol con ramificaciones. El RRT tiene muchas propiedades útiles, incluyendo la integridad probabilística y decadencia exponencial de la probabilidad de fracaso con el número de muestras (Karaman & Frazzoli, 2010).

A medida que el uso de robots móviles se hace más común en la vida cotidiana, la necesidad de una estrategia de navegación precisa en un entorno interior / exterior aumenta; un método para la planificación de movimiento en tiempo real en un terreno nuevo desarrollado en la Universidad Tecnológica de Poznan permite al robot encontrar un camino a la posición deseada y discriminar entre áreas atravesables y no recorridas. El concepto de árboles al azar de exploración rápida se utiliza como columna vertebral de la solución propuesta. El método propuesto emplea varios módulos para planificar la trayectoria del robot, la prevención de colisiones y el análisis de estabilidad. (Borenstein & Koren, 1989).

Todos los robots móviles cuentan con algún tipo de prevención de colisiones, que van desde algoritmos primitivos que detectan un obstáculo y detienen al robot para evitar una colisión, hasta sofisticados algoritmos, que permiten al robot desviar obstáculos. Estos últimos algoritmos son mucho más complejos, ya que implican no sólo la detección de un obstáculo, sino también algún tipo de medidas cuantitativas sobre las dimensiones del obstáculo. Una vez que se han determinado, el algoritmo de evitación de obstáculos necesita dirigir al robot alrededor del obstáculo y reanudar el movimiento hacia el objetivo original. (Borenstein & Koren, 1991).

Para (Jerome Barraquand et al., 1997) la planificación de trayectorias tiene como fin principal crear algoritmos que permitan establecer caminos considerando restricciones en los movimientos de robots móviles, como a su vez la dinámica del entorno trabajado. Las aplicaciones de Path Planing (planificación del trayecto) están orientadas a diferentes labores que interactúan con el ser

humano en diferentes ámbitos: i) salud: porque permite el apoyo de robots en tareas para personas de la tercera edad y personas con paraplejia, ii) militares: debido a que está enfocado en la supervisión de robots teledirigidos, autónomos y armamento inteligente, iii) industriales: monitoreo de robots con inteligencia artificial mediante el uso de robots móviles. (Ortiz & Santana, 2017).

Además (Hauser, Bretl, Latombe, Harada, & Wilcox, 2008), en su trabajo explica en su teoría que la autonomía de un robot móvil definida por alcanzar desde un punto inicial a un punto final se ve delimitada por los problemas de colisión con objetos fijos y móviles tanto en ambientes estructurados y no estructurados, sobre el cual debe determinar una ruta para llegar a la dirección definida, es decir encontrar el camino más corto que dicho robot pueda seguir. También se puede decir que la presencia de sistemas de robots múltiples en dominio de servicios es todavía limitada, acerca de extender sus horizontes a hogares para contribuir con las personas, debido al funcionamiento autónomo de múltiples robots móviles en escenarios reales, lo cual es difícil por la limitación sensorial y altos grados de complejidad computacional (Chinnaiah, Dubey, Vineela, Bindu, & Babu, 2016).

Y por último se realizó un prototipo de robot cuadrúpedo anteriormente por los investigadores de este proyecto, el desarrollo de hardware y software se basó en la plataforma Arduino y se controla de forma remota desde una aplicación de Android con la que se ejecutaba los comandos de movimiento, es decir el robot era tele operado, no tenía ningún tipo de inteligencia, todo lo controlaba una persona. El sistema de comunicación se basa en una conexión Android-Bluetooth que permitía una comunicación confiable en distancias cortas; se decidió continuar con este trabajo debido a que se vio mejoras y complementos que se le podían realizar al mismo, estas se mencionan

en el alcance del proyecto, además se vio la importancia que este podía llegar a tener, la cual se menciona a en el siguiente punto.

1.2 Justificación e importancia

El proyecto está centrado en la realización y desarrollo de un sistema de control y navegación para un robot cuadrúpedo en entornos no definidos con perspectiva cenital, la cual es una perspectiva perpendicular al plano de movimiento del robot y realizada de arriba hacia abajo, al igual que un satélite que fotografía la superficie terrestre.

En diversos campos es necesario el uso de robots capaces de moverse de un punto a otro en zonas irregulares para el logro de sus objetivos, zonas en donde los robots articulados toman importancia, ya que estos presentan una ventaja considerable en su movilidad, contrario a los robots con ruedas, los cuales presentan mejor desempeño en zonas planas. Por esta y más razones, constantemente se realizan investigaciones centradas en este campo, basando su búsqueda en la evasión de obstáculos, toma de decisiones propias e inteligencia artificial la que le permitirá aprender de su propio entorno y desarrollarse con autonomía.

Realizar tareas de evasión de obstáculos es un problema cuya búsqueda de solución presenta gran importancia en el campo de la robótica. Si un robot quiere dirigirse de un lugar a otro en entornos reales, es necesario detectar aquellas zonas u objetos difíciles de atravesar. Casi para todos los problemas de navegación, el mundo o el entorno en el que el robot se va a desplazar pueden ser modelados como un plano; por lo que cualquier punto que no pertenezca a este, puede considerarse un obstáculo. La mayoría de los algoritmos de evasión de obstáculos utilizan sensores de distancia, como sonares, láseres y radares; estos sensores son adecuados para la tarea de detectar obstáculos y pueden ser utilizados para diferentes propósitos, ya que pueden medir la distancia entre el obstáculo y el robot. Sin embargo, estos sensores presentan ciertos inconvenientes, los que

contribuyen a asegurar que ninguno puede realizar la tarea de detección de obstáculos a la perfección. Los sonares tienen un bajo costo, pero por lo general tienen un ángulo de resolución muy estrecho. El láser y los radares proporcionan una mejor resolución, pero son más complejos y costosos. Los sensores visuales son una solución alternativa para evitar obstáculos, y su popularidad ha aumentado significativamente en el campo de la robótica móvil (Bertozzi et al., 1996).

Es por ello que el enfoque de la presente investigación discute la evasión de obstáculos y el desarrollo de un sistema completamente autónomo en entornos no definidos en los que el robot debe tener la capacidad de tomar decisiones a través de un sistema de control. La toma de decisiones se dan de acuerdo con la información capturada del entorno en el que el robot se encuentra a través de la cámara como el sensor visual, desde una perspectiva cenital, para el análisis y procesamiento de imágenes para la evasión de obstáculos se usara el software OpenCV, diseñado para la eficiencia computacional, con un fuerte enfoque en aplicaciones en tiempo real, que proporciona un uso simplificado de su infraestructura para la visión computarizada y que se encuentra disponible para Python cuyo desarrollo se basa en bibliotecas de código abierto; esta última herramienta se utilizará como base para el desarrollo computacional del método de evasión de objetos y con esta información el robot móvil realiza una tarea determinada.

1.3 Alcance del proyecto

Cabe mencionar que nuestro trabajo no está enfocado en el diseño mecánico del sistema sino utilizar diseños existentes en la literatura que han sido probados y se los usara para la impresión en 3D, posteriormente para ser armado.

Mediante el presente proyecto de investigación, se plantea el desarrollo del algoritmo de movimiento tridimensional y el sistema de navegación para un robot cuadrúpedo en entornos no

definidos desde una perspectiva cenital. Inicialmente se realizará el estudio del marco teórico sobre los diferentes tipos de robots móviles, tipos de clasificadores para la detección de obstáculos y algoritmos de planificación de ruta, además se recolectará la información técnica necesaria para obtener la lista de elementos y sus características de funcionamiento para la construcción del Robot móvil.

En la siguiente etapa del proyecto se desarrollará un algoritmo que contiene la lógica de movimiento tridimensional de bajo nivel para desplazarse de un punto a otro, centrándose en la investigación de la cinemática del robot cuadrúpedo, el problema cinemático consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot cuadrúpedo para que su extremo se posicione y oriente según una determinada localización espacial (Grupo Aurova, 2016), esto se lo realizará utilizando la matriz de Denavit Hartenberg, el cual nos sirve para definir el movimiento de una de las articulaciones y replicarle en las restantes, para esto se definen 3 grados de libertad en cada articulación enviando este valor de consigna a los servomotores, cabe mencionar que la simulación del movimiento de la articulación es realizado en el software Matlab.

La unidad de control del robot para cada articulación será comandada por un microcontrolador. Las señales PWM de las salidas de la tarjeta del microcontrolador serán transmitidas a través de una placa PCB de conexión a servomotores. En Total, el robot tendrá 12 motores, los cuales pertenecen a las extremidades delanteras y posteriores. El algoritmo de movimiento será realizado mediante ciertas alternativas entre las cuales tenemos:

- Comandos de Movimiento: Izquierda, derecha, adelante, atrás, con un número de grados y distancias fijas.
- Ángulo de giro: Se enviará al robot un determinado valor de ángulo al que deberá girar.

- Giro a determinada posición (Set Point): El robot recibirá una posición determinada a la que se debe ubicar.

Terminada la etapa de simulación de movimiento se realizará el análisis y procesamiento de imágenes para la evasión de obstáculos mediante OpenCV, con la cual el desarrollo se basa en bibliotecas de código abierto, esta última herramienta se utilizará como base para el desarrollo computacional del método de objetos de evasión. Este trabajo tiene el propósito de evitar colisiones entre un robot móvil y obstáculos estáticos que tienen un color o forma específicos.

Primero se obtiene el entorno con una cámara 2D central, utilizando morfología matemática para la detección de objetos, sobre ese entorne capturado se desarrollará un algoritmo de planificación de caminos automático que permita obtener los puntos en el espacio de trabajo de una trayectoria solución en un espacio libre de colisión.

Además, para realizar la Planificación de ruta en entornos no definidos con perspectiva cenital nos enfocaremos específicamente en los métodos probabilísticos, existen varios algoritmos que se van a explorar en este trabajo, algoritmos como:

- A*, es un algoritmo de búsqueda el cual traza una ruta eficientemente dirigida entre múltiples puntos, llamados nodos. Goza del uso extenso debido a su funcionamiento y exactitud.
- PRM, la idea básica detrás de este algoritmo es tomar muestras aleatorias del espacio en el que se encuentra del robot, probándolas si están en el espacio libre, y utilizar un planificador local para intentar conectar estas configuraciones a otras configuraciones próximas.
- RRT, el cual esta diseñado para buscar eficientemente espacios no convexos de alta dimensión mediante la construcción aleatoria de un árbol que rellena el espacio libre en cada iteración.

De estos algoritmos el que será utilizado es RRT debido a que este, en sistemas prácticos de planificación de movimientos, puede pre-procesar el gráfico para lograr un mejor rendimiento.

Cabe destacar que el algoritmo para la planificación de movimiento será desarrollado en su totalidad por los investigadores del proyecto.

Una vez definido el algoritmo se procederá a optimizar el mismo, eligiendo diferentes parámetros que buscan lograr un equilibrio entre velocidad y resolución del camino que el robot tendrá que seguir, con el objetivo de evitar obstáculos con una mayor facilidad.

La etapa final del proyecto se enfoca en la realización de pruebas de funcionamiento, con lo cual se busca analizar la precisión y velocidad de respuesta del sistema, para así obtener resultados reales que puedan ser posteriormente analizados.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un sistema de navegación mediante planificación de trayectoria para un robot cuadrúpedo en entornos no definidos con perspectiva cenital.

1.4.2 Objetivos Específicos

- Determinar el sistema de navegación autónoma, basado en el movimiento estimado con algoritmos de visión por computadora para robot cuadrúpedo.
- Evaluar técnicas de detección de obstáculos y objetos basados en diferentes métodos de vision para robots móviles en entornos no definidos con perspectiva cenital.
- Analizar los resultados obtenidos para optimizar el algoritmo de planificación de trayectoria o ruta.
- Evaluar los resultados obtenidos de el algoritmo de planificación de trayectoria en el entorno simulado con aquellos obtenidos en el funcionamiento del robot en un entorno físico.

1.5 Descripción del proyecto de investigación

En el presente proyecto de investigación se desarrolla un sistema de navegación mediante planificación de trayectoria para un robot cuadrúpedo en entornos no definidos con perspectiva cenital, en donde el robot y los obstáculos son determinados mediante un sistema de visión por computadora. Se presentan los procedimientos utilizados para el diseño de hardware, diseño de software, construcción y control del robot cuadrúpedo, SpiderBot, para posteriormente establecer un ambiente controlado en el cual se va a desplazar desde un punto de partida a un punto objetivo, identificando y esquivando la presencia de obstáculos en los diferentes escenarios. El desarrollo de este proyecto de investigación está enmarcado en los capítulos que se muestran a continuación:

1.5.1 Capítulo I - Introducción

En este capítulo se describen las características generales del proyecto de investigación, es decir, los antecedentes, justificación, alcance y objetivos, los cuales fundamentan y argumentan las razones de por qué es favorable llevar a cabo el proyecto y cuáles son los beneficios que resultan del mismo, además se señala qué es lo que se va realizar para orientar el proceso de desarrollo completo del proyecto, estableciendo los medios de apoyo necesarios para desarrollar un sistema de navegación mediante planificación de movimiento para un robot cuadrúpedo en entornos no definidos con perspectiva cenital.

1.5.2 Capítulo II – Marco Teórico

Las actividades por realizar en la etapa de revisión del marco teórico están enfocadas a la búsqueda y recolección de información actualizada y relevante de estudios previos elaborados que se relacionan con la navegación autónoma en entornos no definidos de Robots móviles. Esta información será clave para determinar la importancia y el impacto que el proyecto produce para

solucionar problemas relacionados en el campo militar, médico o en entornos catastróficos en los cuales es indispensable el uso de robots autónomos.

1.5.3 Capítulo III – Implementación del Robot Cuadrúpedo

En esta etapa se desarrolla la construcción del robot cuadrúpedo, realizando en primer lugar un análisis de requerimientos, para que, posteriormente, se haga una selección de hardware entre varias opciones necesarios para el funcionamiento de SpiderBot, esta selección se la realizó mediante el Proceso analítico jerárquico (AHP), entre las cuales se varió procesador, batería, servomotores, módulo de comunicación.

Además, se presenta el circuito del robot, elaborado de manera profesional, en donde todos los componentes electrónicos se unen e interconectan entre si. Se explica cada uno de los bloques requeridos para el funcionamiento del cuadrúpedo y, por último, se muestra el diseño elaborado en tres dimensiones para su posterior impresión en 3D.

1.5.4 Capítulo IV – Implementación de Algoritmos Necesarios para el Proyecto

En esta etapa del proyecto se desarrollará la investigación sobre el algoritmo de movimiento necesario para SpiderBot, también se realiza el análisis del modelo cinemático de las partes que conforman al robot cuadrúpedo, utilizando el método de Denavit Hartenberg, para el cual se define 3 grados de libertad en cada una de sus articulaciones definiendo el movimiento para cada articulación.

Por otro lado, se realizarán los algoritmos de visión necesarios para poder detectar a los obstáculos mediante morfología, y a el robot mediante color, por último, hará una revisión de los algoritmos para la planificación de trayectoria.

1.5.5 Capítulo V –Evaluación, Pruebas y Análisis de Resultados

Etapa destinada a realizar varios experimentos para el contraste del desempeño de los algoritmos en un ambiente con obstáculos, el cual permita analizar el funcionamiento de todo el sistema y poder valorar la contribución obtenida gracias al proyecto.

1.5.6 Capítulo VI – Conclusiones y recomendaciones

En la etapa final del proyecto se redactarán las conclusiones y recomendaciones producto de los resultados obtenidos con el desarrollo del proyecto, además de presentar trabajos futuros que puedan aportar a la comunidad científica en temas relacionados con la implementación de algoritmos de planificación de movimiento en entornos no definidos con vista cenital mediante el uso de la robótica móvil.

CAPITULO II

MARCO TEÓRICO

2.1 Introducción

Muchos de nosotros nos preguntamos cómo funciona un robot, qué tipos de tecnologías usan y por qué necesitamos en nuestra vida. La robótica como disciplina ha estado activa durante más de 50 años y comprende una extensa gama de subcampos disjuntos. (Crabbe, 2001). Los conceptos primordiales de la robótica son uno de los temas excepcionales que se manejan como un todo gracias a la gran diversidad de tecnologías científicas que incorpora. Sus raíces incluyen muchas disciplinas de ingeniería y ciencia, por ejemplo; ingeniería mecánica, ingeniería eléctrica, ciencias de la computación, electrónica, sensores, actuadores, inteligencia artificial hasta ciencias cognitivas y sociales. Es un área multidimensional que aprovecha todos los estudios de ingeniería que existen en nuestra vida.

Históricamente, el subcampo más avanzado de la robótica es la robótica de ingeniería, caracterizada por una representación matemática precisa de las propiedades físicas del robot (cinemática) y técnicas para ejecutar secuencias proyectadas de movimientos (teoría de control), dicho punto de vista es exitoso para controlar un robot manipulador que realice acciones precisas y repetibles, lo que los hace ideales para la industria, pero la robótica de ingeniería no afronta el inconveniente de permitir al robot elegir acciones en situaciones inesperadas. Esto se convierte en un problema una vez que tiene un sistema de locomoción (Crabbe, 2001). La inteligencia artificial (IA) en la robótica se desarrolló en paralelo a la robótica de ingeniería para solucionar los inconvenientes de toma de decisiones que un robot debe afrontar. En general, se considera que la robótica AI comenzó en la década de 1960 con las técnicas de planificación de IA todavía en uso hoy en día. (Murphy, 2000)

Un robot que utiliza estas técnicas detecta el entorno, construye un modelo del entorno, busca un plan óptimo y luego actúa en función de ese plan. El primer desafío es la locomoción en sí misma que se relaciona con la tecnología de la movilidad, la cual se centra en como un robot móvil puede moverse sin supervisión a través de escenarios del mundo real para cumplir con sus tareas y las alternativas de mecanismos de locomoción (Siegwart & Nourbakhsh, 2004).

2.2 Robots

La robótica tiene sus inicios con Harold Roselund de la compañía Devilviss, quien fue el creador del primer brazo manipulador, en el año 1938 que se usaba para pintar ¨spray¨, pero al no existir en ese tiempo computadoras digitales la programación para una actividad sencillas era altamente compleja, por lo cual este robot no tuvo éxito, pero a partir de este invento se buscó que los trabajos se automaticen y se los realicen en serie (Madrigal & Idiarte, 2002).

La robótica hasta la fecha ha conseguido su mayor éxito en el mundo de la fabricación industrial, la demanda de una mayor productividad industrial ha llevado a un uso creciente de manipuladores robóticos controlados por computadora, los robots manipuladores, comprenden una industria de 2 mil millones de dólares (Sunada & Dubowsky, 1983). Sin embargo, a pesar de todos sus éxitos, estos robots comerciales adolecen de una desventaja esencial: la falta de movilidad. Un manipulador fijo tiene un rango de movimiento restringido que depende de dónde esté atornillado, en contraste a un robot móvil que podría movilizarse por toda la planta de fabricación, aplicando de carácter flexible sus talentos donde sea más efectivo (Siegwart & Nourbakhsh, 2004).

Un sistema robótico necesita mecanismos de locomoción que le permitan moverse sin límites en su entorno. Actualmente hay una gran variedad de formas posibles de desplazarse, por lo que la selección del enfoque de un robot para la locomoción es un aspecto importante del diseño del mismo. Existen robots de investigación que pueden caminar, saltar, correr, deslizarse, patinar,

nadar, volar y, por supuesto, rodar. En la Figura 1 se muestra la mayoría de estos mecanismos de locomoción los cuales se inspiraron en sus contrapartes biológicas.

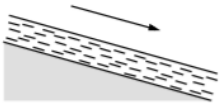
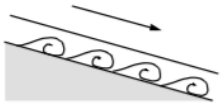
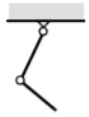
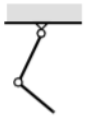
Type of motion	Resistance to motion	Basic kinematics of motion
Flow in a Channel 	Hydrodynamic forces	Eddies 
Crawl 	Friction forces	Longitudinal vibration 
Sliding 	Friction forces	Transverse vibration 
Running 	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum 
Jumping 	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum 
Walking 	Gravitational forces	Rolling of a polygon (see figure 2.2) 

Figura 1. Mecanismos de locomoción basados en sistemas biológicos.
Fuente: (Siegwart & Nourbakhsh, 2004)

Los sistemas biológicos tienen éxito en moverse a través de una amplia variedad de entornos. Por lo tanto, puede ser conveniente copiar su selección de mecanismos de locomoción.

Sin embargo, también existen mecanismos de locomoción aún más inusitados los cuales son necesarios debido a ambientes hostiles como Marte, a manera de ejemplo, en la Figura 2 se muestra el robot móvil Sojourner el cual fue utilizado durante la misión Pathfinder para explorar Marte en el verano de 1997, fue teleoperado casi por completo desde la Tierra ya que algunos sensores a bordo permitieron la detección de obstáculos. (Iagnemma & Dubowsky, 2000)

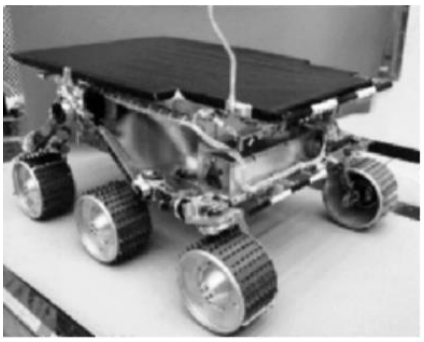


Figura 2. Robot móvil Sojourner
Fuente: (Iagnemma & Dubowsky, 2000)

En entornos peligrosos e inhóspitos, incluso en la Tierra, estos sistemas teleoperados han ganado popularidad, En estos casos, las complejidades de bajo nivel del robot a menudo hacen que sea imposible para un operador humano controlar directamente sus movimientos. El ser humano realiza actividades de localización y cognición, pero se basa en el esquema de control del robot para proporcionar control de movimiento.(Siegwart & Nourbakhsh, 2004)

Por ejemplo, el robot para caminar de Plustech que se muestra en la Figura 3 proporciona una coordinación automática de las piernas mientras que el operador humano elige una dirección general de desplazamiento.

Figura 3. Robot Hexapod, máquina taladora hecha por Plustech
Fuente: (Plustech.fi, n.d.)

Existen muchas publicaciones que mencionan que la locomoción con patas es la mejor forma de moverse a través de un terreno variable, en comparación con el uso de ruedas. La superficie de un terreno puede ser irregular, blanda, fangosa y generalmente no estructurada. Las piernas en el ambiente biológico demuestran una ventaja significativa en tales situaciones. (Zielinska & Heng, 2003). Es sobre esta base que comienza el motivo para diseñar y desarrollar una máquina de caminar altamente flexible para imitar a nuestros primos biológicos de cuatro patas.

2.3 Robots N-Podos

Los robots que se basan en la locomoción con patas se caracterizan por tener una gran relación con el suelo. Las ventajas primordiales de este tipo de robots son: tener una gran adaptabilidad y maniobrabilidad en terrenos irregulares (Siegwart & Nourbakhsh, 2004)

Los robots de locomoción con patas se pueden mover con gran facilidad en terrenos naturales, ya que usan puntos de apoyo discretos para cada pie, en comparación con los robots con ruedas, que requieren de una superficie de soporte continuo. Por lo tanto, son muy buenos para movilizarse en terrenos irregulares, gracias a dos características que poseen:

- Variar la configuración de sus patas para adaptarse a las irregularidades de una superficie.
- Los pies pueden establecer contacto con el suelo en los puntos seleccionados de acuerdo con las condiciones del terreno.

Por estas razones, el uso de patas en robots es la mejor opción para la locomoción en terreno irregular. El uso de múltiples grados de libertad en las articulaciones de las piernas, permite a los robots con patas cambiar su rumbo sin tener que realizar una acción de deslizamiento. Las piernas de un robot caminando causan mucho menos daño al terreno que un robot de ruedas.

Una gran ventaja de los robots con patas es la facilidad que tienen para manipular objetos en el entorno con mucha habilidad; y como pueden presentar un número redundante de piernas tienen la

gran capacidad de mantener el equilibrio estático y continuar su locomoción incluso con una o más de sus piernas dañadas (Hirose & Kato, 1998). Las piernas pueden usarse no solo para la locomoción del robot, sino también para mover la base del cuerpo con la finalidad de que trabaje como soporte de un manipulador (Garcia, Estremera, & de Santos, 2003).

Como ejemplo de robot de patas, se presenta una robot bípedo cuya estructura está inspirada en los dinosaurios, el cual utiliza la cola para ayudarse a mantener el equilibrio durante la locomoción y durante las tareas de manipulación que el robot realiza con su cuello, para que el robot pueda pararse sobre ella , haciendo un trípode de soporte estable (Takita, Hodoshima, & Hirose, 2000).

Las principales desventajas de los robots por patas incluyen la potencia y la complejidad mecánica.

2.3.1 Robots Monópodos

La locomoción en este tipo de autómatas se basa en saltos por lo que también se los conoce como robots de saltos. La masa corporal es especialmente importante para las máquinas para caminar, y la única extremidad minimiza la masa acumulativa de las piernas. Se requiere coordinación de la pierna cuando un robot tiene varias patas, pero con una pierna no se necesita dicha coordinación.

Un robot de una sola pierna requiere solo una secuencia de contactos individuales, lo que lo hace apto para los terrenos muy irregulares.

El mayor reto en la creación de un autómata de una sola pierna es el equilibrio. Para un sistema que posee una sola pierna, el caminar estático no solo es imposible, sino que la estabilidad estática cuando está parado también es imposible. El robot lo que tiene que hacer es equilibrarse cambiando su centro de gravedad o impartiendo fuerzas correctoras. Por lo tanto, el robot exitoso de una sola pierna debe ser dinámicamente estable (Siegwart & Nourbakhsh, 2004).

Un ejemplo de estos robots comparado con un ser vivo es la del canguro o con bípedos, que alternan entre uno o ningún pie en contacto con el suelo.

Matsuoka fue el primero en construir una máquina de acuerdo con estos conceptos, lo que significa, que su objetivo era modelar los saltos cíclicos en la locomoción humana. Para pode cumplir con este objetivo, Matsuoka formuló un modelo, el cual consiste en un cuerpo y una pierna ingrávida para poder simplificar el problema, y consideró que la duración de la fase de soporte era corta en comparación con la fase de vuelo balístico(Tenreiro Machado & Silva, 2006).

Raibert, construyó en la Universidad Carnegie Mellon (CMU) un robot de salto. Este sistema, formado por un cuerpo y una sola pierna como se puede observar en la Figura 4, necesitaba saltar continuamente para mantener el equilibrio. Esta máquina realiza correcciones continuas a la actitud del cuerpo y a la velocidad del robot ajustando el ángulo de la pierna con respecto al cuerpo. La actuación es hidráulica, incluida la extensión longitudinal de alta potencia de la pierna durante la postura para saltar hacia atrás en el aire. Aunque son potentes, estos actuadores requieren una bomba hidráulica grande, fuera del tablero, para conectarse en todo momento.

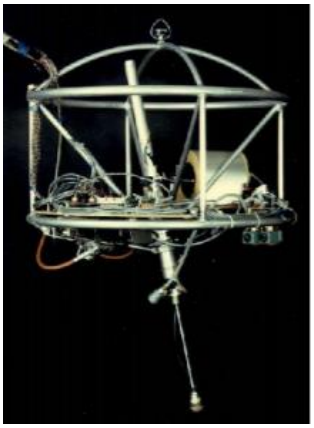


Figura 4. Robot saltador de Raibert
Fuente: (Tenreiro Machado & Silva, 2006)

También se están desarrollando robots monópodos que utilizan el principio de salto para su locomoción, adoptando mecanismos que les permiten mantener el equilibrio cuando se detienen,

es decir, pies con una geometría especial (Iida, Dravid, & Paul, 2002). Se puede pensar que no hay aplicaciones prácticas para los robots con esta configuración. Sin embargo, estos robots permiten saltar obstáculos o posicionarse en lugares donde no exista espacio disponible para colocar los pies, sin tener la necesidad de preocuparse por la estabilidad estática.

Una de las mejores aplicaciones de estos son en la exploración de pequeños cuerpos celestes, donde los robots con patas y ruedas no pueden moverse con éxito, debido a la gravedad local reducida (Shimoda, Wingert, Takahashi, Kubota, & Nakatani, 2004).

2.3.2 Robots bípedos

La investigación en robots bípedos, en comparación con los que tienen algunas patas, ha avanzado muy lento por la gran dificultad de poder definir un control estable porque son más exigentes con respecto a su equilibrio dinámico (Katic & Vukobratovic, 2002).

En la década de 1960, la Universidad de Waseda, en Japón, ha desarrollado una serie de sistemas bípedos controlados por computadora. En 1969 Ichiro Kato desarrolló el bípedo WAP-1Figura 5 en el Humanoid Research Laboratory. Este robot tenía músculos de goma artificiales, accionados neumáticamente, y la locomoción bípeda se logró mediante la reproducción de movimientos previamente enseñados. La principal limitación inicial de estas máquinas era su baja velocidad, que requería 90 segundos para completar un paso. Los últimos avances permitieron conseguir velocidades cercanas a las alcanzadas por el ser humano.

Figura 5. Robot bípedo
WAP-1
Fuente:(Tenreiro Machado & Silva, 2006)

A principios de la década de 1980, Kato y sus compañeros de trabajo construyeron el bípedo WL-9DR el cual tenía 10 grados de libertad que se activaban hidráulicamente y dos pies que eran grandes Figura 6.

Este robot tenía una locomoción estática, el cual se movía a lo largo de una trayectoria planificada previamente, a fin de mantener el centro de gravedad dentro de la base de apoyo suministrada por el pie de apoyo.

Figura 6. Robot bípedo
WL-9DR
Fuente:(Tenreiro Machado & Silva, 2006)

En los últimos diez años se han demostrado robots bípedos muy exitosos, que pueden realizar acciones como: correr, saltar, subir y bajar escaleras e incluso hacen trucos aéreos, como saltos mortales. En el sector comercial, Honda y Sony han logrado avances significativos en cuanto a robots bípedos; ambas empresas han diseñado articulaciones pequeñas y motorizadas que logran un rendimiento de potencia a peso inaudito en los servomotores disponibles comercialmente. Estos nuevos servos que podrían decirse "inteligentes" proporcionan no solo una actuación fuerte sino también una actuación conforme mediante detección de par y control de lazo cerrado.

Un modelo de robot bípedo realizado por Sony, SDR-4X II, se muestra en la Figura 7. Este modelo actual es el resultado de una investigación que inicio en 1997 con el objetivo básico de entretenimiento en movimiento y entretenimiento de comunicación es decir para poder cantar y bailar. Este posee treinta y ocho grados de libertad tiene siete micrófonos para una localización precisa del sonido, reconocimiento de personas basado en imágenes, reconstrucción de mapa de profundidad estéreo en miniatura a bordo y reconocimiento de voz limitado. El SDR-4X II es relativamente pequeño, con una altura de 58 centímetros y un peso de solo 6,5 kilogramos (Siegwart & Nourbakhsh, 2004).

Figura 7. Robot Sony SDR-4X II
Fuente:(Siegwart & Nourbakhsh, 2004)

Otro ejemplo de robot bípedo es el WABIAN (WAseda BIpedal humANoid) (Figura 8) que se han desarrollado en Japón. El desarrollo de estos, tenía como objetivo principal la creación de un robot antropomórfico que presenta patrones de pensamiento y comportamiento similares a los del ser humano. Se pretendía lograr que sea capaz de interactuar de forma natural con los humanos, es decir, poder hablar y presentar emociones(Yamaguchi, Soga, Inoue, & Takanishi, 1999).

Para darle un poco de más parentesco con el ser humano, en este robot las articulaciones de la cadera se activaron de forma antagonista y con rigidez variable, de forma similar a la activación articular humana que es accionada por dos o más grupos musculares que presentan características idénticas a resortes no lineales.

En términos de capacidades de locomoción, este robot fue capaz de avanzar y retroceder, bailar de una manera dinámica agitando sus brazos y caderas y transportar algo de carga, usando sus brazos(Yamaguchi et al., 1999).

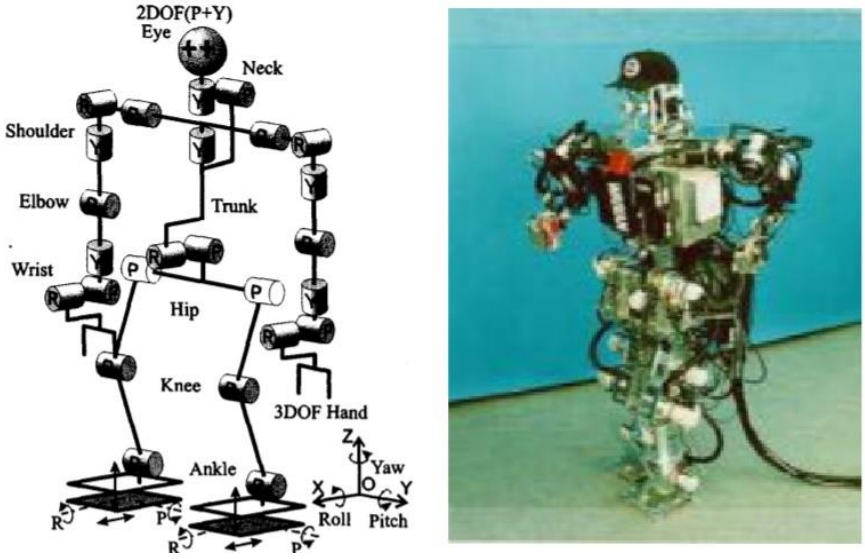


Figura 8. Robot WABIAN y su estructura
Fuente:(Yamaguchi et al., 1999)

2.3.3 Robots cuadrúpedos

Aunque permanecer de pie sobre cuatro patas es relativamente estable, la acción de caminar sigue siendo difícil porque para permanecer estable, el centro de gravedad del robot debe moverse activamente durante la marcha.

Sony desarrolló un autómata de cuatro patas llamado AIBO (Figura 9). Para crearlo, Sony desarrolló un sistema operativo que funcionaba casi en tiempo real y nuevos servomotores con engranajes que tienen un par de torsión suficientemente alto para soportarlo, pero que pueden manejarse de regreso para mayor seguridad(Siegwart & Nourbakhsh, 2004).

Figura 9. Robot AIBO SONY
Fuente:(Siegwart & Nourbakhsh, 2004)

Otro ejemplo de robot cuadrúpedo es el (LAVA) (Figura 10) desarrollado en el Robotics Research Center, NTU, Singapur. Para disminuir el peso y aumentar la producción de energía de cada servomotor, cada pata con sus tres grados de libertad se impulsa a través de un engranaje diferencial inverso. Dos motores están ubicados en la sección de la cadera y el tercero está en la sección de la rodilla. Los motores de cadera trabajan colectivamente para generar el balanceo y la elevación combinados de la pierna. Cada uno de los servomotores está acoplado a través de un sistema de engranaje sinfín para garantizar un bloqueo estable del sistema que permita que los

motores se apaguen cuando el vehículo con patín solo necesita pararse, lo que ahorra energía eléctrica(Zielinska & Heng, 2003).

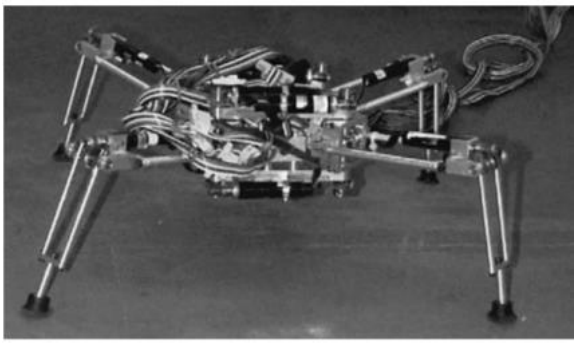


Figura 10. Robot cuadrúpedo (LAVA)
Fuente:(Zielinska & Heng, 2003)

2.3.4 Robots hexápodos

Un robot hexápodo es un autómata que camina sobre seis patas. Puede estar estáticamente estable en tres o más patas, por lo tanto tiene una gran flexibilidad en la forma en que se puede mover. Si una de sus patas se desactiva, todavía podrá moverse. Además, no todas las patas son necesarias para la estabilidad; otras patas son libres de alcanzar nuevas ubicaciones de pie o manipular una carga útil (Krishna, Nandanan, Kumar, Srihari, & Sivraj, 2014).

Los robots hexápodos se pueden dividir en dos categorías rectangulares y hexagonales.

- Los hexápodos rectangulares tienen un cuerpo rectangular y dos grupos de tres patas colocados simétricamente a lo largo de sus dos lados.
- Los hexápodos hexagonales tienen un cuerpo hexagonal o circular, con las piernas distribuidas uniformemente. Las patas de estos, imitan a las patas de los insectos que caminan, con tres grados de libertad cada una, lo que significa que hay tres puntos donde se puede mover la pierna, o tres articulaciones.

Las características tales como mayor capacidad de carga útil, capacidad de navegación y movimientos de precisión hacen que un androide hexápodo totalmente equipado sea altamente

confiable y versátil. Se puede personalizar para su uso efectivo como robots médicos, máquinas herramientas y automatización de fabricación (Krishna et al., 2014).

Dado que puede atravesar terrenos muy irregulares con alta estabilidad, se puede usar un hexápodo completamente equipado para transporte de productos de pequeña escala para fines militares.

Las configuraciones de seis patas son muy populares en cuanto a la robótica móvil debido a su estabilidad estática durante la marcha, reduciendo así la complejidad del control.

Por lo general estos robots tienen en cada pierna tres grados de libertad, incluida la flexión de la cadera, la flexión de la rodilla y la abducción de la cadera. Genghis (Figura 11) es un robot que tiene seis patas, cada una de las cuales tiene dos grados de libertad provistos por los servos. El funcionamiento consiste únicamente en la flexión de la cadera y la abducción de la cadera, tiene menos maniobrabilidad en terrenos abruptos, pero funciona bastante bien en terreno llano (Siegwart & Nourbakhsh, 2004).

Los insectos, son sin duda las criaturas locomotoras más exitosas en la tierra, que se caracterizan principalmente por tener la capacidad de atravesar todo tipo de terreno con seis patas, incluso al revés.

Todavía existe una brecha entre los insectos de seis patas con los robots de seis patas, esto no se debe a la falta de suficientes grados de libertad. Por el contrario, los insectos combinan un pequeño número de grados de libertad activos con estructuras pasivas, tales como púas microscópicas y almohadillas con textura, que aumentan significativamente la fuerza de agarre de cada pierna.

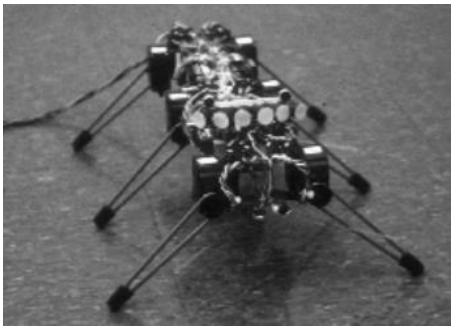


Figura 11. Robot Genghis
Fuente:(Siegwart & Nourbakhsh, 2004)

2.4 Percepción Visual

Visión por computadora es una herramienta ampliamente usada en los sistemas electrónicos para emular la visión que el hombre tiene. Reemplazando al ojo humano con una cámara y al cerebro de la persona con un procesador (W. G. Aguilar & Angulo, 2014a, 2014b, 2016; W. G. Aguilar, Angulo, & Pardo, 2017).

El ser humano puede observar al mundo en tres dimensiones, cada persona logra ver diferentes características de su entorno de manera vívida con esta perspectiva tridimensional, se pueden diferenciar formas, bordes, translucidez, sombras, patrones de luz y así conseguir distinguir los objetos que lo rodean sin ningún tipo de esfuerzo (Szeliski, 2011), es por esto que los humanos obtenemos la mayor parte de nuestras entradas sensoriales a través del sistema visual (Parker & Terzidis, 2011).

En los últimos quince años ha cambiado muy poco la tecnología en el campo de la visión y el procesamiento de imágenes. Es verdad que la parte teórica en visión se ha vuelto más sofisticada y los métodos de visión en tres dimensiones han mejorado positivamente, que algunos sistemas de visión en robots han conseguido cosas que no se podían en el pasado (Parker & Terzidis, 2011), además que el reconocimiento facial ha sido llevado a un nuevo nivel pero aun es un campo activo y en desarrollo (Bradski, 1998). Pero, el reconocimiento hablando en bajos costos sigue siendo

"barato", y aún no está a un nivel en el que pueda utilizarse de manera confiable y aceptable en la mayoría de los casos (Parker & Terzidis, 2011). Como lo mencionado anteriormente nosotros, como humanos, percibimos la estructura del mundo con mucha facilidad, pero, a pesar de todos los esfuerzos por mejorar el campo de la visión artificial, el sueño de tener una computadora para interpretar una imagen al mismo nivel que un niño de dos años sigue siendo difícil de alcanzar (Szeliski & Stockman, 2011).

Existen una infinidad de aplicaciones en las que se puede usar visión por computadora, una de ellas es la interfaz de usuario perceptual en donde se intenta que la computadora tenga la capacidad de detectar y producir señales análogas de los sentidos humanos, como permitir que las computadoras perciban sonidos de su entorno y produzcan voz para el mismo, asimismo dando a las computadoras sentido del tacto y fuerza de retroalimentación, y en este caso, dando a las computadoras la capacidad de ver, pero estos algoritmos de visión por computadora que pretenden formar parte de una interfaz de usuario perceptual deben ser rápidos y eficientes (Bradski, 1998).

Otra aplicación que se ha dado a la visión artificial es en los Deportes, en los cuales es evidente que juegan un papel importante en la sociedad moderna, debido a que los deportes permiten interacción entre personas sin importar edad, estatus social, etc. Entonces, la visión puede a ayudar a dar un valor agregado a la experiencia general del usuario, este valor agregado puede ser en estadísticas (Moeslund, Thomas, Hilton, Carr, & Essa, 2017), hablamos de este tipo de visión por computadora, debido a que usualmente en deportes se utilizan cámaras estacionarias al igual que en este trabajo.

Además de esto la visión por computadora da una nueva perspectiva sensorial y ha desempeñado y seguirá desempeñando un papel importante en el aumento de la autonomía tanto de las naves espaciales, así como de los vehículos robóticos utilizados en la Tierra (Matthies et al., 2007). Un

ejemplo de ello es el uso de esta en Marte, en la misión MER del 2004, en la cual dos vehículos exploradores idénticos, Spirit y Opportunity, como se muestra en la Figura 12, aterrizaron para buscar pistas geológicas sobre, si parte de Marte, tenía anteriormente ambientes lo suficientemente húmedos como para ser hospitalarios para la vida (Matthies et al., 2007), en este trabajo se utilizaron algoritmos de visión por computadora para odometría, visión estereoscópica, navegación móvil y el seguimiento de características para la estimación de velocidad horizontal para los aterrizadores.

Figura 12. MER (izquierda) con Sojourner de la misión Mars Pathfinder de 1997 (derecha).
Fuente: (Matthies et al., 2007)

Son amplios y bastos los algoritmos de visión por computadora existentes, cada uno de ellos puede realizar diferentes tareas, desde el tratamiento inicial de la imagen para poder obtener una mejor calidad para su procesamiento, hasta las etapas próximas como son la detección de color y formas. Usualmente los sistemas están divididos en varios pasos para realizar el proceso de conexión entre los programas y algoritmos en la computadora con la cámara, por ejemplo, en el trabajo realizado por (Firmanda & Pramadihanto, 2014) se hizo un análisis basado en la visión por computadora para el control del cursor a través seguimiento de objetos y detección de color, utilizando los pasos mostrados en la Figura 13.

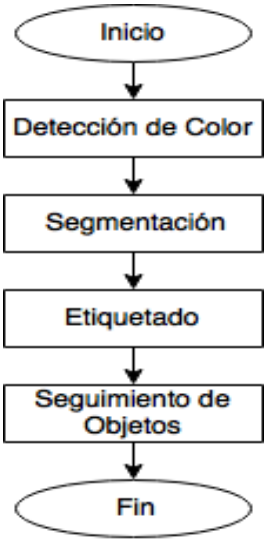


Figura 13. Diagrama del sistema para Seguimiento de Objetos
Fuente: (Firmanda & Pramadihanto, 2014)

Existen algunas formas de posicionar la cámara y poder dar visión al robot que se quiere controlar, una de ellas es con una perspectiva cenital, esto es, con la cámara colocada perpendicularmente al plano del suelo o al plano de movimiento, esta perspectiva fue utilizada por el trabajo realizado por (Antipov, Kokovkina, Kirnos, & Priorov, 2017) en el cual se definía la posición de autómatas y una pelota en un juego llamado robotsoccer; aquí mediante la cámara se distinguían patrones de color, pero antes, el algoritmo creado primero realizaba una detección de círculos. Luego, los círculos se clasifican usando la agrupación k-means que permite que el algoritmo se adapte en condiciones de iluminación dinámica, todo el sistema implementado se lo puede muestra en la Figura 14, donde se observa que existen dos cámaras para realizar la visión por computador, esto es debido a que existen dos equipos diferentes que tendrán que jugar dentro del robotsoccer.

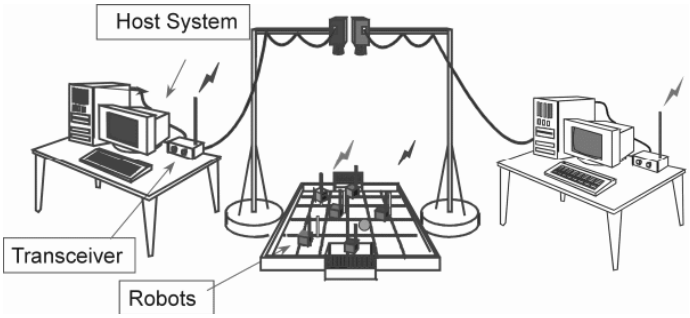


Figura 14. Esquema de Instalación de Robosoccer
Fuente: (Antipov et al., 2017)

Al igual que en el caso anterior, en este trabajo se controlará a un autómata, un tetrápodo, mediante una cámara con perspectiva cenital, la cual se utilizará la visión por computadora para realizar una navegación en entornos caóticos.

2.4.1 Detección de Obstáculos y Robots

El humano puede detectar obstáculos a través de estereovisión, capturando información de profundidad y distinguir un objeto de otro, mediante visión por computadora se puede imitar este funcionamiento, como el trabajo ejecutado por (Williamson, 1997) en el que se presenta un sistema de cámara 3D y algoritmos apropiados para el procesamiento de imágenes para proporcionar reconocimiento de peatones de acuerdo con las propuestas legislativas internacionales. El reconocimiento de peatones (W. G. Aguilar, Luna, Moya, Abad, Parra, et al., 2017; W. G. Aguilar, Luna, Moya, Abad, Ruiz, et al., 2017a, 2017b; W. G. Aguilar, Luna, Moya, Luna, et al., 2017; W. G. Aguilar, Luna, Ruiz, et al., 2017) es un requerimiento fundamental en la industria automotriz, que está obligada a tomar medidas para resguardar a los personas vulnerables de la vía pública.

Pero para realizar un reconocimiento en 3D es necesario un sistema de cámaras estéreo o cámaras RGB-D (W. G. Aguilar, Rodríguez, Álvarez, et al., 2017), que puedan captar profundidad de la imagen. En (Weichselbaum, Zinner, Gebauer, & Pree, 2013) se presenta un sistema de detección de obstáculos para un tren de operación autónoma en entornos de terreno abierto, en el cual, el sistema produce datos de profundidad densa en tiempo real desde un sistema de cámara

estéreo con una línea de base de 1,4 metros para cumplir con los requisitos de precisión para la detección confiable de obstáculos 80 metros más adelante.

Además de detecciones en 3D donde se utilizan usualmente dos cámaras, existen sistemas que realizan detecciones mediante una sola cámara, pero estas son de formas predefinidas, patrones, colores, bordes, líneas, etc. En la publicación *A vision-based autonomous detection scheme for obstacles on the runway* de (Zhou & Dong, 2017) se adopta un algoritmo basado en la descomposición y reconstrucción adaptativa del valor singular (AS VDR) para identificar y segmentar la región de una pista con precisión, combinando el algoritmo de detección de bordes Canny y el algoritmo de detección de línea Hough. En el artículo mencionado anteriormente se utilizan bordes para obtener los obstáculos, pero también se pueden utilizar colores, por ejemplo, en el algoritmo realizado por (Jerome Barraquand et al., 1997) consiste en detectar píxeles de apariencia diferente a la del suelo y clasificarlos como obstáculos como se observa en la Figura 15. Cada píxel de imagen individual se clasifica como si fuera un obstáculo o el suelo en función de su apariencia de color. El algoritmo funciona en tiempo real, proporciona una imagen de obstáculo de alta resolución y opera en una variedad de entornos.

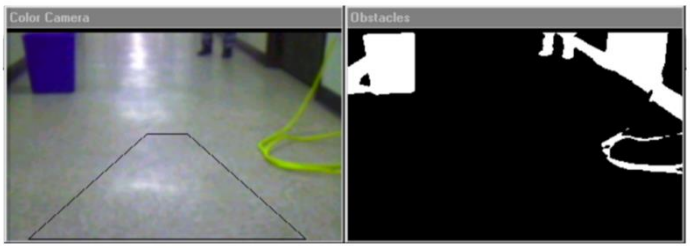


Figura 15. a) Imagen de color de entrada
b) Imagen de salida de obstáculo binaria
Fuente: (Jerome Barraquand et al., 1997)

A este proceso de tomar diferentes partes de la fotografía entregada por la cámara y separarla por características se llama segmentación de imagen, los algoritmos que realizan esto han ido

madurando hasta llegar al punto en que se han creado buenas segmentaciones. Se han realizado comparativas entre dos diferentes algoritmos de (Pantofaru & Hebert, 2005), el algoritmo de segmentación basado en el cambio promedio (Comaniciu & Meer, 2002) y un esquema de segmentación basado en gráficos (Felzenszwalb & Huttenlocher, 2004), y se concluyó que una combinación de ambos algoritmos entregó los mejores resultados, teniendo una estabilidad ligeramente mejorada.

Hay varias herramientas y librerías que permiten realizar estas segmentaciones a partir de formas, patrones, colores, bordes, líneas, una de estas Librerías es OpenCV del inglés *Open Source Computer Vision Library*. OpenCV es una biblioteca de software abierta y de aprendizaje automático, que fue construido para dar una plataforma común para aplicaciones de visión por computadora / visión artificial (OpenCV Team, 2018).

(Druzhkov et al., 2011) considera el problema de detección de objetos y proporciona una breve descripción de las implementaciones del sistema de detección de objetos con un modelo basado en partes discriminativamente entrenadas y un algoritmo de árboles que aumentan el gradiente. La biblioteca de OpenCV tiene más de 2500 algoritmos optimizados, y estos algoritmos se han utilizado en varios artículos para detectar y reconocer rostros (Goyal, Kartikey, & Kumar, 2011), identificar objetos (Guennouni, Ahaitouf, & Mansouri, 2015), clasificar acciones humanas en videos (Sharma, Kumar, Yadav, Sharma, & Bhardwaj, 2014), rastrear movimientos de la cámara, rastrear objetos en movimiento (Michalko, Onuška, Lavrin, Cymbalák, & Kainz, 2016), extraer modelos 3D de objetos (Yin & Yang, 2016), producir nubes de puntos 3D desde cámaras estéreo, etc. (OpenCV Team, 2018).

En este documento se utiliza OpenCV para la detección de obstáculos, asumiendo que los mismos son rectángulos, es decir se realizará una detección por formas. En cambio, para hallar

donde está situado el robot, se utilizará dos colores, así se podrá obtener posición y dirección en el espacio de trabajo.

2.5 Planificación de Ruta

Ahora dirigimos nuestra atención al nivel cognitivo del sistema. La cognición generalmente representa la toma de decisiones para lograr sus objetivos de mayor orden. La navegación abarca la capacidad del robot para actuar en función de sus conocimientos para alcanzar la posición del objetivo de la manera más eficiente y fiable posible. Dado un mapa y una ubicación de objetivo, la planificación de ruta implica identificar una trayectoria que hará que el androide llegue a la ubicación de la meta cuando se ejecute. La planificación de rutas es una competencia estratégica para la resolución de problemas, ya que el robot debe decidir qué hacer a largo plazo para lograr sus objetivos. (Siegwart & Nourbakhsh, 2004)

Antes de la llegada de los robots móviles asequibles, el campo de la planificación de caminos fue muy estudiado debido a sus aplicaciones en el área de la robótica manipuladora industrial. Curiosamente, el problema de planificación de ruta para un manipulador con, por ejemplo, seis grados de libertad es mucho más complejo que el de un robot de accionamiento diferencial que funciona en un entorno plano. Por lo tanto, aunque podemos inspirarnos en las técnicas inventadas para la manipulación, los algoritmos de planificación de ruta utilizados tienden a ser aproximaciones más simples debido a los grados de libertad muy reducidos.

Además, los robots industriales a menudo operan a la velocidad más rápida posible debido al impacto económico del alto rendimiento en una línea de fábrica. Por lo tanto, la dinámica y no solo las cinemáticas de sus movimientos son importantes, lo que complica aún más la planificación y ejecución de rutas. Por el contrario, varios autómatas operan a velocidades tan bajas que la

dinámica rara vez se tiene en cuenta durante la planificación del camino, lo que simplifica aún más la creación de un robot móvil del problema.

En la década de los años 90, el estudio relacionado con el problema de generar trayectorias libres de colisiones provocó un enorme interés, debido a la implementación de diferentes mecanismos para la planificación de rutas, podemos decir que se han propuesto dos enfoques diferentes, el "global" y el "local" (Jerome Barraquand, Langlois, & Latombe, 1992) .

El enfoque global consiste en construir primero una representación concisa de la conectividad del conjunto de configuraciones libres de colisión en forma de un "gráfico de conectividad" y luego buscar un camino en este gráfico. Se han ideado diversas técnicas, por ejemplo, descomposición celular exacta (Schwartz & Sharir, 1983), descomposición celular aproximada (Brooks & Lozano-Pérez, 1985), retracción en una red de curvas unidimensionales (ó' Dúnlaing, Sharir, & Yap, 1983). El enfoque local consiste en buscar una cuadrícula colocada en el espacio de configuración del robot (Donald, 1987) . La heurística calculada a partir de información parcial sobre la geometría del espacio de configuración se usa para guiar la búsqueda de la cuadrícula. La mayoría de las heurísticas propuestas toman la forma de un campo potencial que guía la búsqueda a lo largo del flujo de su vector de gradiente negado (Gouzènes, 1984; Khatib, 1986; Koditschek, 1989).

El principal inconveniente de la planificación de ruta es su complejidad computacional ya que limita su uso a espacios de configuración de baja dimensión. Esta limitación y la fuerte motivación para manejar problemas de planificación práctica ha incitado el desarrollo y éxito de muchos métodos de planificación de trayectoria que usan aleatorización. La clave es desarrollar métodos aleatorios que converjan rápidamente en la práctica, pero que sean lo suficientemente simples como para producir un comportamiento y análisis consistentes. (Kuffner & LaValle, 2000)

Los algoritmos de planificación de rutas aleatorizadas o probabilísticos generalmente se diseñaron para uno de dos contextos: planificación de consulta única y planificación de consultas múltiples (Kavraki, Svestka, Latombe, & Overmars, 1996). Para la planificación de consulta única, se supone que un problema de planificación de ruta única se debe resolver rápidamente, sin ningún pre procesamiento. Uno de los primeros y más populares métodos para resolver este problema fue el enfoque de campo potencial aleatorizado (Jérôme Barraquand & Latombe, 1993). Para la planificación de consultas múltiples, se supone que muchos problemas de planificación de rutas se resolverán para el mismo entorno. En este caso, vale la pena pre procesar la información y almacenarla en una estructura de datos que permita consultas rápidas de planificación de ruta. El enfoque de hoja de ruta probabilística fue el primero en abordar este problema (Kavraki et al., 1996)

En conclusión, el problema básico de la planificación de rutas se basa en hallar caminos sin colisiones para un objeto en movimiento, un robot, entre obstáculos estacionarios y completamente conocidos. Idealmente, el tiempo requerido para la planificación debe estar relacionado con la dificultad de la tarea de planificación, es decir, una ruta simple en un entorno despejado debe encontrarse rápidamente, mientras que una ruta más complicada puede requerir más tiempo. De manera similar, el tiempo de planificación debe relacionarse con la calidad deseada de la ruta devuelta. (Bohlin & Kavraki, 2000)

En la actualidad la planificación de rutas es cada vez más importante tanta para la fabricación automatizada como para los robots móviles, pero también ha encontrado aplicaciones en animaciones por computadora, cirugía médica y biología molecular. (Bohlin & Kavraki, 2000)

2.5.1 RRT

Los planificadores de ruta de árbol aleatorio de exploración rápida (RRT del inglés *rapidly exploring random tree*) (W. G. Aguilar, Abad, Ruiz, Aguilar, & Aguilar-Castillo, 2017; W. G. Aguilar, Morales, Ruiz, & Abad, 2017a, 2017b; W. Aguilar & Morales, 2016) han sido demostrados que son adecuados para resolver diversos problemas de planificación de rutas de alta dimensión. Un RRT se expande iterativamente mediante la aplicación de entradas de control que dirigen el sistema hacia puntos seleccionados aleatoriamente, en lugar de requerir convergencia punto a punto, como en el enfoque de hoja de ruta probabilística (LaValle, 1998).

El algoritmo RRT presentado en (LaValle, 1998) se muestra en Algoritmo 1. El árbol crece desde el vecino más cercano en el árbol, xnear, hacia una configuración que se generó al azar, xrand. La longitud del paso descrita en este trabajo debía ser una distancia pequeña.

Algoritmo 1. Método de expansión RRT básico.

Requiere: Árbol τ y método de iteración

```
For i=1 … K do
    X_rand = random configuration
    X_near = nearest neighbor in tree τ to X_rand
    X_new = extend X_near toward X_rand for step length
    If (X_new can connect to X_near along valid edge) then
        τ.AddVertex(X_new). T.AddEdge(X_new,X_near)
    end if
end for
return τ
```

En la Figura 16 se puede observar como árbol del RRT va avanzando mientras se van creado los puntos aleatorios, se expande rápidamente en algunas direcciones para explorar rápidamente las cuatro esquinas del cuadrado.

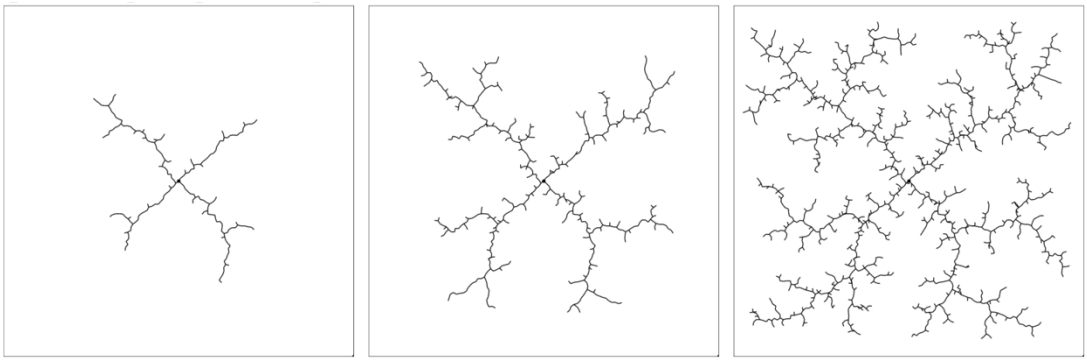


Figura 16. Avance de RRT
Fuente: (LaValle, 1998)

2.5.2 RRT*

Además de conectar el nodo hijo a el padre como lo hace RRT, RRT* también inspecciona cada uno de los nodos que se encuentran dentro de un barrio del recién agregado hijo. Si un nodo puede rastrearse hasta la raíz del árbol a través de un hijo a una distancia más corta que la conexión actual, el padre de este nodo se desplazará como secundario.

El RRT * básicamente trata de suavizar las ramas de los árboles en cada paso, por lo tanto, el zigzag desaparecerá. Matemáticamente se ha demostrado que cuando el número de nodos llega al infinito, la ruta devuelta desde RRT * será la más corta.

En la Figura 17, semillas y nodos para el RRT y RRT* son las mismas; la única diferencia es que RRT* trata de alterar las conexiones de los nodos en la cercanía del nodo hijo. La raíz del árbol está colocada en el origen (0,0) y la meta es explorar la configuración del espacio de 100x100.

En las primeras iteraciones de los algoritmos no se puede notar mucho la diferencia entre los dos, a partir de 500 iteraciones se nota un poco esta diferencia como se puede observar en la Figura 17 dentro de los círculos rojos. Como RRT* intenta suavizar las ramas de los árboles, el árbol RRT* está más organizado que el árbol de RRT.

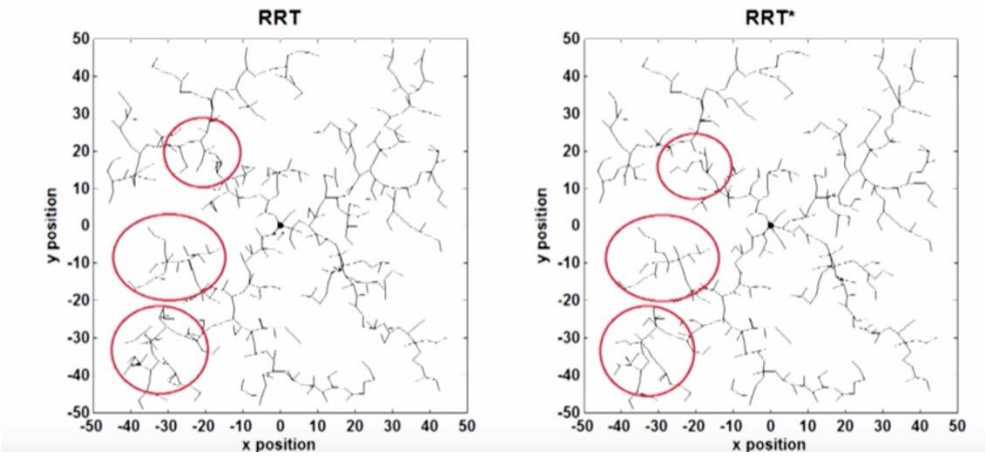


Figura 17. RRT y RRT* con 500 iteraciones
Fuente: (Dong, 2015)

Mientras más iteraciones se haga en los algoritmos más obvia es la diferencia entre los dos como se ve en la Figura 18, el gráfico de la izquierda (RRT) es más caótico que el de la derecha (RRT*). La ruta determinada por RRT* tiene menos zigzag y por ende tendrá una menor distancia hacia la raíz del árbol.

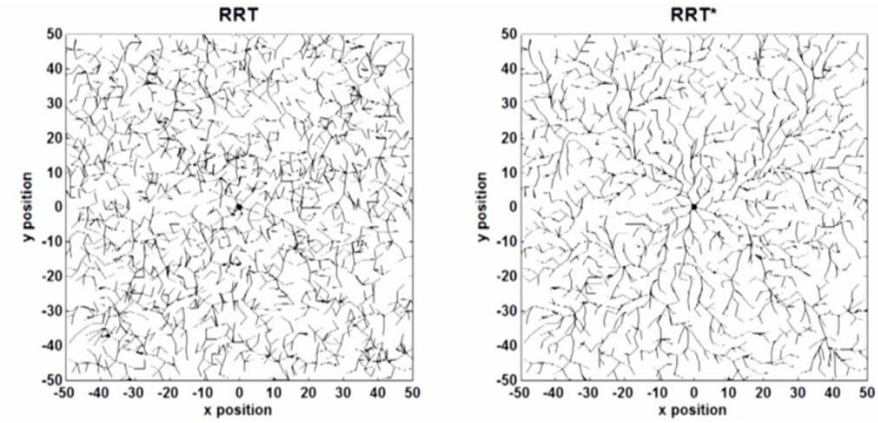


Figura 18. RRT y RRT* con 1000 iteraciones
Fuente: (Dong, 2015)

RRT* como se observó tiene ventajas en comparación al RRT básico, pero requiere un mayor número de recursos computacionales para poder comprobar la cercanía de nodos adyacentes y para realizar su reconexión. Además, para que se vea que RRT* escoja la menor distancia se necesitan un mayor número de iteraciones, por ejemplo, si solo se requieren únicamente 100 iteraciones RRT

será mejor opciones ya que requiere un menor costo computacional y no tendrá mucha diferencia con la ruta encontrada por RRT*

En (Rodríguez, Tang, Lien, & Amato, 2006) se presenta una variante del algoritmo de planificación de ruta de árbol aleatorio de exploración rápida (RRT) que es capaz de explorar pasajes estrechos o áreas difíciles de manera más efectiva. El método incluye muchas formas de hacer crecer el árbol, algunas teniendo en cuenta los obstáculos en el entorno. Este planificador funciona mejor en áreas difíciles cuando se planifica el vuelo libre de robots rígidos o articulados. El algoritmo realizado por (Rodríguez et al., 2006) se muestra en Algoritmo 2, las modificaciones importantes al algoritmo están en la línea 5 donde el nodo objetivo se genera y en las líneas 6-9 donde x_{near} se extiende hacia x_{rand} Rand teniendo el enfoque codicioso descrito.

Algoritmo 2. Método de expansión RRT basado en Obstaculos

Requiere: Árbol τ y método de iteración

1 For i=1 … K do

2 X_{rand} = random configuration

3 X_{near} = nearest neighbor in tree τ to $X_{rand.}$

4 G_i = Select a Growth Method.

5 X'_{rand} = GenerateTargetNode(X_{near}, $X_{rand,}$ G_i)

6 X_{new} = extend X_{near} toward X_{rand} for step length

7 If (X_{new} can connect to X_{near} along valid edge) then

8 τ.AddVertex(X_{new}). T.AddEdge(X_{new},X_{near})

9 end if

10 end for

11 return τ

2.5.3 Mapas probabilísticos PRM

Durante las última dos décadas, los métodos de mapas de ruta probabilística han surgido como un enfoque eficaz para resolver la planificación de movimientos complejos, han demostrado ser capaces de calcular el movimiento de un solo robot y varios sistemas con muchos grados de libertad que operan en entornos geométricos complejos. También pueden tener en cuenta varios tipos de constantes de movimiento, tales como prevención de colisiones, estabilidad, visibilidad y constantes de contacto. Más allá de la robótica, se han utilizado para sintetizar los movimientos de los actores digitales y para predecir los movimientos de macro moléculas biológicas, como las proteínas. (Kavraki, Latombe, & kolountzakis, 1998)

La planificación mediante PRM (RRT del inglés *probabilistic roadmap*) pertenece al enfoque de hoja de ruta la cual no intenta construir una representación exacta de la forma del espacio libre F del robot. De hecho. en la mayoría de los escenarios prácticos, calcular la forma de F es excesivamente costoso. Por lo tanto, la planificación probabilística PRM funciona mediante la construcción de una representación aproximada extremadamente simplificada, basada en la muestra de F, llamada hoja de ruta probabilística. Esta hoja de ruta es un gráfico cuyos nodos, llamados hitos, son configuraciones muestreadas desde el espacio libre de F a una medida de probabilidad adecuada, que refleja la incertidumbre sobre la forma real de F. Un borde, llamado camino local, conecta un par de hitos y un camino de forma simple los une en F. (Cortes, Siméon, & Laumond, 2002)

Las configuraciones de inicio y objetivo del robot, s y g respectivamente, se incluyen entre los hitos en la hoja de ruta. Una vez que se construye la hoja de ruta, se extrae una ruta que conecta s y g usando técnicas de búsqueda de gráficos estándar. Estos pasos de la planificación de PRM también se representan en la Figura 19, detallada de la siguiente forma: (a) La planificación se

realiza en el espacio libre, cuya forma se desconoce. (b) Las configuraciones se muestrean usando alguna medida de probabilidad. © Las configuraciones muestreadas se prueban para colisión y las libres de colisiones (que se encuentran en el espacio libre) se conservan como hitos. (d) Cada hito está vinculado por caminos rectos a sus k vecinos más cercanos. (e) Los enlaces libres de colisiones se retienen para formar el PRM. (f) Finalmente, s y g están conectadas al PRM y el PRM busca una ruta (polilínea roja) de s a g.

La planificación de PRM se basa en la disponibilidad de técnicas eficientes de comprobación de colisiones para comprobar si las configuraciones muestreadas y las rutas locales se encuentran en F. Uno puede pensar en una hoja de ruta probabilística como una red de guardias (los hitos) vigilando F. Estudios teóricos anteriores han demostrado que la razón por la cual la planificación de PRM funciona bien en la práctica es que un espacio libre F típicamente encontrado en un escenario de vida real verifica propiedades de visibilidad, de modo que un número relativamente pequeño de hitos y rutas locales sean suficientes para representar F lo suficientemente bien como para responder correctamente a las consultas de planificación de movimiento con alta probabilidad. (Saha, 2006)

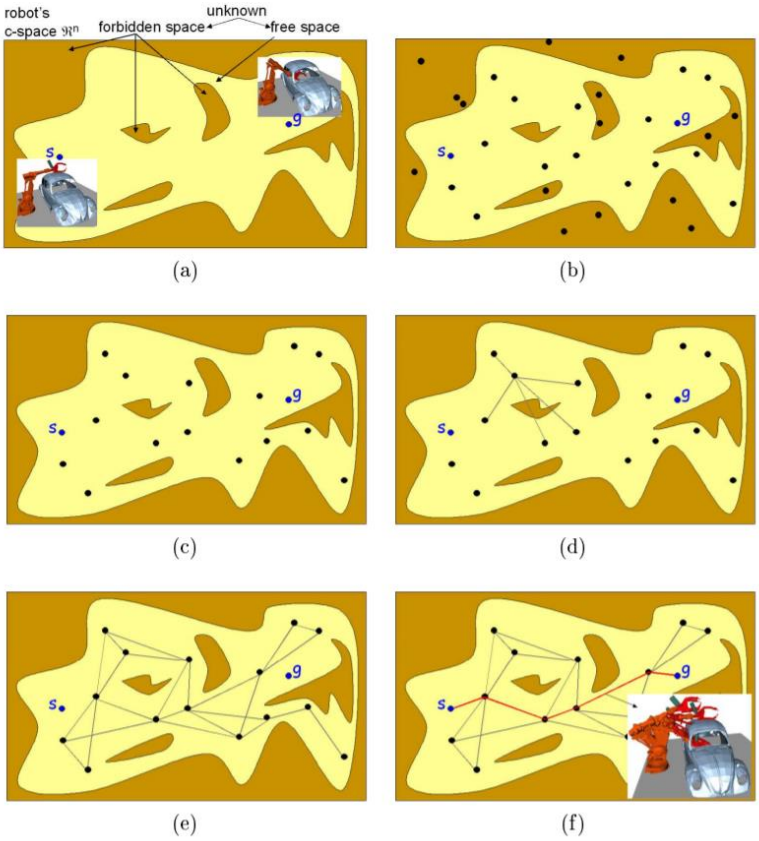


Figura 19. Planificación básica PRM
Fuente: (Saha, 2006)

Dicho método procede en dos fases: una fase de aprendizaje y una fase de consulta. En la fase de aprendizaje, se construye una hoja de ruta probabilística generando de forma repetida configuraciones libres al azar del robot y conectando estas configuraciones usando un planificador de movimiento simple pero muy rápido. Llamamos a este planificador el planificador local. La hoja de ruta así formada en la configuración libre. Después de la fase de aprendizaje, se pueden responder múltiples consultas. Una consulta solicita un camino entre dos configuraciones libres del robot. Para procesar una consulta, el método primero intenta encontrar una ruta desde el inicio y las configuraciones de objetivos a dos nodos de la hoja de ruta. A continuación, se realiza una búsqueda de gráficos para encontrar una secuencia de bordes que conecte estos nodos en la hoja de

ruta. La concatenación de los segmentos de ruta sucesivos transforma la secuencia encontrada en una ruta factible para el robot. (Kavraki et al., 1996)

2.6 Control de Seguimiento de Ruta

Una vez que se ha obtenido la ruta que el robot tiene que seguir por los algoritmos mencionados anteriormente hay que realizar un seguimiento de la misma (W. Aguilar, Casaliglla, & Pólit, 2017; W. G. Aguilar, Angulo, & Costa-Castello, 2017; W. G. Aguilar, Casaliglla, & Polit, 2017; W. G. Aguilar, Casaliglla, Pólit, Abad, & Ruiz, 2017). Una cosa importante que hay que saber es que seguimiento de ruta no es igual que seguimiento de trayectoria, en la primera, como se puede observar en la Figura 20 a) se da con una parametrización libre de tiempo, una velocidad constante hacia adelante y una convergencia "más suave" hacia el camino, mientras que la segunda es una trayectoria con referencia de tiempo y espacio y el vehículo puede dar marcha atrás en su intento de estar en el punto de referencia dado en un momento prescrito como se ilustra en la Figura 20 b) (EECI Institute, 2011). Básicamente en el seguimiento de ruta, el vehículo debe converger y seguir un camino que se especifica sin una ley temporal (Samson, 1992).

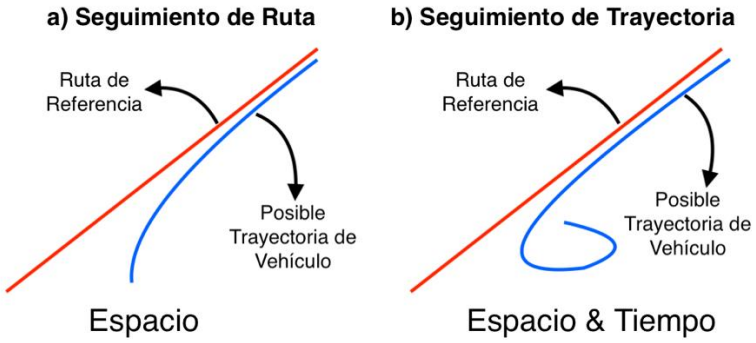


Figura 20. a) seguimiento de ruta
b) seguimiento de trayectoria
Modificado de: (EECI Institute, 2011)

Para los vehículos bajo accionados, es decir, los sistemas con menos actuadores que grados de libertad, seguimiento de la trayectoria sigue siendo un tema de investigación activa. El estudio de

estos sistemas está motivado por el hecho de que generalmente es costoso y a menudo no práctico activar completamente vehículos autónomos debido a consideraciones de peso, fiabilidad, complexidad y eficiencia (Aguiar & Hespanha, 2004).

En el trabajo realizado por (W. G. Aguilar, Angulo, & Costa-Castello, 2017) se observa un seguimiento de trayectoria por un vehículo aéreo en el que luego de obtenida la ruta se utiliza un interpolador de segundo orden que genera puntos intermedios como consignas locales. Se usa para reducir la distancia de error respecto a la trayectoria recta. En la Figura 21 se puede observar a la izquierda que la trayectoria del dron (en azul) sigue perfectamente los puntos de la ruta, pero no sigue la línea que conecta a estos puntos (en rojo). Utilizando el interpolador, a la derecha se puede observar que el vehículo sigue los puntos y la línea que los conecta. Este este trabajo nos recuerda que, si se requiere una mayor precisión durante el seguimiento de la trayectoria, el tiempo de procesamiento aumenta inevitablemente. Es importante buscar un equilibrio entre el tiempo y la precisión. Adicionalmente, el modelo del sistema (W. G. Aguilar, Manosalvas, Guillén, & Collaguazo, 2018; W. G. Aguilar, Salcedo, Sandoval, & Cobeña, 2017; Orbea, Moposita, Aguilar, Paredes, León, et al., 2017; Orbea, Moposita, Aguilar, Paredes, Reyes, et al., 2017), en nuestro caso modelo cinemático, es un requisito indispensable para el diseño del correspondiente controlador.

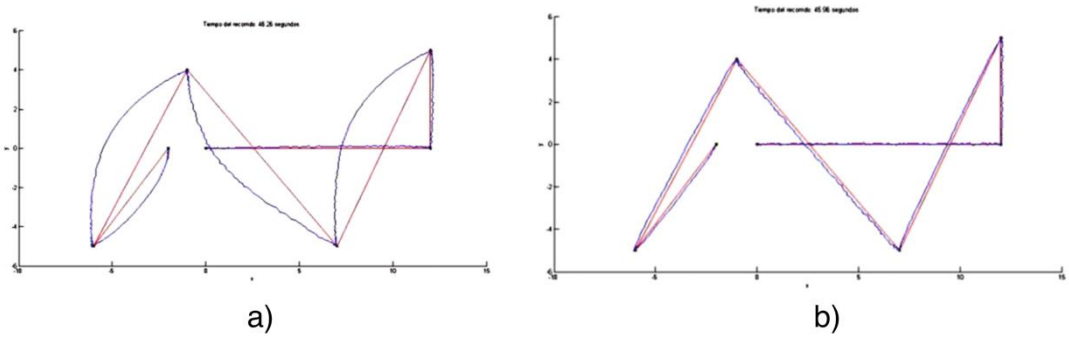


Figura 21. a) Ruta sin Interpolador b) Ruta con Interpolador
Fuente: (W. G. Aguilar, Angulo, & Costa-Castello, 2017)

Buscando en la literatura se han realizado varios estudios en cuanto a control de seguimiento de ruta para vehículos con ruedas (ver, por ejemplo, (Sampei, Tamura, Itoh, & Nakamichi, 1991; Samson, 1995), o los capítulos de libros (de Wit, 1998; de Wit, Siciliano, & Bastin, 1997)), y la convergencia suele estar probada para trayectorias de curvatura constante (Altafini, 2002), pero no se ha tratado mucho este tema en cuando a vehículos con patas, el cuál es el problema en este trabajo.

(Aguiar & Hespanha, 2004) resuelven el problema de ruta descomponiéndolo en dos subproblemas: i) una tarea geométrica, que consiste en converger el vehículo y permanecer dentro de un tubo centrado alrededor de la ruta deseada, y ii) una tarea de asignación dinámica, que asigna un perfil de velocidad en el camino. La hipótesis en el seguimiento de ruta es que la velocidad de avance del vehículo persigue un perfil de velocidad deseado, mientras que el controlador ejerce en la orientación del vehículo para llevarlo a la ruta definida inicialmente. Este es el enfoque que se va tomar como idea y a considerar para realizar el seguimiento de ruta en este trabajo.

CAPITULO III

IMPLEMENTACIÓN DEL ROBOT CUADRÚPEDO

En el desarrollo de este capítulo se presenta los procedimientos utilizados para el diseño de hardware, diseño de software, construcción y control del robot cuadrúpedo, SpiderBot, para posteriormente en los siguientes capítulos, establecer ambientes controlados en los cuales se va a desplazar desde un punto de partida a un punto objetivo, identificando y esquivando la presencia de obstáculos en los diferentes escenarios.

La impresión en 3D permite obtener la estructura del robot articulado, el cual se la adquirió de la literatura estudiada previamente, una vez hallado el diseño de las piezas en 3D se procede a transformarlo en objetos tangibles, dicho proceso de convertir una idea plasmada en un plano a la realidad requiere de algunos elementos y maquinaria específica. El material utilizado para imprimirlo es PLA (ácido poliláctico) un material termoplástico hecho a base de almidón de maíz o caña de azúcar, provenientes de fuentes renovables, siendo así un material biodegradable y amigable con el medio ambiente.

Con las piezas obtenidas de la impresión 3D, se implementó un Robot articulado equipado con cuatro extremidades y cada extremidad con tres eslabones y tres articulaciones, para la implementación de las articulaciones se utiliza 12 micro servos, debido a su alta precisión de posicionamiento, altas velocidades de respuesta gran calidad y diminutas dimensiones, para poder determinar el resto de elementos necesarios para la ejecución del proyecto, como el tipo de microcontrolador, el protocolo de comunicación y la batería ideal, se propone tres alternativas de solución a los requerimientos presentados a en la siguiente sección.

La toma de decisiones del sistema se lo realiza en base al entorno captado mediante una cámara web ubicada en posición cenital y conectada al puerto USB de nuestro ordenador, permitiendo

visualizar y utilizar estas imágenes para realizar el algoritmo de planificación de ruta, el cual una vez hallado el trayecto más óptimo, envía al microcontrolador del Robot la información suficiente para que SpiderBot realice el movimiento necesario para alcanzar el objetivo.

3.1 Diseño Electrónico

3.1.1 Requerimientos de Hardware

A continuación, se mencionarán los requerimientos principales del sistema dado el uso que se le dará al robot cuadrúpedo (SpiderBot) presentado. Sin embargo, este sistema también es capaz de cubrir otros escenarios.

En cuanto a los requisitos de hardware del robot debe contener, al menos, una tarjeta inalámbrica para realizar la comunicación con el computador, además, es necesario un microcontrolador para el movimiento de cada pata, por otra parte, los servomotores elegidos deben ser lo suficientemente pequeños y con la fuerza adecuada para levantar a SpiderBot. Para la alimentación de este se deben utilizar baterías que no ocupen mucho espacio ni sean muy pesadas para libertad de movilidad y que logren entregar la suficiente corriente para los 12 servos, y debido a que el voltaje dado por las baterías tendrá cierta variabilidad mientras se las vaya utilizando es necesario un regulador de voltaje para entregar la alimentación al microcontrolador, servomotores y otros componentes electrónicos.

Para que el robot no utilice la energía de las baterías cuando no se lo esté utilizando se requieren 3 interruptores, uno para la parte de potencia, uno para la de control y el ultimo para encender o apagar todo el sistema.

Por último, una placa de circuito impreso (PCB) será necesaria para realizar la conexión de todos los elementos, en la Tabla 1se muestran los requerimientos de hardware del robot.

Tabla 1

Requerimientos de Hardware de SpiderBot

Nº	Elemento	Cantidad
1	Tarjeta Comunicación inalámbrica	1
2	Microcontrolador	1
3	Servomotores	12
4	Batería	1
5	Regulador de Voltaje	1
6	PCB	1
7	Interruptores	3

3.1.2 Alternativas de Solución

Para poder elegir el hardware que se acople mejor a las necesidades del robot encontradas en la sección anterior, se plantearán tres diferentes soluciones, las cuales se elegirán mediante la metodología de AHP (Analytic hierarchy process) por sus siglas en inglés. En cada una de las alternativas se modificará el microcontrolador, el protocolo y dispositivo de comunicación, los servomotores y batería.

3.1.2.1 Alternativa de Solución 1

3.1.2.1.1 Microcontrolador

Arduino Nano es una tarjeta de desarrollo open-source construida con un microcontrolador modelo ATmega328 que posee pines de entradas y salidas (E/S). En la Tabla 2 se muestran las especificaciones técnicas de este microcontrolador.

Tabla 2

Especificaciones Técnicas Arduino Nano

Característica	Descripción
Microcontrolador	ATmega328
Arquitectura	AVR
Voltaje de Operacion	5 V
Flash Memory	32 KB

CONTINÚA ➡

SRAM	2 KB
Velocidad de Reloj	16 MHz
Pines de Entrada Analógicos	8
EEPROM	1 KB
Corriente DC por Pin I/O	40 mA
Voltaje de Entrada	7-12 V
Pines Digitales I/O	22
Salidas PWM	6
Consumo de Energía	19 mA
Tamaño PCB	18 x 45 x 5 mm = 4.05 cc
Costo	$8

3.1.2.1.2 Protocolo y Dispositivo de Comunicación

En esta solución, se utilizará el módulo de comunicación Wi-Fi ESP8266EX ESP-01, el mismo que se lo puede utilizar fácilmente con la tarjeta Arduino. En la Tabla 3 se encuentran las especificaciones técnicas de esta tarjeta.

Tabla 3

Especificaciones Técnicas ESP8266EX ESP-01

Característica	**Descripción**
Microcontrolador	Tensilica L106 32-bit
Conexión de Perifericos	UART/SDIO/SPI/I2C/I2S/IR
Voltaje de Operacion	3.0 V ~ 3.6 V
Corriente de Operacion	80 mA
Antena	PCB Trace, External, IPEX Connector
Protocolos	802.11 b/g/n/e/i
Estándares	FCC/CE/TELEC/SRRC
Rángo de Frecuencia	2.4 G 2.5 G
Consumo Eléctrico	135 mA (modo 802.11n)
Tamaño PCB	14.3 x 24.8 x 2 mm = 0.709cc
Costo	$7

3.1.2.1.3 Servomotores

Para esta Solución utilizaremos los servomotores MG-90, los cuales son diminutos y ligeros con alta potencia de salida, servos tienen engranajes internos de aluminio. Las características de esos servos se muestran en la Tabla 4.

Tabla 4.
Características Servos MG-90

Característica	Descripción
Velocidad	0.10 sec/60° @ 4.8V
Torque	2 Kg-cm @ 4.8V
Voltaje de Operación	3.2-7.2V
Ancho de Pulso	500-2400 µs
Ángulo de rotación	180°
Consumo Eléctrico	750 mA
Tamaño	22.8 x 12.2 x 28.5mm = 7.93cc
Costo	$7

3.1.2.1.4 Batería

La batería elegida en esta solución son Baterías de 9V recargables debido a su bajo costo y alto voltaje que proporcionan, una de las desventajas de esta opción de batería es que no entregan una corriente muy elevada, otras características de esta batería las podemos encontrar en la Tabla 5.

Tabla 5
Características Baterías Recargables 9V

Característica	Descripción
Voltaje Nominal	8.4 Volts
Temperatura de Operación	-18°C to 55°C
Peso	45.0 grams
Capacidad	175 mAh* at 21°C (70°F)
Volumen	22.0 cc
Costo	$8

3.1.2.2 Alternativa de Solución 2

3.1.2.2.1 Microcontrolador

En esta opción de solución se eligió el ATmega328p, es un microcontrolador de bajo consumo CMOS de 8 bits basado en la arquitectura RISC mejorada AVR, este tiene un empaquetado SMD, con esto ocupara un espacio menor en la PCB. En la Tabla 6 se muestran las especificaciones técnicas del mismo.

Tabla 6
Especificaciones Técnicas ATmega328p

Característica	Descripción
Microcontrolador	ATmega328
Voltaje de Operación	1.8 - 5.5V
Flash Memory	32 KB
SRAM	2 KB
Velocidad de Reloj	16 MHz
Pines de Entrada Analógicos	8
EEPROM	1 KB
Pines Digitales I/O	22
Salidas PWM	6
Consumo de Energía	0.2mA
Tamaño	8.75 x 8.75 x 1.20 mm=0.092cc
Costo	$4.5

3.1.2.2.2 Protocolo y Dispositivo de Comunicación

El Módulo HC-05 es un dispositivo Bluetooth que facilita la realización de una conexión serial inalámbrica. A continuación (Tabla 7) se mencionan las características que posee este modulo.

Tabla 7
Especificaciones Técnicas Módulo HC-05

Característica	Descripción
Voltaje de Operación	3.6 ~ 6 V
Corriente de Operación	30 mA
Antena	PCB Trace

CONTINÚA ➡

Protocolo	Bluetooth 802.15.1
Estándares	Serial communication (USART)
Baud Rate	9600,19200,38400,57600,115200,230400
Consumo Eléctrico	30mA
Tamaño PCB	16.1 x 37.5x3 mm = 1.811cc
Costo	$6

3.1.2.2.3 Servomotores

Para esta Solución utilizaremos los servos SG-90. Estos servomotores tienen características parecidas a los de la Solución 1, los MG-90, la diferencia más grande entre estos dos modelos es que los SG-90 tienen engranajes de plástico. Las características de esos servos se muestran en la *Tabla 8*.

Tabla 8
Características Servos SG-90

Característica	**Descripción**
Velocidad	0.10 sec/60° @ 4.8V
Torque	1.8 Kg-cm @ 4.8V
Voltaje de Operación	3.0-7.2V
Ancho de Pulso	500-2400 µs
Ángulo de rotación	180°
Consumo Eléctrico	550 mA
Tamaño	22.2 x 11.8 x 31 mm = 8.12cc
Costo	$5.5

3.1.2.2.4 Batería

Las baterías en esta solución son Baterías Lipo recargables, la mayor ventaja de esta opción de batería es que entregan una corriente alta, necesaria para que los servos funcionen correctamente, su desventaja es que sólo entregan 3.7V, voltaje insuficiente para hacer funcionar el sistema, por esto se tendrán que usar 2 baterías en serie para dar un total de 7.4V, otras características de esta batería las podemos encontrar en la Tabla 9.

Tabla 9

Características Baterías Recargables 9V

Característica	Descripción
Voltaje Nominal	3.7 Volts
Temperatura de Operación	-18°C to 55°C
Peso	23.0 grams
Capacidad	500 mAh
Volumen	8.5 x 25 x 40mm = 8.5cc
Costo	$12

3.1.2.3 Alternativa de solución 3

3.1.2.3.1 Microcontrolador

La Raspberry Pi Zero W es una computadora increíblemente barata, tiene unas características muy parecidas a sus hermanos mayores como son la Raspberry Pi 2 o Raspberry Pi 3, su mayor atractivo es su tamaño reducido. En la Tabla 10 se muestran las especificaciones técnicas de esta tarjeta.

Tabla 10

Técnicas Raspberry Pi Zero W

Característica	Descripción
CPU	ARM11 Broadcom CPU
Voltaje de Operación	5V
Flash Memory	microSD card slot
RAM	512MB
Velocidad de Reloj	1 GHz
Voltaje de Entrada	5V
Pines Digitales I/O	26
Consumo de Energía	200mA
Tamaño PCB	65 x 30 x 0.2 mm = 0.39cc
Costo	$39

3.1.2.3.2 Protocolo y Dispositivo de Comunicación

En esta solución, se utilizará el módulo de comunicación integrado directamente en la Raspberry Pi Zero W, de aquí viene la letra W, de Wireless, con esta opción no se tiene que incluir otro módulo adicional, introduciendo más espacio y gasto. En la **Tabla 11** se encuentran las especificaciones técnicas los módulos de comunicación integrados en la Raspberry Pi Zero W.

Tabla 11
Especificaciones Comunicación Raspberry Zero W

Característica	Descripción
Microcontrolador	Incluido en Microcontrolador
Voltaje de Operación	
Corriente de Operación	
Consumo Eléctrico	
Tamaño PCB	
Costo	
Antena	Impresa en la Tarjeta
Protocolos	802.11 b/g/n, Bluetooth 4.1/BLE
Frecuencia	2.4GHz en Wi-Fi

3.1.2.3.3 Servomotores

Para esta Solución utilizaremos los servomotores SG-50, los cuales tienen unas dimensiones menores a los SG-90 y MG-90 y, estos servos tienen engranajes internos de plástico. Las características de esos servos se muestran en la Tabla 12.

Tabla 12
Características Servos SG-50

Característica	Descripción
Velocidad	0.10 sec/60° @ 4.8V 0.08 sec/60° @ 6.0V
Torque	2 Kg-cm @ 4.8V
Voltaje de Operación	4.8-6 V
Ancho de Pulso	500-2400 µs

CONTINÚA ➡

Ángulo de rotación	**180°**
Consumo Eléctrico	500 mA
Tamaño	21.5 x 11.7 x 25.1mm = 6.31cc
Costo	$4.5

3.1.2.3.4 Batería

Las baterías en esta solución son Baterías Lipo recargables, al igual que en la Solución 2, pero esta a diferencia de las anteriores tienen un número mayor de celdas, entregando así un mayor voltaje, su desventaja es que para recargarlas se necesita de un cargador especial para cargas balanceadas. Las características de esta batería se las puede encontrar en la Tabla 13. Este tipo de batería tiene la mayor capacidad que las opciones anteriores, pero tiene un costo elevado.

Tabla 13
Características Baterías Recargables 9V

Característica	**Descripción**
Voltaje Nominal	14.4 V
Peso	172.6g
Capacidad	1300mAh
Volumen	69 x 32 x 29 mm = 64.03mm
Costo Batería	$25

3.1.3 Selección de Solución en Base a AHP

En esta sección se pretende indicar cuál de las alternativas presentadas será utilizada para continuar con el desarrollo del proyecto, esto se lo realizará mediante AHP. AHP es una herramienta práctica para tratar la toma de decisiones complejas, estableciendo prioridades y así poder tomar la mejor decisión con un sustento matemático a partir de un número determinado de alternativas de solución.

Se desea equipar al robot cuadrúpedo entre varias alternativas, vistas en las anteriores secciones (Solucion1, Solucion2, Solucion3) y seleccionar la mejor opción basado en criterios (Costo, Consumo, Tamaño). Para tomar la mejor decisión de equipamiento de hardware, se usará AHP.

3.1.3.1 Objetivo

Equipar al robot cuadrúpedo del hardware necesario para cumplir los requerimientos.

3.1.3.2 Definir los Criterios de Selección

- Costo
- Consumo
- Tamaño

3.1.3.3 Seleccionar las Alternativas

- Alternativa de Solución 1.
- Alternativa de Solución 2.
- Alternativa de Solución 3.

3.1.3.4 Árbol de Jerarquía

Toda la información vista anteriormente esta entonces dispuesta en un Árbol Jerárquico mostrado en la Figura 22. La información está sintetizada para determinar rankings relativos de alternativas. Criterios cuantitativos (Costo, Consumo Eléctrico, Tamaño) serán comparados usando criterios encontrados en las necesidades de los requerimientos finales del proyecto para derivar pesos y prioridades

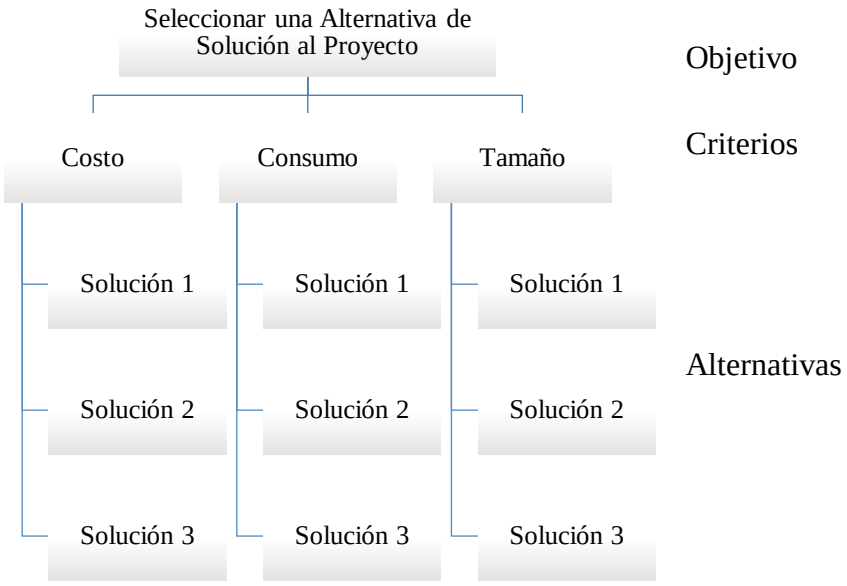


Figura 22. Árbol Jerárquico Ejemplo Ilustrativo

3.1.3.5 Comparaciones Pareadas

Se realiza el uso de juicios para determinar el ranking de los criterios con la siguiente escala:

- 1- igual
- 3 - moderado
- 5 - fuerte
- 7 - muy fuerte
- 9 – extremo

Utilizando comparaciones pareadas, se puede expresar la importancia relativa de un criterio sobre otro:

1. El Consumo Eléctrico (CE) es 2 veces importante que el Costo (C).
2. El Tamaño (T) es 3 veces importante que el Costo (C).
3. El Consumo Eléctrico (CE) es tan importante como el Tamaño (T).

En la Tabla 14 se puede encontrar resumida numéricamente toda esta información.

Tabla 14

Comparaciones Pareadas: Costo, Tamaño, Consumo Eléctrico

	Costo	**Consumo**	**Tamaño**
Costo	1/1	1/2	1/3
Consumo	2/1	1/1	1/1
Tamaño	3/1	1/1	1/1

3.1.3.6 Resolución del Vector Propio

A continuación, se muestra la resolución del Vector Propio mediante una función realizada en Matlab que incluye Elementos del Proceso Jerárquico Analítico (AHP) para la toma de decisiones, este código fue realizado en inglés y fue traducido por el grupo y se muestra en el ANEXO 1.

Ingresando estos datos al programa de Matlab, se obtiene el resultado de la Figura 23.

```
Criterios de comparacion de pares Matriz PCM

PCM =

    1.0000    0.5000    0.3333
    2.0000    1.0000    1.0000
    3.0000    1.0000    1.0000

Vector propio de matriz:

ePCM =

    0.1692
    0.3874
    0.4434
```

Figura 23. Vector Propio obtenido en Matlab para Criterios de Selección de Hardware

Por lo Tanto, en la Tabla 15 con los pesos obtenidos se puede observar qué criterios es el más importante para el proyecto.

Tabla 15

Importancia de Criterios de Selección de Hardware

Tamaño	0.4434	El criterio más importante
Consumo Eléctrico	0.3874	El segundo criterio más importante
Costo	0.1692	El criterio menos importante

3.1.3.7 Pesos a las Alternativas (Soluciones)

3.1.3.7.1 En términos de Consumo eléctrico

Debido a que el Consumo eléctrico es una cantidad cualitativa no es necesario realizar la comparación a pares y sólo se necesita determinar el Vector normalizado de Consumo eléctrico. Se utilizarán los datos de consumo totales de cada solución, pero hay algo que se tiene que tomar en cuenta en el consumo eléctrico y esto es que, a mayor consumo menor beneficio, es por esto que se hará un complemento de cada una de las cantidades obtenidas, este cálculo se encuentra en la Tabla 16.

Tabla 16

Comparación del Consumo Eléctrico: Alternativas Cuantitativas

	Consumo Eléctrico (W)	**Complemento**	**Vector Normalizado**
Solución 1	19mA(5V)+135mA(5V)+12*750mA(5V) =45.77W	109.92-45.77 =64.15	0.2918
Solución 2	0.2mA(5V)+30mA(5V)+12*550mA(5V) =33.15W	109.92-33.15 =76.77	0.3492
Solución 3	200mA(5V)+12*500mA(5V) =31W	109.92-31 =78.92	0.3590
	Total = 109.92W	Total = 219.84	1

El resultado en Matlab se lo puede ver en la Figura 24.

```
Comparacion del Consumo Electrico: Alternativas Cuantitativas

ACM_CE =

   64.1500
   76.7700
   78.9200

Matriz Normalizada

eACM_CE =

    0.2918
    0.3492
    0.3590
```

Figura 24. Vector Propio obtenido en Matlab para Comparación del Consumo Eléctrico

3.1.3.7.2 En términos de Costo

En términos de Costo, aquí, al igual que en el Consumo Eléctrico a mayor costo, menor beneficio, entonces también se hará un complemento de cada una de las cantidades obtenidas (Tabla 17).

Tabla 17
Comparación del Costo: Alternativas Cuantitativas

	Costo ($)	**Complemento**	**Vector Normalizado**
Solución 1	8+7+7(12)+8=107	313.5-107=206.5	0.3293
Solución 2	4.5+6+5.5(12)+12=88.5	313.5- 88.5=225	0.3589
Solución 3	39+4.5(12)+25=118	313.5- 118=195.5	0.3118
	Total = 313.5	Total = 627	1

El resultado en Matlab se obtiene en la Figura 25.

```
Comparacion del Costo: Alternativas Cuantitativas

ACM_C =

  206.5000
  225.0000
  195.5000

Matriz Normalizada

eACM_C =

    0.3293
    0.3589
    0.3118
```

Figura 25. Vector Propio obtenido en Matlab para Comparación del Costo

3.1.3.7.3 En términos de Tamaño

En términos del Tamaño de los componentes. A mayor tamaño, existe un menor beneficio para SpiderBot, ya que se requiere que este sea pequeño y compacto, entonces también se hará un complemento de cada una de las cantidades obtenidas, este cálculo se encuentra en la Tabla 18.

Tabla 18

Comparación del Tamaño: Alternativas Cuantitativas

	Tamaño (cc)	**Complemento**	**Vector Normalizado**
Solución 1	4.05+0.709+12(7.93)+22=121.919	369.902-121.919=247.983	0.3352
Solución 2	0.092+1.811+12(8.12)+8.5=107.843	369.902-107.843=262.059	0.3542
Solución 3	0.39+12(6.31)+64.03 =140.14	369.902-140.14=229.762	0.3106
	Total = 369.902	Total = 739.804	1

Comprobando en Matlab se obtuvo el mismo resultado, y se lo puede ver la Figura 26.

```
Comparacion del Tamano: Alternativas Cuantitativas

ACM_T =

  247.9830
  262.0590
  229.7620

Matriz Normalizada

eACM_T =

    0.3352
    0.3542
    0.3106
```

Figura 26. Vector Propio obtenido en Matlab para Comparación del Tamaño

Volviendo a la Jerarquía e incluyendo los pesos se obtiene la Figura 27.

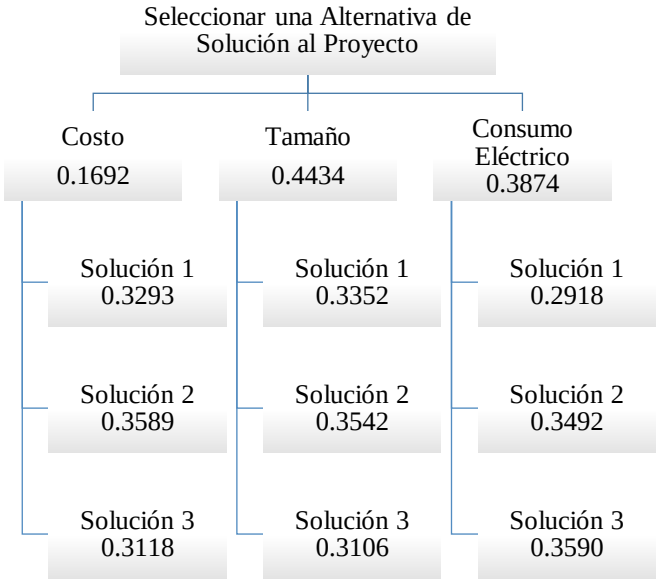


Figura 27. Jerarquía del Elección de alternativa del Proyecto con pesos

3.1.3.8 Selección de Solución

En el Programa de Matlab (ANEXO 1), se obtiene el vector final con los pesos de beneficios de cada alternativa de solución (Figura 28).

```
Puntajes para: solucion1, solucion2, solucion3

Puntajes_Soluciones =

    0.3150
    0.3528
    0.3322

Solucion ganadora basada en beneficios:

Maximo_beneficios =

    0.3528
```

Figura 28. Resultado de AHP

Por lo tanto, la solución 2 es la alternativa con más alto valor en el resultado, por ende, esta es la solución ganadora.

3.1.4 Diseño de la Placa PCB

En el diseño, es indispensable establecer el esquemático del circuito y el diagrama de conexiones entre los componentes de la placa, para esto se dividió en dos fases, primero partimos de las especificaciones y funcionalidad del circuito a diseñar, decidiendo que elementos se va a utilizar y la interconexión entre ellos, luego con dicha información se define la máscara en el editor de placas de circuito impreso el cual permite una representación virtual de la ubicación de los componentes y su conexión física en la placa.

Para realizar el circuito electrónico se necesitó de documentación, hojas técnicas y principalmente el apoyo de una herramienta computacional. La aplicación que se utiliza para el diseño de la placa PCB es el paquete informático Proteus 8 Professional, el cual es un programa de diseño y simulación electrónica, desarrollado por Labcenter Electronics. Se usa dicha herramienta ya que está compuesta por dos módulos necesarios para la fabricación de circuitos impresos: el primer módulo ISIS, permite el diseño del esquema electrónico y a partir de esto el segundo módulo ARES, ejecuta el diseño de la placa de circuito impreso final.

El proceso para la fabricación se define con las etapas mostradas en la Figura 29.

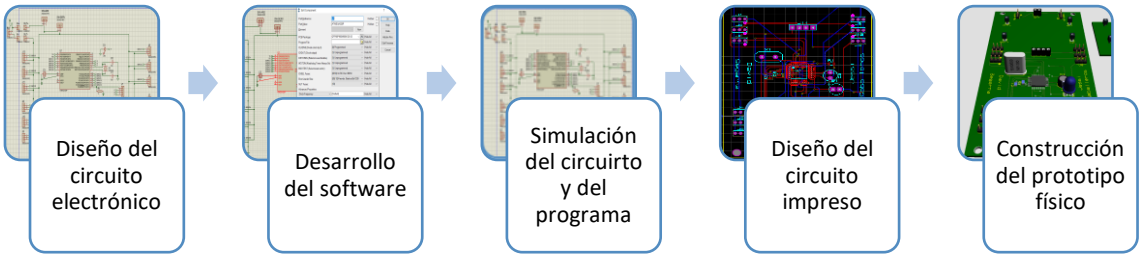


Figura 29. Etapas para la elaboración PCB utilizando Proteus

En el diseño del diagrama electrónico se usa la herramienta ISIS de Proteus, la cual se la compara con un lienzo colocando en él la representación de los distintos componentes e interconexiones de los elementos, permitiendo su modelización para la simulación.

Durante esta etapa también se decide el tipo de encapsulado de cada elemento para el desarrollo del PCB se utiliza dos tipos de tecnologías: THD (Through Hole Device) componentes con pines para su instalación y SMD (Surface Mounted Device) componentes que se montan superficialmente, montando componentes de ambos tipos. En la Figura 30 se muestra el diagrama esquemático del circuito electrónico del robot cuadrúpedo SpiderBot

En el diseño del diagrama PCB se usa la herramienta ARES de Proteus, desarrollada únicamente para el diseño de placas de circuito impreso, pero previamente necesita el diagrama electrónico realizado en ISIS para asegurarse que la placa una los pines de forma idéntica al diseño del esquema electrónico.

Debido a la complejidad de las conexiones físicas del circuito y a la necesidad de un espacio reducido, se diseña una placa con pistas conductoras en las dos caras y uniendo mediante Vías las pistas de un lado con el otro, empleando un circuito multicapa, como se puede ver en la Figura 31 se muestra los dos lados de la placa.

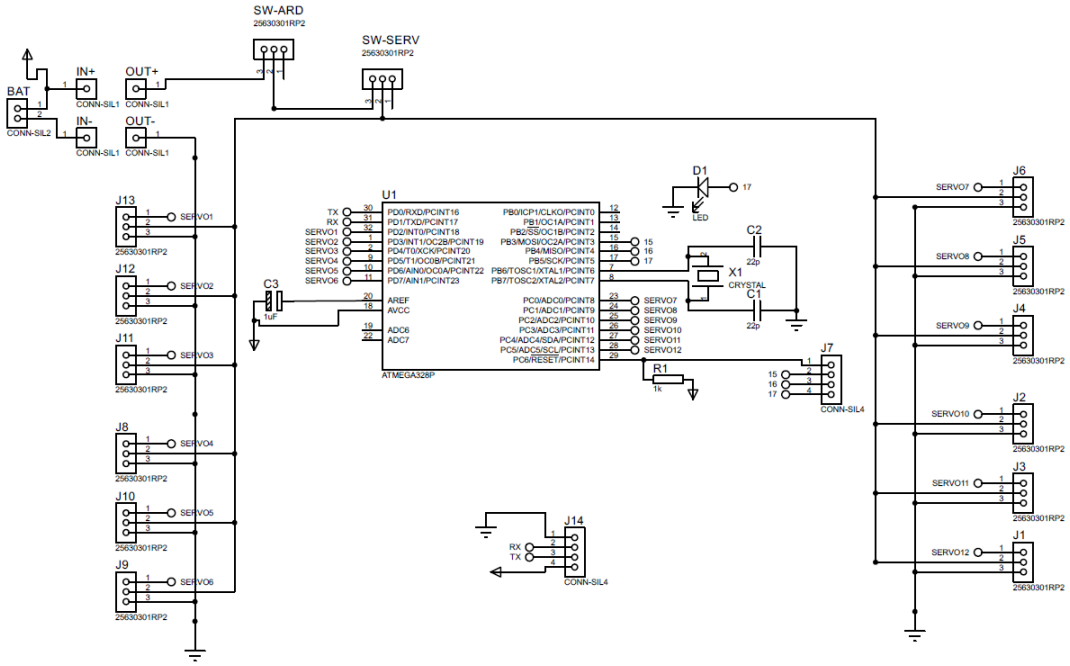


Figura 30. Diagrama del circuito electrónico de SpiderBot diseñado en Proteus

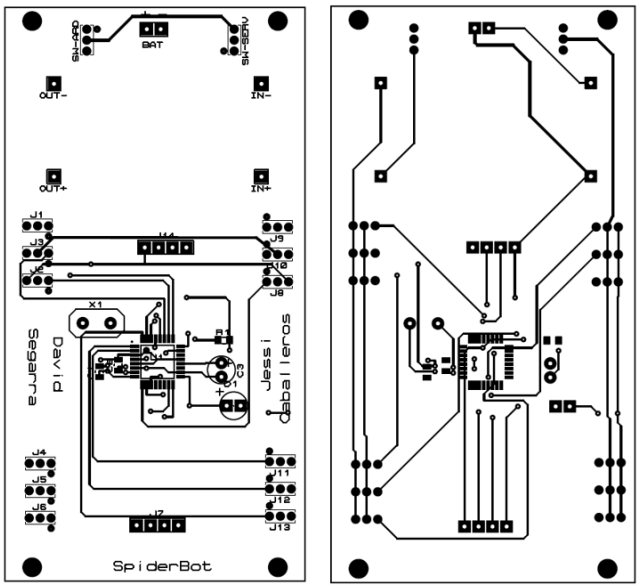


Figura 31. Diagrama PCB multicapa de SpiderBot diseñado en Proteus

Y finalmente se obtiene la vista tridimensional del diseño PCB, como se ve en la Figura 32.

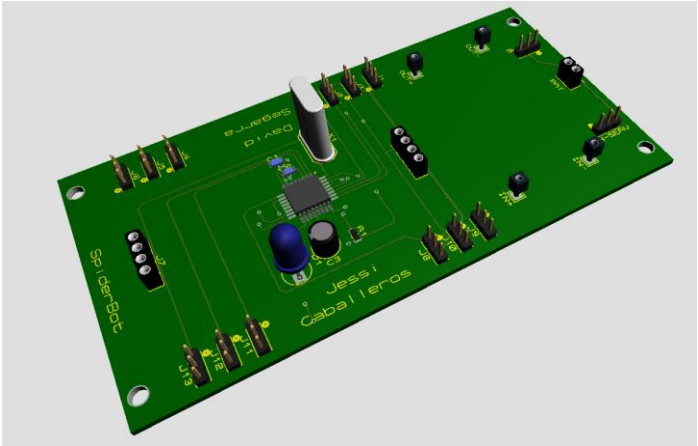


Figura 32. PCB del Robot Cuadrúpedo SpiderBot visualizado en 3D

El proceso de fabricación de la PCB, se lo hace a través del servicio de manufactura de prototipos electrónicos que ofrece una empresa que realiza placas profesionales, para esto fue necesario exportar los archivos en formato Gerber del diseño PCB, y posteriormente obtenemos la placa completamente lista para montar y soldar todos los componentes como se muestra en la Figura 33.

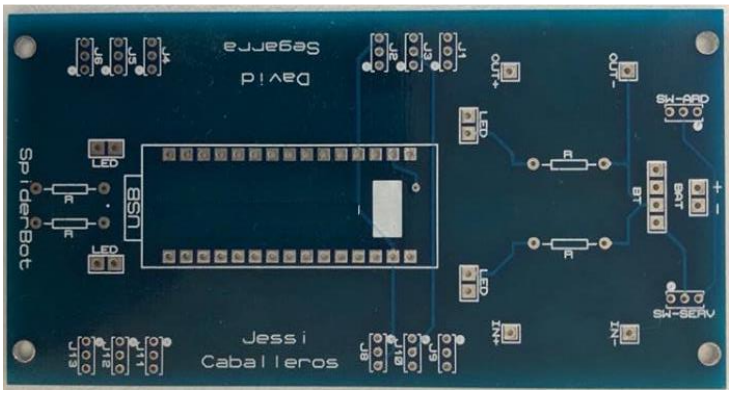


Figura 33. PCB final del Robot Cuadrúpedo SpiderBot

3.2 Arquitectura de Hardware

La unidad de control del robot se basa en la tecnología de ATmega328p. Las señales PWM que comandan cada pierna, son entregadas por este microcontrolador y están conectadas por medio de la placa diseñada en el ítem anterior a los servomotores. Totalmente, se tiene 12 motores que pertenecen a las patas. El diagrama de bloques del sistema simplificado se puede expresar como se muestra en la Figura 34.

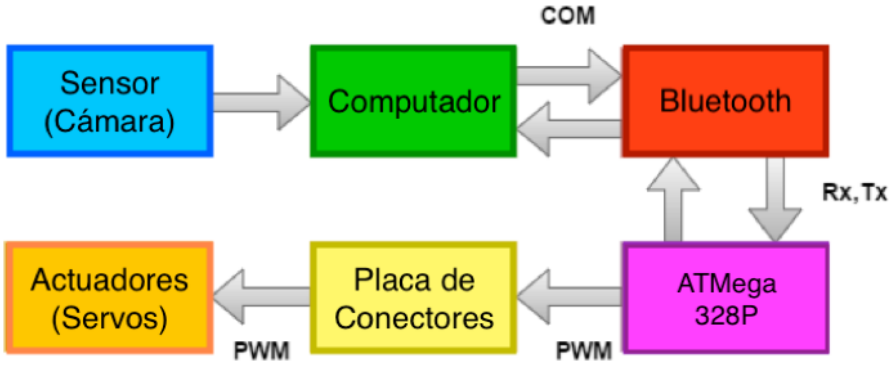


Figura 34. Diagrama de Bloques del Sistema

3.2.1 Cámara

La cámara servirá como sensor principal de todo el sistema, esta permitirá tener una visión del entorno al que el robot se encuentra expuesto.

Esta cámara tiene que estar debidamente calibrada y enfocada, ya que con ella se observarán los obstáculos por medio de formas y a SpiderBot por medio de colores.

Además, para que la cámara tenga una visión total, estará ubicada desde una perspectiva cenital, esto es desde arriba, pero no de cualquier manera, sino en un ángulo perpendicular al plano de movimiento de SpiderBot, es decir, desde arriba hacia el suelo. Una analogía de esta perspectiva es como se ven las calles, avenidas, casas y edificios desde una aplicación de mapas como lo es Google Maps.

La resolución elegida para esta cámara es de 1280×720 pixeles, y cubrirá un área de 2 x 1.6 m, espacio suficiente para realizar las pruebas con el Robot, el cual tiene dimensiones relativamente pequeñas.

3.2.2 Computador

El computador será uno de los dos cerebros que componen el sistema, en este se conectará la cámara mediante USB, recibirá las imágenes que esta proporciona, y, mediante el código realizado en Python, realizará el procesamiento necesario para poder detectar a SpiderBot y a los obstáculos que tiene que evadir. Posteriormente, muestra estas imágenes al usuario en la pantalla, el usuario elige un punto en el espacio donde desea que se mueva la araña y el computador calcula mediante los diferentes algoritmos implementados la ruta que deberá seguir el robot y por último envía por medio una conexión serial Bluetooth comandos a la araña diciéndole a que dirección debe moverse (adelante, izquierda, derecha) para ir desde el punto inicial hacia el punto final evadiendo todos los obstáculos en su camino.

3.2.3 Módulo Bluetooth

El módulo Bluetooth cumplirá las funciones de comunicación entre el computador y el microcontrolador ubicado en SpiderBot. La comunicación será de una forma serial, con una velocidad de 9600 bps. Este módulo recibirá los comandos de dirección (adelante, izquierda,

derecha) desde el computador de una manera inalámbrica y los traducirá a señales eléctricas alámbricas que pasaran de manera serial hacia el microcontrolador ATMega 328P.

3.2.4 ATMega 328P

El ATMega 328P es el microcontrolador que desempeñará todas las tareas de control del robot, una vez que reciba un comando desde el computador, trasladará macro instrucciones como adelante, izquierda, derecha a funciones y operaciones que van a ser entendidas por los actuadores, en otras palabras, si el computador envía la instrucción de que el robot vaya hacia adelante, el microcontrolador dirá que servos deben moverse para girar las articulaciones y así poder mover cierta pata que permita el avance de SpiderBot. Es decir, el microcontrolador contendrá todo el algoritmo de movimiento.

3.2.5 Placa de Conectores (PCB)

La PCB es uno de los elementos más importantes de todo el sistema ya que tiene las funciones de proveer una forma de unir todos los elementos de SpiderBot, acoplando cada pin del microcontrolador a los 12 servos, para transmisión de señales PWM. Además, realiza la conexión entre el módulo Bluetooth y el ATMega 328P. Y por último entrega el voltaje y la corriente necesaria que la batería elegida suministra, siendo así el núcleo o corazón del sistema, logrando que todo funcione.

3.2.6 Servomotores

Los servomotores serán los principales y únicos actuadores del Robot, son doce servos en total, de los cuales tres van ubicados en cada una de las cuatro piernas. Un servo para la articulación entre el tórax y coxa (primer segmento de la pata de un insecto), otro que une la coxa con el fémur y, por último, un servomotor entre la tibia y el fémur.

Con cada uno de los servos se obtendrá una respuesta mecánica controlada mediante señales PWM enviadas desde el microcontrolador ATMega 328P y trabajando en conjunto, darán motricidad y locomoción a SpiderBot, es decir, serán los encargados de transformar la energía eléctrica recibida en energía mecánica.

3.3 Construcción del Robot

Hoy en día existe un sin número de materiales útiles para la fabricación de piezas plásticas, como se mencionó en la sección anterior el material utilizado para la creación de cada parte de SpiderBot es hecha en PLA (ácido poli láctico), un material termoplástico ligero y biodegradable hecho a base de almidón de maíz. La estructura mecánica del Robot Cuadrúpedo está compuesta por el soporte central y cuatro patas simétricamente similares, como se puede ver en la Figura 35.

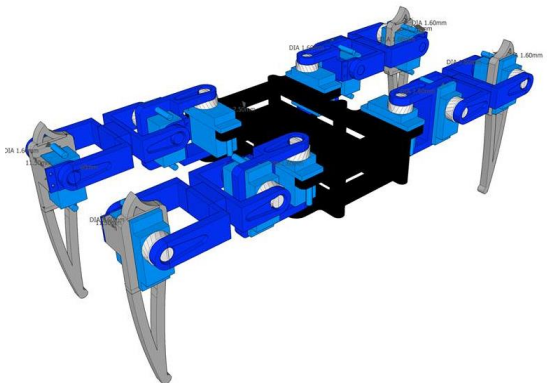


Figura 35. Diseño del Robot Cuadrúpedo

3.3.1 Diseño del soporte central de SpiderBot

Son dos piezas ligeras y fáciles de adaptar, constituyen el cuerpo del Robot, con dimensiones de 16 cm de largo, 10 cm de ancho y 1.6 cm de espesor cada una. Para mantener un equilibrio contrastante al caminar, en estas piezas se fijan las patas en puntos opuestos al cuerpo logrando conservar la gravedad en el punto medio del soporte. La batería, el interruptor de encendido, el módulo Bluetooth y la placa de control, están incrustados en el centro del cuerpo. El diseño del cuerpo se muestra en la Figura 36.

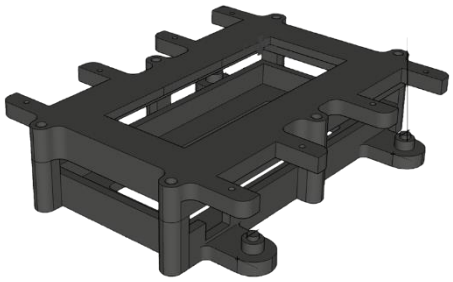


Figura 36. Diseño del Soporte central del Robot Cuadrúpedo

3.3.2 Diseño de las articulaciones de SpiderBot

Cada una de las patas del robot tienen el mismo sistema mecánico y tres grados de libertad (DOF), manteniendo una longitud igual de eslabones. Cada pata está formada por tres eslabones que se asemejan a la pata de un insecto a su distancia y disposición. La igualdad de las cuatro extremidades en SpiderBot Facilitara más adelante el modelado cinemático de las patas en Matlab. En Tabla 19 se indica las medidas de cada eslabón.

Tabla 19

Medida de las partes de una pata del Robot

Eslabón	Largo (mm)	Ancho (mm)	Alto (mm)
Uno	35	44	16
Dos	38	67	12
Tres	100	20	16

El eslabón número uno es en similitud a la coxa de un insecto, y su movimiento lo ejecuta el primer servomotor, el cual está fijo en el cuerpo de robot, en este eslabón se fija el segundo servomotor en el que se monta la siguiente parte de la articulación. El eslabón número dos es en similitud al fémur de un insecto, y es el que fija el servomotor tres para unir con el eslabón número tres el cual es en similitud a la tibia de un insecto, el movimiento de esta articulación la realiza el ultimo servomotor. La composición de la articulación se puede observar en la Figura 37.

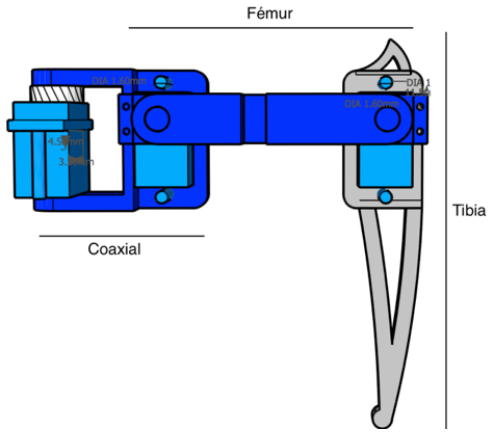


Figura 37. Diseño de la articulación del Robot

3.3.3 Montaje de SpiderBot

En la sección anterior se mostró el proceso de diseño y elaboración del PCB, en la ***Figura 38*** se muestra la placa de control con todos sus elementos solados, totalmente terminada y funcional.

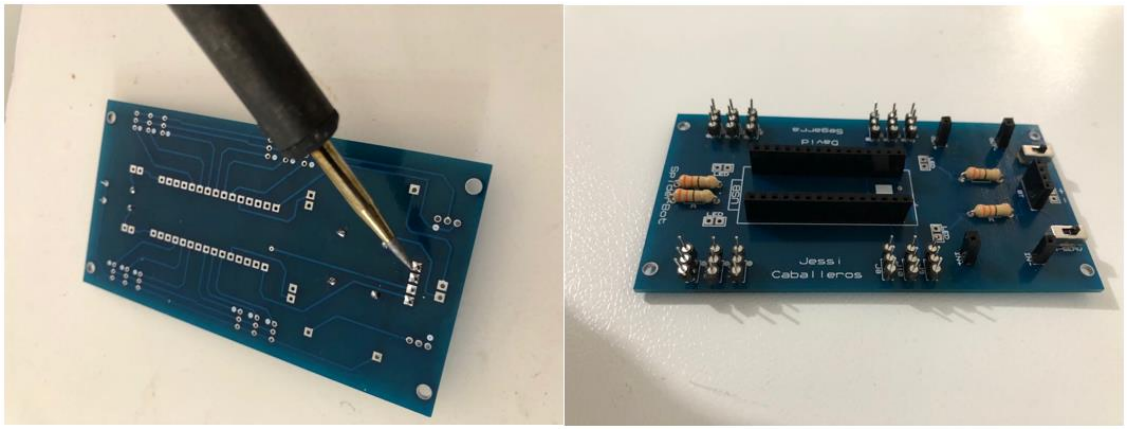


Figura 38. Placa PCB final de SpiderBot

Una vez obtenida la parte mecánica del Robot procedemos a Fijar y armar cada parte del SpiderBot, construimos cuatro patas con tres grados de libertad, un servomotor por cada DOF, el resultado de unir todas las piezas se puede observar en la ***Figura 39***.

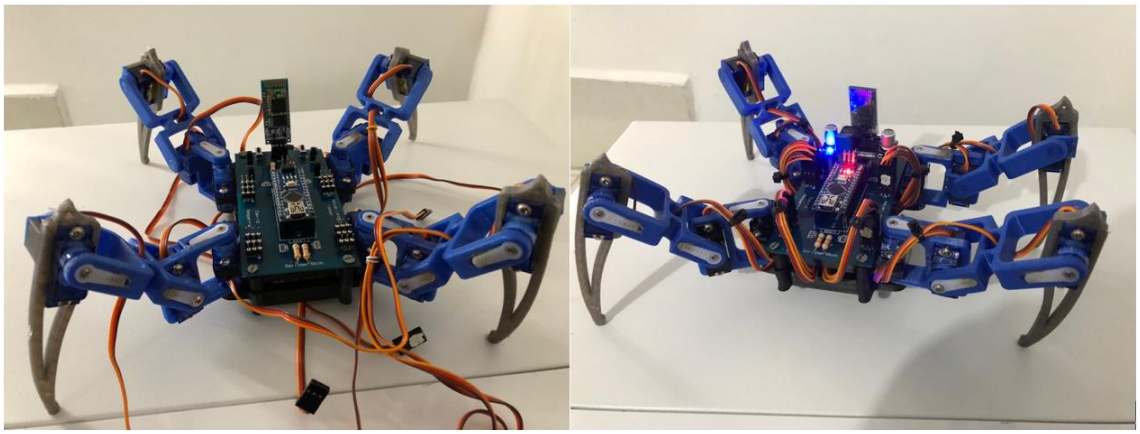


Figura 39. Implementación de SpiderBot

CAPITULO IV

IMPLENTACIÓN DE ALGORITMOS NECESARIOS PARA EL PROYECTO

En este capítulo se presenta el diseño y desarrollo de los algoritmos utilizados para la realización del proyecto, en los cuales establece su modelo cinemático, el algoritmo de movimiento, métodos para la detección de obstáculos y por último los algoritmos para la planificación de trayectoria.

Una vez que se ha definido la estructura mecánica que forma el esqueleto de SpiderBot con las dimensiones de cada una de sus extremidades, se plantea como objetivo lograr la locomoción del Robot. Para poder determinar que estado deben tener las articulaciones para posicionar el extremo de sus patas en un punto determinado, respecto al sistema de coordenadas del plano de desplazamiento, se modela la cinemática de una de las patas, y replicar dicho análisis en las demás. El procedimiento utilizado para desarrollar el control de posición de las articulaciones en coordenadas espaciales es el método propuesto por Denavit-Hartenberg, modelando la pata como una cadena cinemática de eslabones para así calcular las coordenadas del punto final en función de los ángulos de cada articulación que conforman la extremidad.

Los robots articulados presentan mejoras respecto a los que tienen ruedas, ya que pueden moverse de forma variada y ser utilizados en terrenos irregulares, pero también al tener grandes beneficios también conlleva más costo computacional, complejidad mecánica mayor y complicaciones al momento de programar su algoritmo de movimiento, para empezar, se debe analizar la forma de dar los pasos para un robot cuadrúpedo, primero se estudió la literatura sobre la marcha de animales que caminan en cuatro patas, como los perros y gatos, para poder establecer una estabilidad pasiva durante el movimiento, esto consiguió manteniendo una velocidad

relativamente baja y manteniendo 3 patas en el piso y una en movimiento, formando un trípode. El algoritmo de movimiento se programa en el microcontrolador ATMega328P de SpiderBot, utilizando el lenguaje de programación C, basándonos en los resultados obtenidos del modelo cinemático de cada extremidad del Robot.

Hoy en día existen varios métodos para la generación y planeación de rutas, desde los algoritmos más simples hasta algoritmos complejos de programación, por lo tanto, es importante elegir un procedimiento que se adapte a nuestras necesidades y aplicación. En este proyecto se implementa un nuevo algoritmo de planificación de trayectoria para un Robot Cuadrúpedo que se mueve entre obstáculos poliédricos tridimensionales. El algoritmo tiene una serie de ventajas: es simple de implementar, es rápido para la aplicación y puede tratar entornos no definidos y convexos. Uno de los objetivos del proyecto es tener la capacidad de planificar rutas automáticas sin colisiones, especificando la trayectoria a nivel de tarea, como "del punto inicio A moverse al punto final B", sin la necesidad de tener que enviar comandos a nivel de Robot, como "mover a 0.1, 0.35, 1.6". La programación a nivel de tarea es uno de los objetivos principales de la investigación en robótica. (Lozano-Perez, 1987) ya que consigue una navegación autónoma.

El inconveniente con la planificación de trayectoria, en su forma más simple, es encontrar una ruta desde una posición determinada de inicio a una posición de objetivo especificada que evite colisiones con un conjunto conocido de obstáculos estacionarios, aquí radica el primer problema de la planificación, ya que es necesario tener un algoritmo que detecte obstáculos. Para que el Robot Articulado obtenga la capacidad de navegar autónomamente es necesario que previamente se conozca el entorno de trabajo y la ubicación de los obstáculos. Esto se consigue por medio de un sensor visual, que visualizara todo el espacio en el que puede moverse el Robot, para la

detección de obstáculos se utiliza dos métodos, detección utilizando el color del objeto y la detección utilizando la morfología del obstáculo.

4.1 Algoritmo de Movimiento

Para establecer la secuencia de paso que va a realizar nuestro Robot Cuadrúpedo con tres grados de libertad por pata primero se estudió la literatura sobre la marcha de animales mamíferos que caminan en cuatro patas, como los gatos y perros, logrando distinguir dos tipos de pasos, llamados paso de trote y paso de arrastre, esta última es la utilizada para implementar en el software de control, ya que es la más simple de las dos y ofrece estabilidad y eficiencia al desplazarse.

4.1.1 Paso de arrastre

El modo de caminar lento es el modo de caminar más fácil de usar. El robot mantiene tres pies en el suelo conocido como “paso del trípode” y mantiene su centro de gravedad (CG) dentro del triángulo formado por esos tres pies. Si el CG sale de este triángulo por mucho tiempo, se caerá en la Figura 40 se muestra el triángulo que debe formar a la hora de caminar.

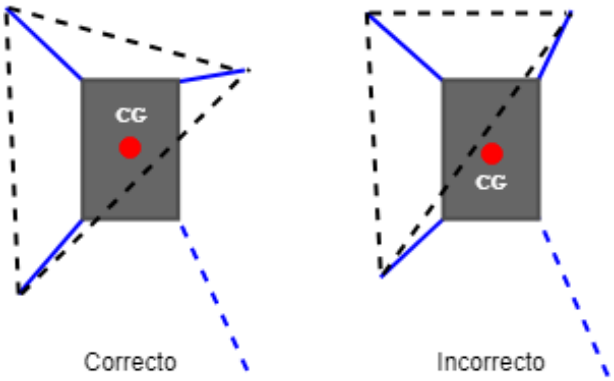


Figura 40. Estabilidad del trípode

4.1.2 Análisis de secuencia de pasos

Cuando los cuadrúpedos caminan de forma constante, las cuatro patas se mueven coordinadamente y cada una de las patas se divide en dos estados diferentes, estado de soporte y estado de elevación, en la primera fase hay al menos tres patas tocando el suelo, mientras que en

la fase de elevación solo hay una pierna levantada sobre el suelo. La relación entre el tiempo que el pie está en el piso y fuera del suelo, es decir, el ciclo de trabajo es 0.75.

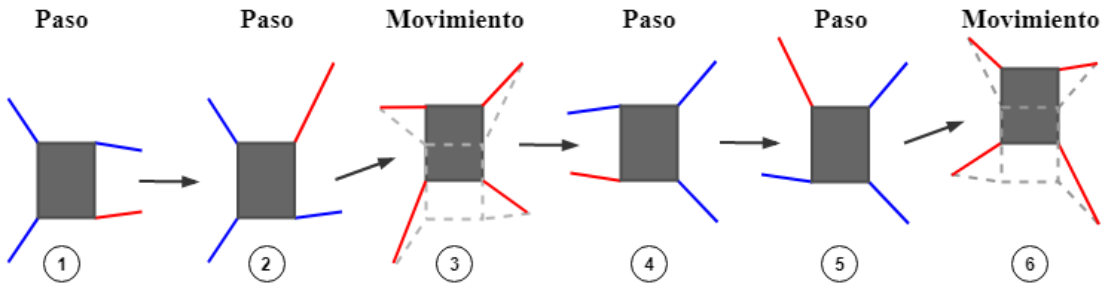


Figura 41. Secuencia de pasos de la marcha de arrastre

La secuencia de pasos que sigue SpiderBot se muestra en la Figura 41.

1. Esta es la posición inicial, con dos extremidades extendidas hacia un lado y las otras dos hacia adentro.
2. La extremidad superior derecha se eleva y se estira, muy por delante del robot.
3. Todas las extremidades se mueven hacia atrás, moviendo el cuerpo hacia adelante.
4. La extremidad trasera izquierda se eleva y avanza al costado del cuerpo. Esta posición es la misma imagen de la posición inicial.
5. La extremidad superior izquierda se eleva y se estira, muy por delante del robot.
6. Nuevamente, todas las patas se mueven hacia atrás, moviendo el cuerpo hacia adelante.
7. La pierna trasera derecha se levanta y regresa al cuerpo, regresando a la posición inicial.

El patrón de locomoción que sigue SpiderBot se divide esencialmente en dos conjuntos de movimiento, paso, paso y cuerpo, seguido de otro paso, paso y cuerpo en el otro lado. También se debe tener en cuenta que, en todo momento, el triángulo formado por las patas en el suelo contiene el centro de gravedad dentro del triángulo de estabilidad. Esta es la esencia de la marcha de arrastre.

4.1.3 Movimiento de rotación

Para que el Robot Cuadrúpedo realice el giro hacia la derecha o izquierda seguimos el siguiente patrón de pasos, como se muestra en la Figura 42.

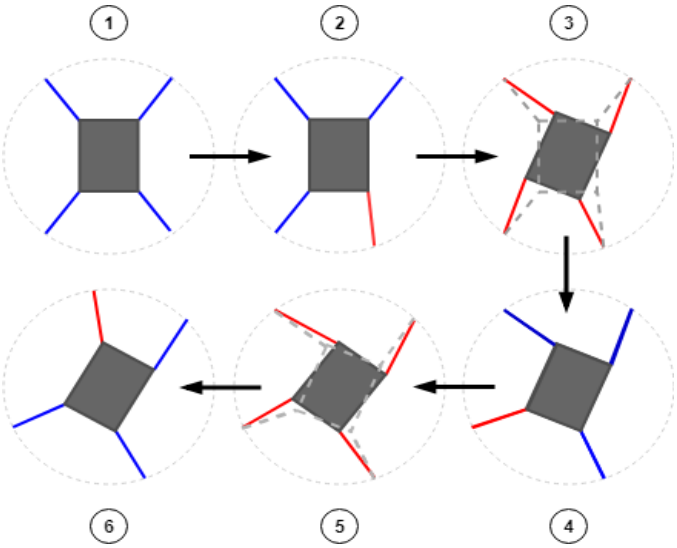


Figura 42. Secuencia de pasos para el movimiento de rotación

A continuación, se describe los pasos que realiza SpiderBot para un giro hacia la derecha.

1. Esta es la posición inicial, con las cuatro extremidades extendidas hacia afuera.
2. La extremidad inferior derecha se eleva y gira al costado derecho del cuerpo en sentido horario.
3. Todas las extremidades mantienen su posición y la articulación coxa realiza un movimiento angular en sentido horario, girando el cuerpo hacia la derecha.
4. La extremidad inferior izquierda se eleva y gira al costado del cuerpo en sentido horario.
5. Todas las extremidades mantienen su posición y la articulación coxa realiza un movimiento angular en sentido horario, girando el cuerpo hacia la derecha.
6. La pata superior izquierda se eleva y gira al costado del cuerpo en sentido horario.

Cabe mencionar que durante el movimiento de rotación solo actúan los servomotores de la articulación denominada coxa, las cuales están ligadas al cuerpo del robot, recordar que en todo momento se mantiene el mismo eje de rotación, girando en sentido antihorario cuando la rotación

es hacia la izquierda y en sentido horario cuando la rotación es hacia la derecha. El patrón que sigue el movimiento de rotación consiste en mover una extremidad y girar el cuerpo, de esta forma se obtiene una rotación completa

Una vez determinado el patrón de pasos que va a realizar el Robot procedemos a traducir todo el análisis en lenguaje C++ para la programación en el ATMega328P, en el ANEXO 2 se puede observar el programa completo para el control de movimiento del Cuadrúpedo.

4.1.4 Estructura de la programación

Las funciones utilizadas para el algoritmo de movimiento del Robot básicamente cumplen la función de controlar los doce servomotores de las patas de SpiderBot, los valores de los ángulos que debe girar cada articulación se envía secuencialmente a cada motor de cada extremidad, pero la respuesta mecánica del sistema al comando de movimiento, en comparación con la velocidad de la señal electrónica enviada por el controlador son al mismo tiempo, haciendo que el movimiento de las articulaciones sean simultaneas.

A continuación, en la Figura 43 se muestra el número asignado a cada extremidad para mantener un orden en la programación.

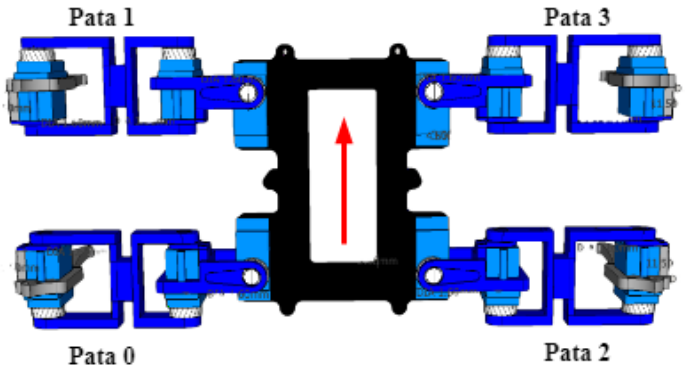


Figura 43. Distribución de patas en SpiderBot

En el siguiente diagrama de flujo de la Figura 44 se muestra la lógica de programación que se usó para el control de movimiento.

El sistema central ubicado en el computador es el encargado de enviar al microcontrolador el tipo de comando para ejecutar la función de desplazamiento en el algoritmo de movimiento.

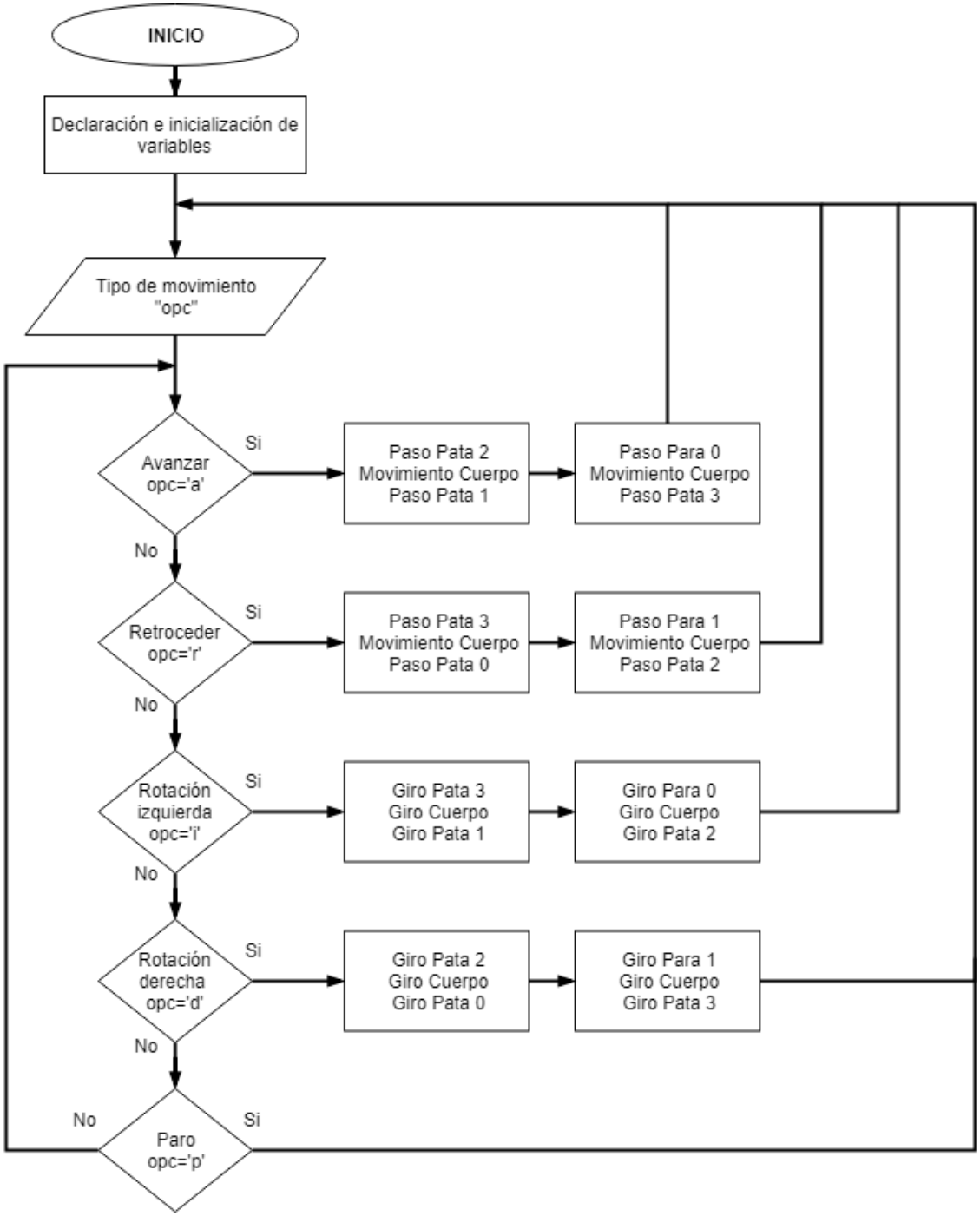


Figura 44. Diagrama de Flujo para el control de movimiento de SpiderBot

4.2 Modelo Cinemático

En el análisis cinemático hay que asignar un marco de referencia a cada eslabón que existe dentro de la cadena cinemática (los ejes, eslabones y la forma de movimiento de cada articulación se muestran en la Figura 45). El primer marco está en la base del robot y el último en su punto terminal también llamado TCP por sus siglas en inglés (*Tool Central Point*). La convención fundamental para asignar marcos de referencia es la de Denavit y Hartenberg, la cual ayuda a la modelación del robot y la eficiencia en cuanto a cálculos computacionales. (Barrientos Antonio , Peñín Luis F., Balaguer Carlos, 1997; Spong MW, Hutchinson S, 2004).

Para realizar el modelo cinemático y dinámico es indispensable aplicar el método de Denavit y Hartenberg de manera que se describa al robot en función de las variables de cada una de las articulaciones (rotación) y los parámetros estáticos de los eslabones (desplazamiento).

La Figura 45 muestra la simbología necesaria para representar a SpiderBot, sobre esta representación están los 3 marcos de referencias para modelar cada de una de las patas de SpiderBot. De acuerdo con el convenio de Denavit y Hartenberg los marcos de referencia se colocan teniendo en cuenta lo siguiente (Cardoso, Fernández, Marrero, & Guardado, 2017):

1. Las articulaciones se enumeran desde $i = 1$ hasta n, siendo i, la i-ésima articulación. Para el caso de SpiderBot, existen tres servomotores en cada para, por lo tanto hay tres articulaciones.
2. Los ejes z_i se ponen a lo largo de la articulación $i+1$ sucesiva.
3. Si z_i y z_{i-1} se cruzan, el origen del marco $x_iy_iz_i$ se sitúa en ese punto.
4. El eje x_i se toma a lo largo de la normal común entre los ejes z_i y z_{i-1} con dirección desde la articulación i hacia $i+1$. Por ejemplo, en la Figura 45 entre z_0 y z_1 la normal queda en dirección al eslabón 2, es ahí donde se coloca a x_2.

5. El eje y_i se toma de tal forma que cumpla la regla de la mano derecha, de forma completando así el marco de referencia.
6. Para la base ($x_0y_0z_0$) solo se determina la dirección del eje z_0 y se elige x_0 y y_0 a conveniencia. Aquí escogimos que x_0 vaya a lo largo del eslabón 1.

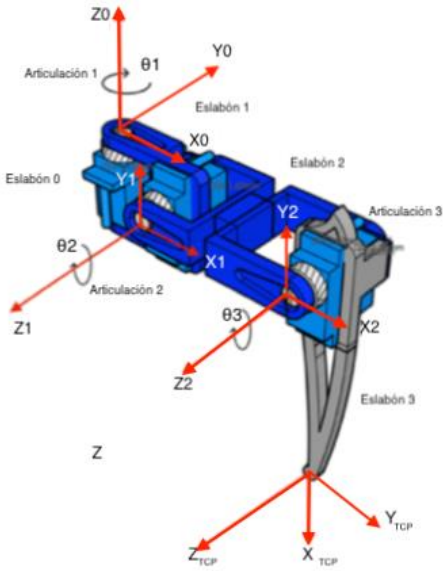


Figura 45. Ejes de movimiento de cada pata.

En la Ecuación (1) se muestra la matriz con el análisis matemático resultado de las consideraciones anteriores del método de Denavit y Hartenberg.

$$N = \begin{bmatrix} \theta_1 & 0 & 3 & \pi/2 \\ \theta_2 & 0 & 5.4 & 0 \\ -\pi/2 + \theta_3 & 0 & 7.8 & 0 \end{bmatrix} \quad (1)$$

Con el uso del software de Matlab, se creó una simulación con el movimiento de las patas y sus tres articulaciones introduciendo valores de 3 ángulos, cada uno de los cuales modifican una articulación, como se muestra en la Figura 46, el código de esta simulación se encuentra en el ANEXO 3.

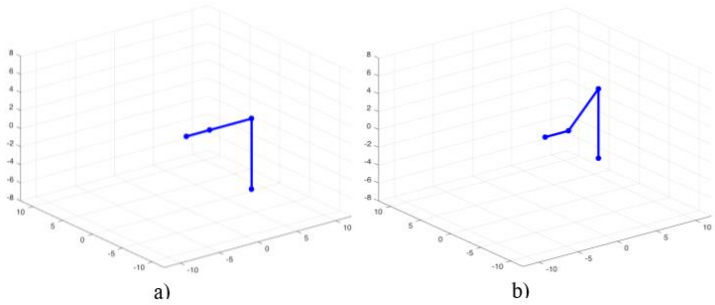


Figura 46. Simulación: a) θ_0=0, θ_1=0, θ_3=0
b) θ_0= 0, θ_1= π/4, θ_3= $-\pi$/4

4.3 Detección de Obstáculos y Objetos

La toma de decisiones se basa en la información recogida del entorno en el que se encuentra el robot a través del sensor visual, la cámara, desde una perspectiva cenital.

Para realizar el análisis y procesamiento de imágenes para la evasión de obstáculos se utilizará la librería OpenCV, la cual está diseñada para una eficiencia computacional, proporciona una simplificada utilización de su infraestructura para realizar visión por computadora y está disponible para el lenguaje de programación de Python cuyo desarrollo se fundamenta en código abierto; este último lenguaje mencionado se utilizará como base para el desarrollo computacional en el método de evasión de obstáculos.

Para la localizar los obstáculos se realizará una detección por medio de morfología, es decir se los encontrará por su forma y en este caso serán rectángulos o cuadrados. Además de usar visión por computadora para localizar los obstáculos, se la utilizará para hallar a la araña en el entorno de prueba, esto se lo realizará por medio del color a diferencia de la morfología.

Con esta información el robot implementado realiza una determinada tarea, esta es la de seguir una ruta calculada automáticamente por un algoritmo desde un punto inicial A, a un punto final B, evadiendo los obstáculos.

4.3.1 Detección basada en color

Para la detección basada en color se utilizará una técnica llamada Thresholding, la cual es esencial para la visión por computadora y el procesamiento de imágenes en donde se reemplazan cada píxel en una imagen con un valor binario, “1” o “0”, dependiendo de si los píxeles coinciden con ciertos criterios, con estoque se crean una versión en blanco y negro de la imagen original, en la cual se destacan las características de los criterios mencionados anteriormente como se puede observar en la Figura 47.

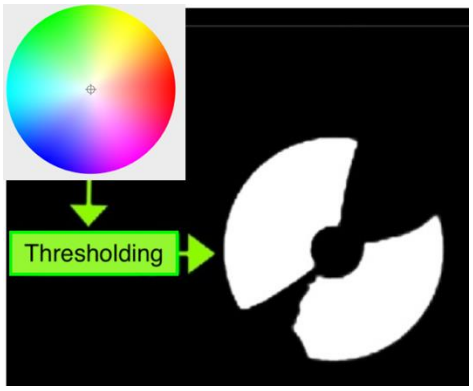


Figura 47. Proceso de Thresholding

4.3.1.1 Descriptor de imagen

Por lo general, las imágenes se representan como una secuencia de 3 elementos: Rojo, Verde y Azul (RGB) por sus siglas en inglés *Red, Green, Blue*. El espacio de color RGB puede ser representado como un "cubo", como se muestra en la Figura 48 a).

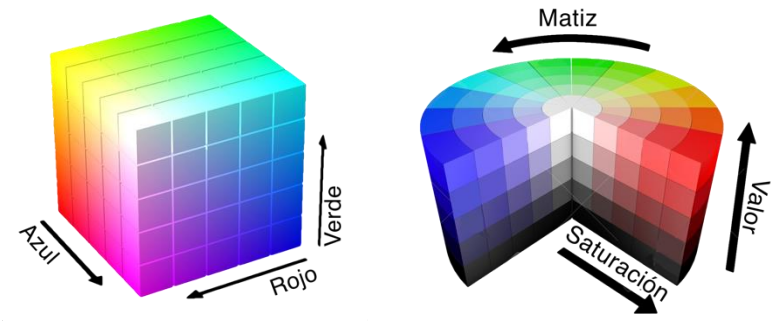


Figura 48. a) Interpretación RGB, b) Interpretación HSV

A pesar de que los valores RGB son simples de entender, el espacio de color RGB no reproduce cómo los humanos perciben el color. En su lugar, se va utilizar el espacio de color Matiz/Color, Saturación y Valor (HSV) por sus siglas en inglés *Hue, Saturation, Value*. HSV que mapea las intensidades de los píxeles en un cilindro como se puede observar en la Figura 48 b). Existen diferentes rangos para los valores de H, S y V pero los que se utilizan en OpenCV son H de 0 a 180, S y V de 0 a 255, con estos tres valores se puede definir a un color. Encontrar un color en el espacio HSV es una pregunta antigua pero común.

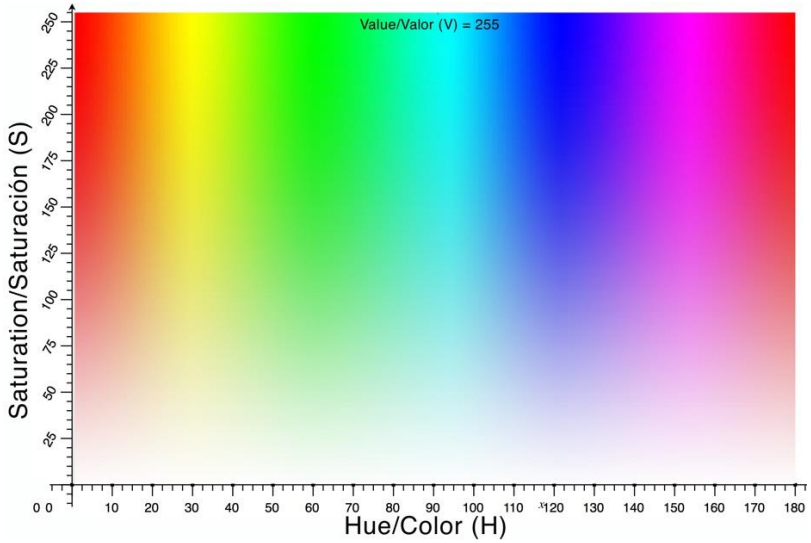


Figura 49. Mapa de Color HSV con V=255

En la Figura 49 se puede observar un mapa de color HSV para buscar rápidamente un color en especial. El eje x representa Color (H) [0,180], el eje y representa Saturación (S) [0,255], mientras que se mantiene Valor (V) en 255. Para encontrar un color, generalmente solo se busca el rango de H y S, y establece V en el rango [20, 255]. Por ejemplo, para hallar el color azul, se busca en el mapa y observamos el mejor rango: H: [105, 125], S: [150, 255] y V: [20, 255]. Entonces la máscara en Open CV es:

```
mask = cv2.inRange(hsv, [105, 150, 20], [125, 255, 255])
```

Luego usamos el rango encontrado para buscar el color azul, obteniendo así el resultado de la Figura 50. En la imagen en blanco y negro, las áreas blancas son píxeles que caen dentro del rango de color que se configuró. Se puede ver que aparece el lado de cubo azul, pero también lo hacen los objetos con superficies azuladas como la caja de fósforos.

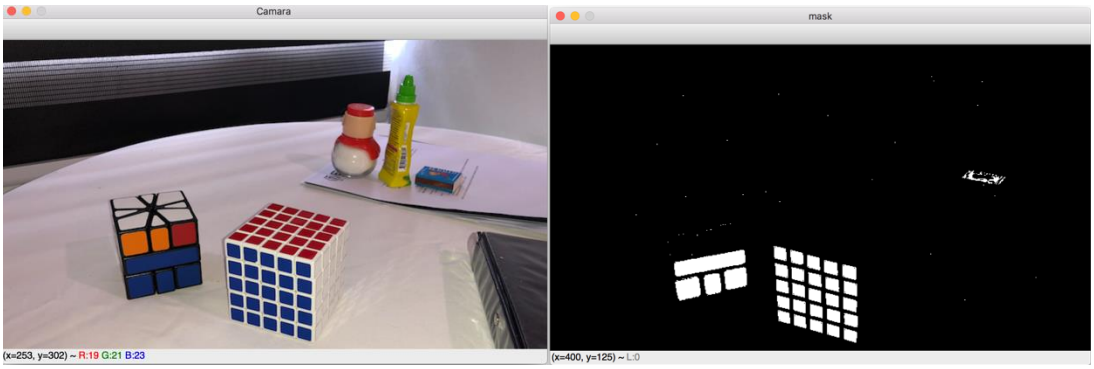


Figura 50. Resultado de Máscara con el color azul

En la máscara de la Figura 50 se puede observar que se detectan muy bien el color azul de la imagen original, pero existen ciertos falsos positivos en ella, vistos como pequeños puntos blancos en toda la máscara para poder corregir esto se realizará un filtrado del ruido con un CLOSE seguido de un OPEN.

```
kernel = np.ones((6,6),np.uint8)

mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
```

Dando como resultado la máscara obtenida en la Figura 51 en la cual se puede observar que no existen ya los pequeños puntos de ruido o falsos positivos.

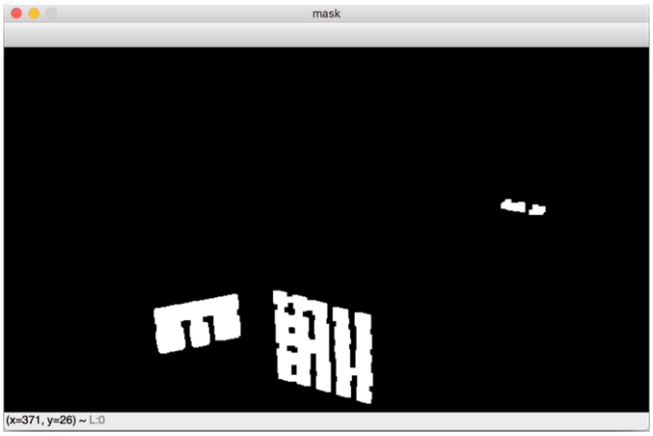


Figura 51. Máscara Filtrada con OPEN y CLOSE

4.3.1.2 Rangos HSV y Colores utilizados para SpiderBot

El robot tiene dos colores para ubicar a SpiderBot (ver Figura 52), un círculo rojo atrás, y en la parte frontal, uno verde. Se utilizan dos colores debido a que, además de conocer la posición, se necesita saber la dirección a la que está enfrentando para poder enviar los comandos de movimiento para seguir la trayectoria que el algoritmo de ruta va a encontrar.

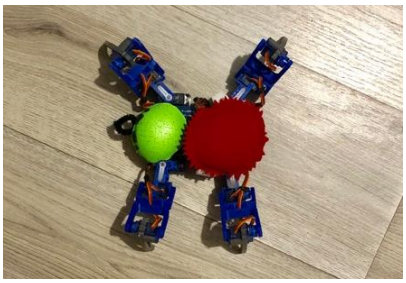


Figura 52. SpiderBot con Colores para Detección

Para detectar al rojo observamos nuevamente la Figura 49 y elegimos los rangos HSV en los que el color aparece. Para H el rango apropiado es [170,180], para S se eligió un rango grande para poder detectar tonos claros si hay mucha luz, este fue [100, 255] y por último se seleccionó V, igualmente con un rango grande, con [10, 255] para detectar tonos más obscuros. Con esto y

aplicándolo a la imagen de prueba se obtiene la Figura 53, comprobando así que los rangos elegidos son los correctos para poder detectar al color rojo.

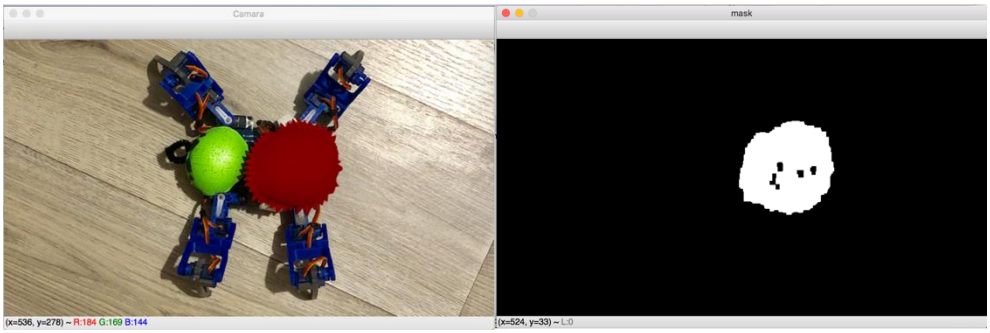


Figura 53. Resultado de Máscara con el color rojo

Por último, se necesita detectar el color verde, de nuevo elegimos rangos para H, S, y V observando la Figura 49. Para H se eligió [40, 80], para S se seleccionó [100, 255] y para V se optó por [20, 255] al igual que en el color rojo, obteniendo así la máscara de la Figura 54.

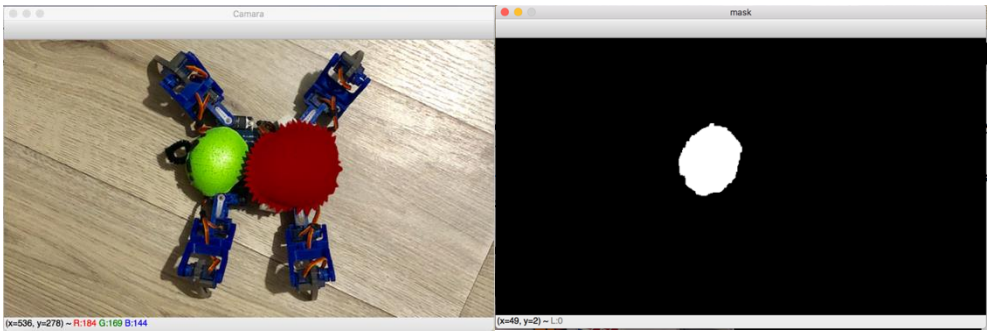


Figura 54. Resultado de Máscara con el color verde

4.3.1.3 Encontrar a SpiderBot

La Figura 53 y Figura 54, es decir, las máscaras que se obtuvieron como resultado del Thresholding no son de mucha utilidad para el robot cuadrúpedo o para el algoritmo de planificación de ruta. Se necesita, de alguna manera, encontrar las posiciones de las áreas blancas. Usando una función llamada "EnclosingCircle" podemos convertir estas áreas blancas en matrices de puntos centrales y radios, es decir círculos con un punto central, estos círculos luego se aplican a la imagen original (ver Figura 55) y de esta manera no se podrá obtener una variedad de

posiciones con los valores x e y que nuestro programa puede usar para colocar el robot en el espacio y luego enviar comandos para avisarle a dónde ir. El código implementado para realizar la detección de obstáculos se encuentra en el ANEXO 4.

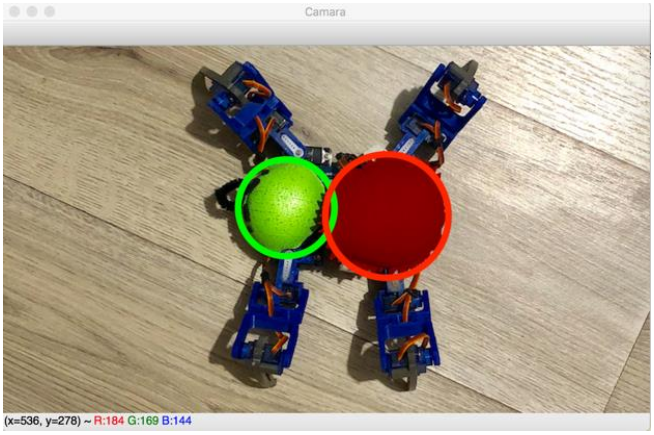


Figura 55. Detección de la posición de SpiderBot con la función EnclosingCircle.

4.3.2 Detección basada en morfología

Para los encontrar a los obstáculos se lo hará de una manera diferente a la detección de SpiderBot, ya no usando color, si no usando morfología, es decir por la forma del objeto. Se hará cierto procesamiento a la imagen detectada por la cámara para que, como resultado final, los obstáculos sean hallados, estos tendrán una forma rectangular definida anteriormente.

El proceso que hay que seguir para realizar la detección por formas se muestra en la Figura 56.

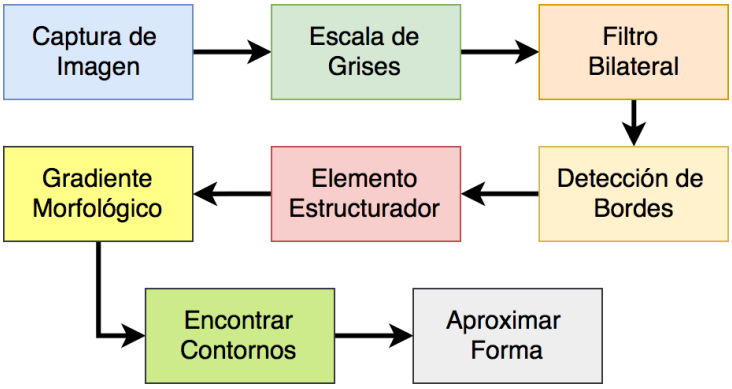


Figura 56. Proceso de detección de Formas

Se toma la imagen capturada por la cámara, y hay que asegurarse que la misma esté en el descriptor de imagen correcto con el que OpenCV trabaja, este es BGR, a diferencia de lo usual que es RGB, esto se debe a que BGR era un formato popular en los años cuando se creó OpenCV y con el paso de los años nunca cambio este estándar en la librería. El primer paso posterior a este es poner a la imagen en una escala de grises como se muestra en la Figura 57, esto se realiza debido a que no se requiere de información de color para este caso.

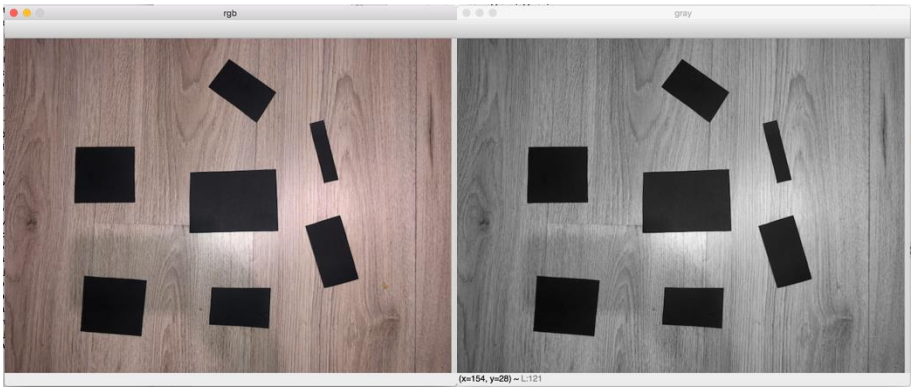


Figura 57. Detección de Formas: Escala de grises

Luego, a esta imagen en blanco y negro se le aplica un Filtro Bilateral, el cual es altamente efectivo en la eliminación del ruido mientras mantiene los bordes afilados, este filtro se asegura de que solo los píxeles con una intensidad similar al píxel central se consideren borrosos, por lo tanto, conserva los bordes ya que los píxeles en los bordes tendrán una gran variación de intensidad con el píxel central. En la Figura 58 se muestra el cambio de la imagen luego de haber sido aplicado el filtro, toda la imagen se ha puesto borrosa, pero no se han perdido los bordes de los rectángulos los cuales queremos detectar.

Figura 58. Detección de Formas: Filtro Bilateral

Después, se realiza una detección de bordes mediante Canny el cual es un algoritmo desarrollado por John F. Canny, OpenCV tiene lo tiene implementado en sus funciones, con ello lo que finalmente conseguimos son bordes fuertes en la imagen, esto se puede observar en la Figura 59 donde se distinguen claramente los bordes de los rectángulos.

El siguiente paso por realizar es definir al elemento estructurador del tamaño y la forma especificados para ejecutar las operaciones morfológicas, este es un kernel con el cual dice cómo cambiar el valor de un píxel dado combinándolo con diferentes cantidades de los píxeles vecinos. El kernel se aplica a cada píxel de la imagen uno por uno para producir una la imagen final. En este caso tendrá una forma rectangular. Con este kernel se puede realizar el Gradiente morfológico, el cual es la diferencia entre la dilatación y erosión de la imagen.

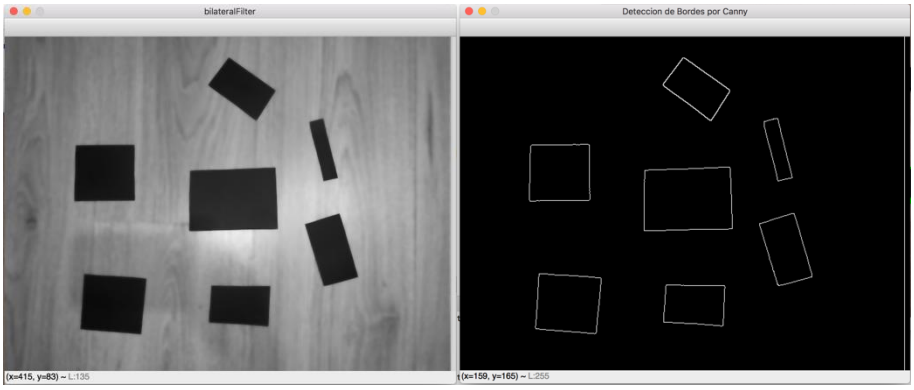


Figura 59. Detección de Formas: Detección de bordes con Canny

El principio básico de la erosión es igual que la erosión del suelo, se erosiona los límites del objeto en primer plano. Para esto el kernel se desliza a través de la imagen de entrada píxel a píxel y se considerará 1 solo si todos los píxeles debajo del kernel son 1, de lo contrario se erosiona (se convierte en cero). La dilatación simplemente es el opuesto de la erosión se considerará 1 si uno de los píxeles debajo del kernel es 1. Al unir a estos dos procesos obtenemos el Gradiente morfológico que se muestra en la Figura 60. Aunque en este caso no se nota mucho la diferencia entre la imagen de la izquierda y la de la derecha, existirán algunos casos en que el Gradiente morfológico ayude a mejorar la imagen cuando que exista más ruido en ella y las formas no sean muy definidas luego de la detección de bordes por Canny.

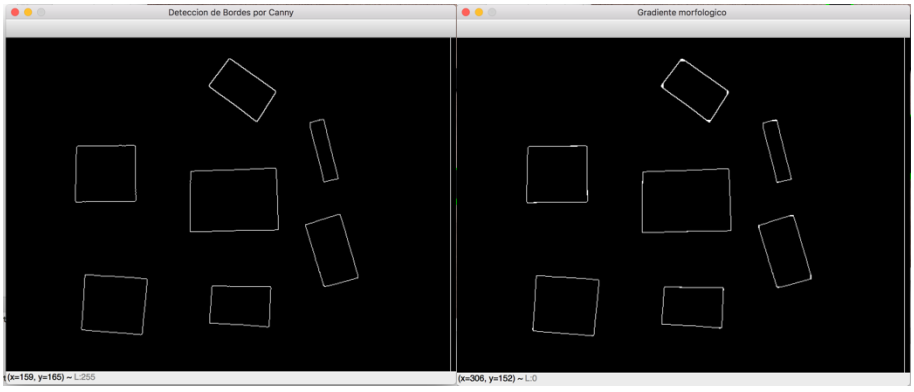


Figura 60. Detección de Formas: Gradiente Morfológico

A continuación, lo que se necesita realizar es utilizar la función "findContours" que encuentra todos los "contornos" o bordes de una imagen como una matriz de puntos. De hecho, crea una matriz de estos conjuntos de puntos, cada conjunto de puntos representa una región distinta en la imagen. Es decir, de la imagen binaria de la Figura 60 se transforma a coordenadas de todos los pixeles con el valor de 1 como se ve en la Figura 61, en ella solo muestran cinco coordenadas encontradas de la imagen con los rectángulos pero en realidad son bastantes las coordenadas obtenidas, las cuales se intentarán reducir en el siguiente paso.

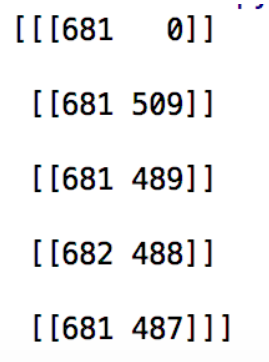


Figura 61. Detección de Formas: Encontrar Contornos

Y por último, se requiere aproximar la forma, con esto, lo que se intenta es aproximar una forma de contorno a otra forma con menos vértices dependiendo de la precisión que se especifique, en este caso se requieren formas rectangulares. El código implementado en OpenCV es una implementación del algoritmo Douglas-Peucker y se lo puede realizar mediante el comando cv2.approxPolyDP. Con esto se obtienen las coordenadas de cada vértice de los 7 rectángulos en el ejemplo realizado (ver Figura 62).

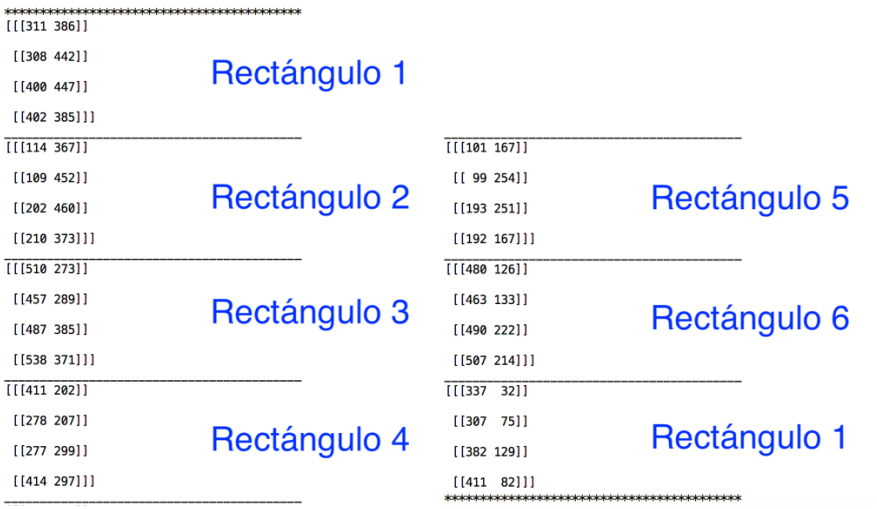


Figura 62. Detección de Formas: Aproximación de Forma

Para poder comprobar que estas coordenadas son las indicadas, se pueden dibujar polígonos mediante el comando cv2.drawContours en la imagen original, y estos deben coincidir con los

rectángulos originales como se puede apreciar en la Figura 63. El código implementado para realizar la detección de obstáculos se encuentra en el ANEXO 5.

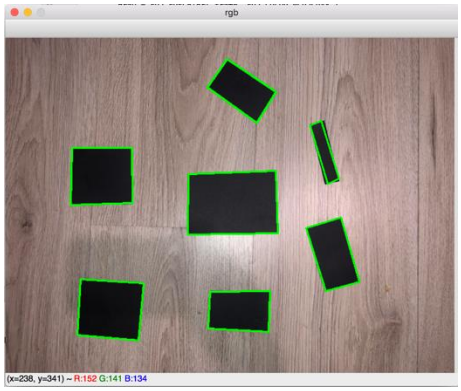


Figura 63. Comprobación Final de Detección de Rectángulos

En la Figura 64 se puede observar una imagen de cómo ve la cámara en perspectiva cenital, en esta se puede ver un obstáculo detectado y los dos colores colocados sobre la araña.

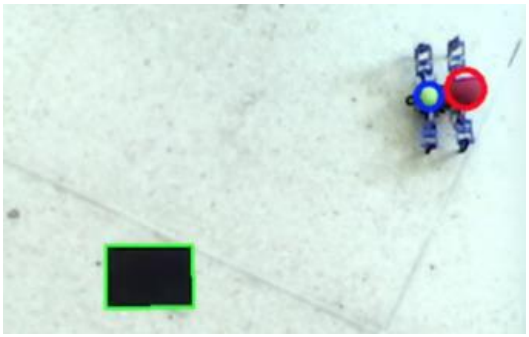


Figura 64. Robot Cuadrúpedo y Objetos desde perspectiva cenital

4.4 Algoritmos para la planificación de trayectoria

En la actualidad el uso de robots con capacidad de desplazamiento está en auge, cada vez se crean más de estos para aplicaciones en campos de la medicina, industriales, laborales, etc., o en actividades más cotidianas como limpieza, vigilancia, fumigación, riego, usándolos en hogares y oficinas, su única restricción es la necesidad de tener una mayor capacidad de movimiento.

La planificación de ruta cumple un rol esencial dentro de la rama de la robótica, ya que permite desarrollar las labores demandadas a los autómatas. Cada día las exigencias van en aumento requiriendo que los robots se desenvuelvan en entornos reales, dejando a un lado los espacios de trabajo en los laboratorios.

Dentro de la planificación de ruta es necesario conocer la respuesta a dos preguntas básicas ¿Dónde estoy? y ¿Cómo es el entorno que me rodea?, la primera pregunta se refiere a la posición y orientación del robot dentro del área de trabajo, y la segunda pregunta hace referencia a conocer los obstáculos que lo rodean, para esto es necesario implementar sensores dentro del ambiente o en el autómata.

El problema de la planificación de ruta es tan antiguo como los robots, y muy importante de abordar ya que permite la navegación autónoma de los mismos, para realizar una tarea específica en un espacio limitado, conformado por otros objetos que podrían resultar como obstáculos a la hora de ejecutar una misión. Irrefutablemente lo ideal sería que el robot este integrado completamente en su espacio de operación, de manera que pueda funcionar en conjunto con el resto de los elementos del entorno.

Al tratar con los problemas de la planificación de ruta, se sabe que no es nada sencillo y tiene varios inconvenientes por analizar, pero en este proyecto nos enfocamos en el problema inicial expresado en el marco teórico, que consiste en hallar un camino continuo libre de colisiones, para un objeto, conociendo su configuración inicial y su configuración de objetivo.

En esta sección se explica los beneficios de algunos métodos probabilísticos para la planificación de caminos, para concluir que algoritmo se va a implementar para SpiderBot.

4.4.1 A*

El algoritmo A* es uno de los métodos más populares para encontrar la ruta más corta ente dos configuraciones inicial y final en un área mapeada, generando el resultado en un tiempo finito. Fue desarrollado en 1968 para combinar enfoques heurísticos como Best-First-Search (BFS) y enfoques formales como el algoritmo de Dijsktra. (Kala, Shukla, & Tiwari, 2010)

Una de las características más importantes del algoritmo A*, es la construcción de dos listas, la primera que sirve para almacenar nodos para expansiones, denominada “lista abierta” registrando áreas adyacentes a las ya evaluadas y la segunda que sirve para almacenar nodos que hemos explorado, denominada “lista cerrada” registrando áreas ya evaluadas. También este algoritmo calcula las distancias recorridas desde el "punto de inicio" con distancias estimadas al "punto de meta". En la Figura 65 se muestra el diagrama de flujo del algoritmo A*, extraído de (Mills-Tettey, Lee-Shue, Prasad, & Tantisevi, n.d.).

Para comprender mejor el funcionamiento del algoritmo en la Figura 66 se muestra las partes de este, el recuadro color amarrillo representa la configuración inicial, la cual es parte de la lista cerrada al ser considerada una ubicación que ha sido evaluada y explorada. Los cuadrados color verde que rodean al recuadro amarillo son los bordes, y son las posibles opciones que puede experimentar el algoritmo, formando parte de la lista abierta ya que son las ubicaciones adyacentes a las áreas ya exploradas.

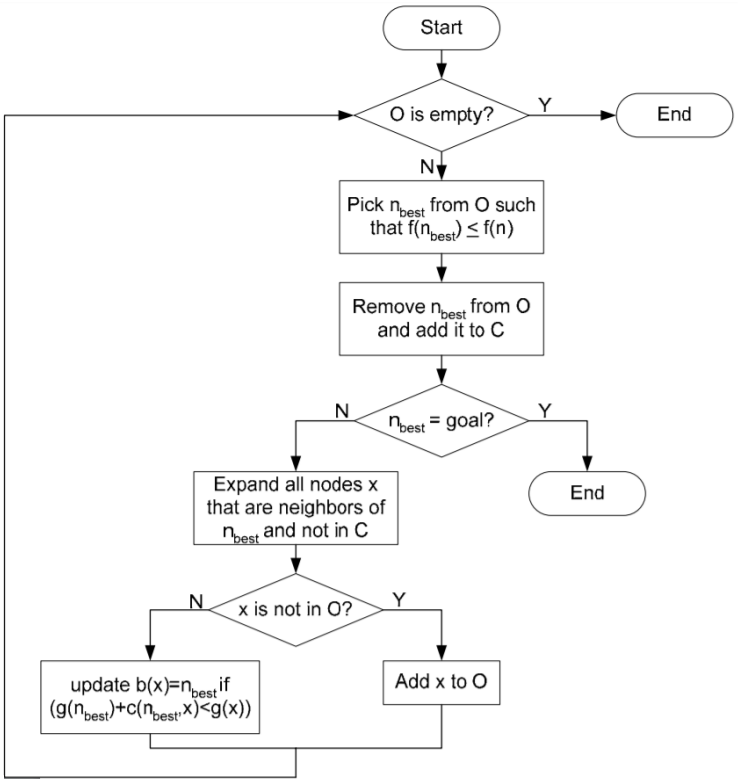


Figura 65. Diagrama de flujo del Algoritmo A*
Fuente: (Mills-Tettey et al., n.d.)

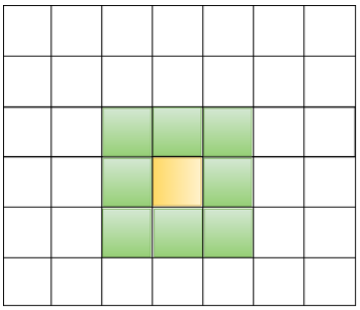


Figura 66. Configuración inicial del algoritmo A*

A medida que la ruta avanza, las listas cerradas y abiertas crecen, revisando y evaluando todo el mapa, cabe recordar que el algoritmo no rodea esquinas, por esta razón si se tiene obstáculos cuadrados es necesario aumentar en el algoritmo reglas para girar, esto se puede observar en la Figura 67, en el grafico superior se observa el algoritmo A* normal y en la parte inferior el algoritmo con reglas para girar, con esto la ruta será más adecuada para evitar obstáculos.

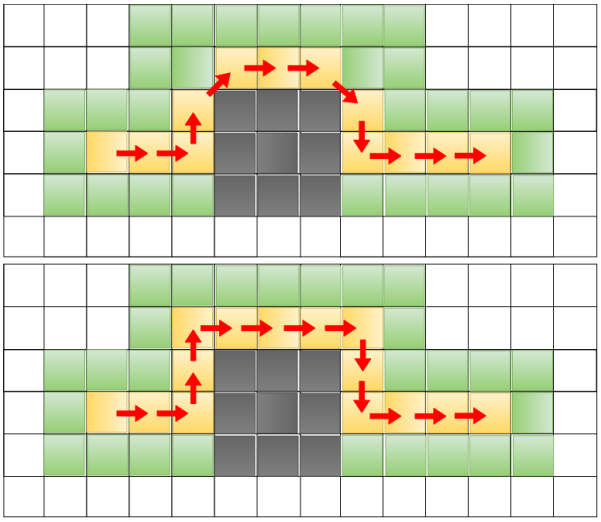


Figura 67. Algoritmo A*, sin y con reglas para virar

La heurística utilizada para calcular las distancias en el algoritmo A * es:

$$F(n) = G(n) + H(n) \quad (2)$$

La Ecuación (3) es una fórmula para calcular el valor de las posiciones adyacentes a la ubicación origen, y poder determinar que posición tiene el menor valor y así poder avanzar.

Donde *G(n)* representa el costo de distancia del camino desde el punto de partida hasta cualquier vértice n, los valores pueden ser de 10 (línea recta) o 14 (diagonal) dependiendo a que posición se va a dirigir, como se muestra en la Figura 68.

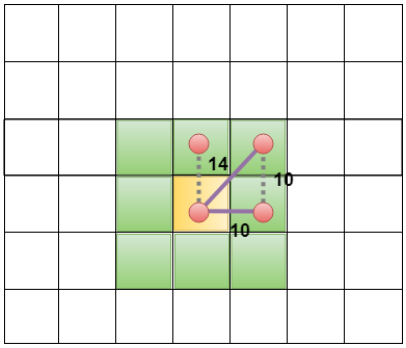


Figura 68. Representación de G(n) para calcular la distancia en el Algoritmo A*

Y *H(n)* representa el costo estimado desde el vértice *n* hasta el objetivo final, y se calcula como se muestra en la Figura 69 , contando el número de saltos desde la ubicación actual hasta, la configuración final y multiplicarlo por 10.

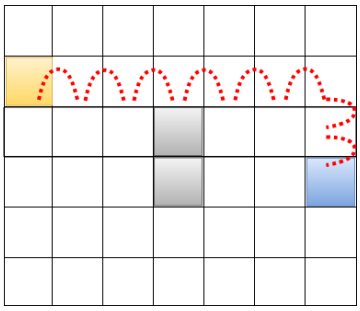


Figura 69. Representación del cálculo para H(n)=80

Cuando la ubicación origen está en línea recta con la configuración final, y se calcula como una distancia euclidiana (distancia entre dos puntos), *H(n)* mostrada en la Figura 70.

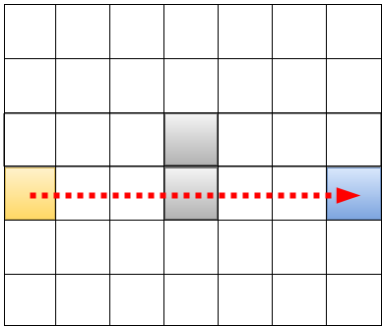


Figura 70. Distancia euclidiana para calculas H(n)=60

En conclusión, el algoritmo A* es simple de comprender y ejecutar, consiste en tener dos listas, una abierta y otra cerrada, con esto podemos jugar y combinar para que se desarrolle el algoritmo. Inicialmente la lista abierta contiene un solo elemento, la configuración inicial y la lista cerrada está vacía, esta lista va a contener todos aquellos nodos que han sido evaluados, en la Figura 66 se puede observar que la lista abierta es la frontera y la lista cerrada es el interior de las áreas adyacentes, no olvidar que cada nodo tiene un puntero que apunta hacia su nodo “padre”, esto servirá para formar la ruta al termino del algoritmo.

4.4.2 PRM

Los mapas de caminos probabilísticos (PRM), desde un inicio, han demostrado ser un método con un gran potencial para solucionar la problemática de hallar una trayectoria libre de colisiones dentro del entorno de trabajo, dicho método consiste en un pre-proceso y una fase de consulta. En el pre procesamiento se genera una hoja de ruta, que consiste en un conjunto de puntos denominados "nodos", distribuidos de forma aleatoria en el espacio de configuración, $Q_{free.}$ Posteriormente en la fase de consulta sirve para encontrar el camino mas corto, utilizando un planificador local, como el algoritmo A* o Dijsktra, el cual establece los enlaces que conectan los nodos para encontrar el camino que une la configuración inicial $q_{Start.}$ y final q_{Goal}, eludiendo aquellos nodos que produzcan una colisión. En la Figura 71 se muestra en la imagen izquierda el Roadmap, compuesto de enlaces y nodos y en la imagen de la derecha, la unión con enlaces de los nodos que conforman la trayectoria para unir el punto inicial y final sobre el Roadmap.

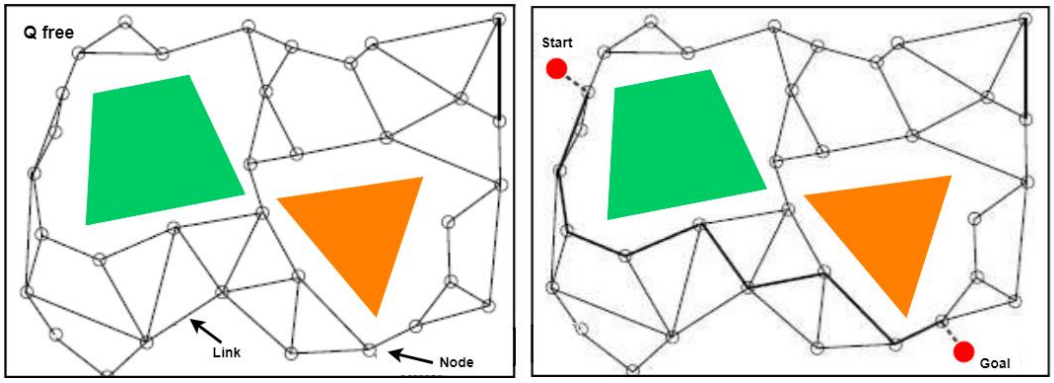


Figura 71. Método mapas probabilísticos

4.4.2.1 Algoritmo PRM

El conjunto de enlaces y nodos se lo llama Roadmap, a continuación, se muestra el algoritmo para desarrollar el Roadmap.

Algoritmo: Construcción del Roadmap.

Entrada: n es el número de nodos en G. k es el número de vecinos más próximos por cada configuración.

Salida: El roadmap G = (V, E).

```
1       V ← 0
2       E ← 0
3       Mientras |V | < n hacer
4           Repetir
5               q ← muestra()
6           hasta estaLibreColision(q)
7           V ← V ∪ {q}
8       Para toda q ∈ V hacer
9           Nq ← el k-ésimo vecino más cercano de
10          q de V usando dist
11          for all q' ∈ Nq hacer
12              si (q, q') ∉ E y caminoLocal(q, q' ) ≠ NIL
13              entonces
14                  E ← E ∪ {(q, q' )}
```

Un inconveniente en usar al algoritmo PRM es su costo computacional, ya que este depende del número de veces que se ejecute el algoritmo para detectar las colisiones y luego utilizar un método para trazar la trayectoria, para poder lograr que el algoritmo PRM sea más eficiente es necesario reducir el tamaño del Roadmap. Por esta razón con el paso del tiempo se han desarrollado algoritmos de naturaleza probabilística como Expansive Spaces Trees (EST) y Rapidly Exploring Random Trees (RRT), este último es el más utilizado actualmente.

4.4.3 RRT

El algoritmo RRT, es una técnica muy conocida en el área de la planificación de trayectorias, ya que es un algoritmo capaz de examinar eficientemente el entorno de trabajo, desde la

configuración inicial. En términos simples, RRT básico, construye un árbol T formado por nodos y enlaces, que crece paulatinamente, buscando estados alcanzables al intentar aplicar acciones aleatorias en estados de acceso conocido, en el algoritmo se muestra la creación de los nodos, libres de colisión q_{rand} y se utiliza la función "Extend" para que crezca el enlace

Algoritmo: Construcción RRT

1 $T(0) \leftarrow q_{start}$
2 for $i = 1$ to K do
3 $q_{rand} \leftarrow$ RANDOM_ CONFIG
4 EXTEND (T, q_{rand})
5 Return T

Algoritmo: EXTEND (T, q)

1 $q_{near} \leftarrow$ NEAREST-NEIGHBOR(T, q)
2 if NEW-CONFIG (q, q_{near}, q_{new})
3 ADD-VERTEX(T, q_{new})
4 ADD-EDGE(T, q_{near}, q_{new})
5 if q_{new} = q then
6 Return Reached
7 Else
8 Return Advanced
9 Return Trapped

Para poder hallar una ruta que esté libre de colisiones, el algoritmo comienza su exploración en la configuración inicial q_{start}, hasta alcanzar la configuración objetivo q_{goal}, con un numero K de iteraciones, al alcanzar la meta el algoritmo culmina y se traza la trayectoria desde q_{start} a $q_{goal.}$ Si en el entorno de trabajo existe más áreas con espacio libre, el RRT crece y avanza con mayor fluidez, ya que ahí existe más posibilidad de hallar puntos q_{rand} posibles, esto se puede observar en la Figura 72.

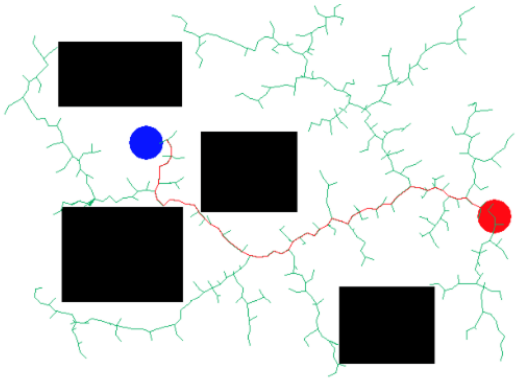


Figura 72. Convergencia del Algoritmo RRT Utilizado en el proyecto

Con la construcción de cada enlace del árbol, el RRT se considera exitoso a menos que el robot haga contacto con un obstáculo o viole alguna restricción dinámica y el estado resultante se agrega al árbol de estados alcanzables. El RRT se ha aplicado con éxito a vehículos con ruedas y patas, así como a autómatas y aviones submarinos.

4.5 Seguimiento de Trayectoria

Como se mencionó en la Sección 4.3.1.3, en el cuadrúpedo tenemos dos colores para ubicar la orientación de SpiderBot, rojo en la parte de atrás y verde en la de adelante, de esto se obtienen dos puntos con coordenadas y en conjunto de un tercer punto que está dado por ruta que el robot debe seguir, se forma un triángulo escaleno, este triángulo se lo puede observar en la Figura 73.

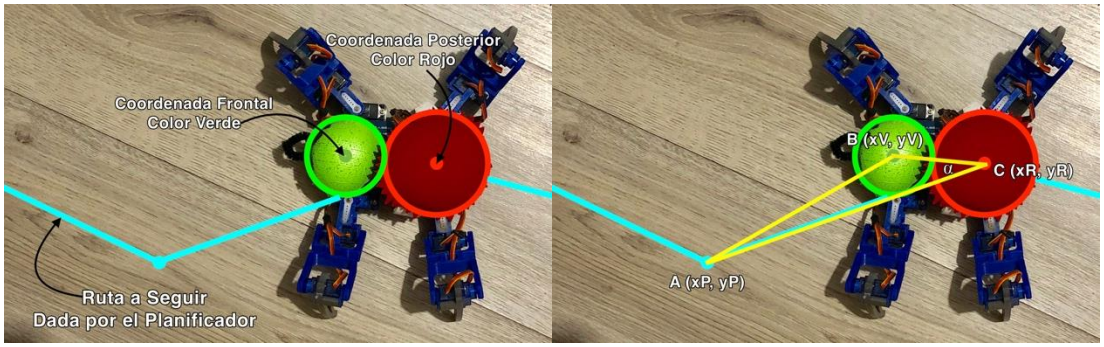


Figura 73. Coordenadas de Puntos y triángulo para realizar el Seguimiento de Ruta

En la Figura 74 se muestra el triángulo formado por las coordenadas mencionadas anteriormente y todas las variables necesarias para poder calcular al ángulo α el cual dirá si el robot está

orientado correctamente hacia la ruta para ir hacia adelante, o si tiene que realizar un giro a la izquierda o derecha para alinearse y continuar.

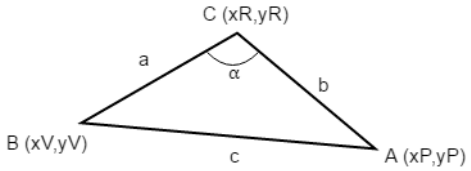


Figura 74. Triangulo escaleno, para determinar el ángulo de rotación del Robot

Para encontrar el ángulo del robot con respecto a la trayectoria, usamos la ley de los cosenos y despejamos el ángulo alfa obteniendo la Ecuación (3).

$$\alpha_1 = cos^{-1}\left(\frac{c^2 + b^2 - a^2}{2 \cdot c \cdot b}\right) \tag{3}$$

Se calcula *a*, *b* y *c* mediante las coordenadas de los puntos *xR*, *yR*, *xV*, *yV*, *xP*, y *yP* mediante las Ecuaciones (4), (5) y (6).

$$a = \sqrt{(xR - xV)^2 + (yR - yR)^2} \tag{4}$$

$$b = \sqrt{(xP - xR)^2 + (yP - yR)^2} \tag{5}$$

$$c = \sqrt{(xP - xV)^2 + (yP - yV)^2} \tag{6}$$

Sólo utilizando la Ecuación (3) no se pueden obtener ángulos que vayan de 0º a 360º, debido a que existe cierta ambigüedad, por ejemplo, si se calcula el valor de $cos^{-1}(0.707)$ nos dá como resultado 45º, pero también puede ser igual a 315º porque con el cos se obtienen los mismos resultados del primer cuadrante, con el cuarto cuadrante y el segundo cuadrante con el tercero en el circulo trigonomético, esto es un problema debido a que si el robot está orientado a 135º y necesita girar hacia la izquierda, pero al usar la fórmula se obtiene el valor de 45º el algoritmo de control le dirá que gire hacia la derecha, siento esta la dirección contraria, causando que el SpiderBot se desvíe.

Para resolver esto también calculamos α con las pendientes de las líneas que forman el ángulo usando la Ecuación (7), aquí también existe un problema similar, con la tan se obtienen los mismos resultados en el primer cuadrante, con el tercer cuadrante y el segundo con el cuarto cuadrante en el circulo trigonomético pero, combinando los dos ángulos se puede obtener precisamente que dirección la araña está enfrentando sin dar lugar a esta ambigüedad.

$$\alpha_2 = \tan^{-1}\left(\frac{m_a - m_{\mathrm{b}}}{1 + m_a \cdot m_{\mathrm{b}}}\right) \tag{7}$$

El cálculo de las pendientes en la Ecuación (7) está dada por las Ecuaciones (8) y (9).

$$m_a = \frac{yP - yR}{xP - xR} \tag{8}$$

$$m_{\mathrm{b}} = \frac{yV - yR}{xV - xR} \tag{9}$$

La lógica para la elección del signo del ángulo α se muestra en la Figura 75.

Una vez calculado este ángulo se puede enviar comandos a la araña para que esta gire hacia la derecha o izquierda si la dirección a la que está enfrentando no es la misma que la Ruta, y si está alineada a ella, enviar comandos hacia adelante para que SpiderBot avance. En la Figura 76 se muestra el Diagrama de Flujo utilizado para realizar el seguimiento de ruta dado por el algoritmo de planificación, en donde θ es un ángulo definido como una constante que va a depender de la configuración del robot, este representa cuantos grados rota el cuadrúpedo cada vez que da un giro a la izquierda o derecha.

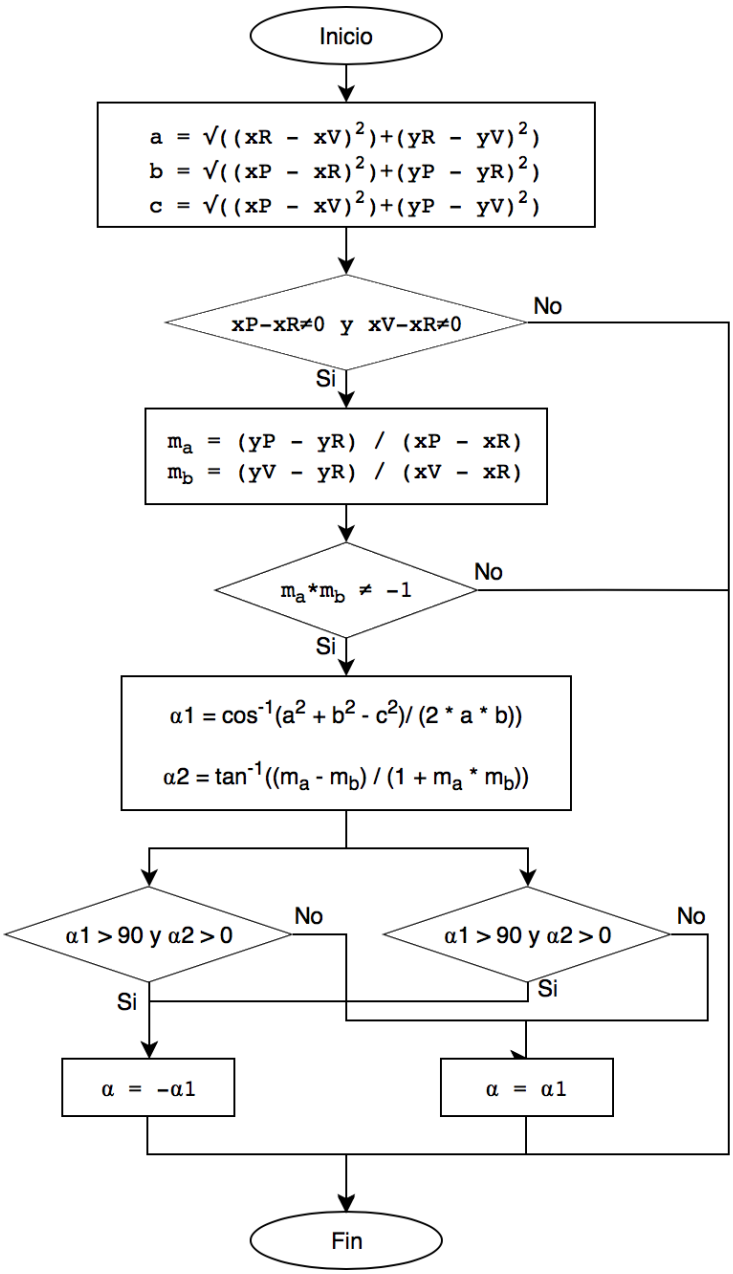


Figura 75. Diagrama de Flujo para cálculo del ángulo α y elección del signo

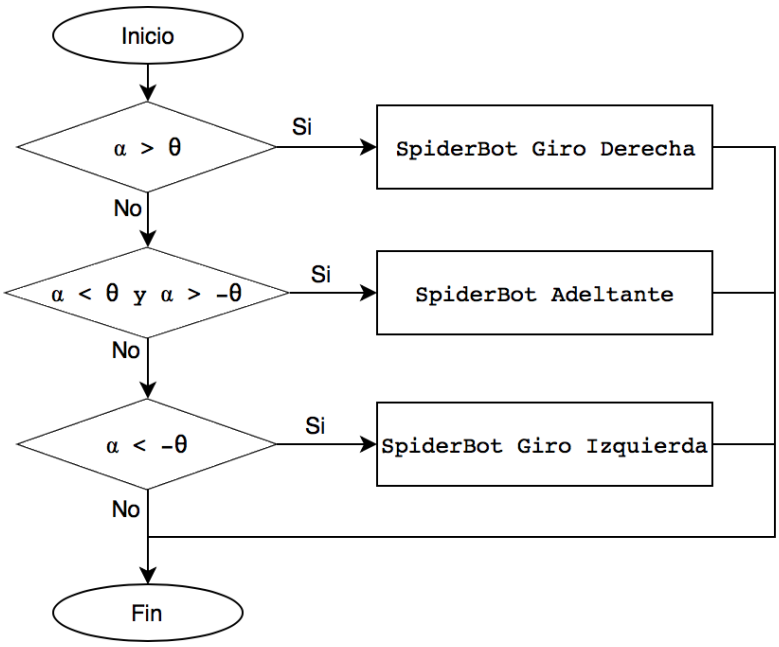


Figura 76. Diagrama de Flujo para Seguimiento de Ruta

CAPITULO V

EVALUACIÓN, PRUEBAS Y ANÁLISIS DE RESULTADOS

En el siguiente capítulo se describe las pruebas y resultados obtenidos, para evaluar el alcance del objetivo general del proyecto de titulación, respecto al desarrollo de un sistema de navegación mediante planificación de trayectoria para un robot cuadrúpedo en entornos no definidos con perspectiva cenital. Con la finalidad de alcanzar los objetivos específicos planteados, se diseña tres casos experimentales realizados en orden, partiendo desde la evaluación de la metodología usada para la detección de los obstáculos y objetos, la comparación del camino trazado por el algoritmo de planificación de trayectorias versus el camino ideal y finalmente la optimización del algoritmo de planificación de ruta.

También se presenta y examina a detalle cada parámetro que define el éxito de la experimentación, explicando el efecto de la principal variable operativa para la optimización del sistema de navegación.

5.1 Diseño del Experimento y Métricas de Evaluación

Realizar el análisis de los datos generados a partir de un experimento es una de las partes más importantes de todo el proyecto. Es adecuado tomarse el tiempo y el esfuerzo necesario para constituir, diseñar y organizar el experimento de forma adecuada para garantizar que los datos obtenidos sean correctos y suficientes, además que serán aprovechables para responder a las preguntas de interés de la forma más clara y eficiente posible. Los experimentos que se van a realizar están de acuerdo con los objetivos del proyecto y se muestran en las siguientes secciones.

5.1.1 Características y Parámetros Generales para los Experimentos

El ordenador utilizado para realizar las tareas de procesamiento de imágenes para la percepción visual del entorno es un computador portátil con las siguientes características.

- Sistema operativo macOS High Sierra
- Pantalla retroiluminada por LED de 13,3 pulgadas; resolución nativa de 2560 por 1600 pixeles.
- Gráficos Intel Iris Graphics 6100
- Procesador Intel Core i5 a 2,7 GHz (Turbo Boost de hasta 3,1 GHz) de doble núcleo.
- 16 GB de memoria LPDDR3 a 1.866 MHz
- Tecnología inalámbrica Bluetooth 4.0
- 100 a 240 V AC a 50 - 60 Hz
- Batería integrada de polímeros de litio de 74,9 vatios/hora
- Versión de Python 2.7.14 con OpenCV 3.4.0.12

Para el sensor visual, es decir la cámara, la resolución elegida es de 1280×720 pixeles, colocada a una altura de 2.5 m y cubrirá un área de 2 x 1.6 m, espacio suficiente para realizar las pruebas con el Robot de dimensiones reducidas.

Los obstáculos serán de color negro para poder tener un buen contraste con la superficie de trabajo, las dimensiones de estos se muestran en la Tabla 20 y son utilizados de acuerdo a cada experimento.

Tabla 20

Tamaño de obstáculos utilizados.

Obstáculo	Ancho (cm)	Largo (cm)	Supercicie (cm^2)
1	14.5	21	304.5
2	5	7	35
3	6	29	174
4	10.5	14.5	152.25
5	4	6	24
6	10	10	100
7	11	4	44
8	10	3	30

5.1.2 Evaluación del algoritmo de Detección de Obstáculos

Una de las bases de la investigación es poder detectar correctamente a los obstáculos que SpiderBot tendrá que evadir, ya que si esto no funciona de manera correcta el robot colisionará contra el obstáculo no localizado, evitando que llegue al punto final o punto objetivo de la planificación de ruta. Para poder evaluar las técnicas de detección de obstáculos y objetos basadas en diferentes técnicas de visión en entornos no definidos con perspectiva cenital se pondrán a prueba los algoritmos de visión creados variando: iluminación, superficie del entorno, posición y tamaño de los obstáculos, creando así diferentes ambientes. La métrica de evaluación para este experimento será el porcentaje de obstáculos detectados calculado con la Ecuación (10), y será realizado con tres niveles de iluminación, tres superficies, seis posiciones distintas y nueve tamaños diferentes de obstáculos, dando en total quince configuraciones distintas.

$$\mathrm{P}(\%) = \frac{\#\ Obstáculos\ Detectados}{\#\ Obstáculos\ Totales} \times 100\% \tag{10}$$

5.1.3 Evaluación del algoritmo de Planificación de Trayectoria

El núcleo del proyecto es desarrollar un sistema de navegación mediante planificación de trayectoria para un robot cuadrúpedo en entornos no definidos con perspectiva cenital, para probar que se ha cumplido con el objetivo principal de la investigación se realizarán varias pruebas. Debido a que normalmente es muy difícil para los experimentadores eliminar ciertas tendencias o parcialidades para beneficio del proyecto utilizando solo su juicio, el uso de la aleatorización en experimentos es una práctica común. En un diseño experimental aleatorizado, los objetos o individuos se asignan aleatoriamente, en este caso a SpiderBot se lo colocará en un punto inicial al azar en la zona de trabajo, e igualmente se pondrá un punto final de llegada generado aleatoriamente por el software. Se evaluará el desempeño del algoritmo comparando la trayectoria encontrada contra la ruta ideal, la cual es una línea recta desde el punto inicial hasta el punto final, evitando

los obstáculos, esta comparación se la realizará obteniendo el error de la raíz cuadrada de la media o también llamada la desviación de la raíz cuadrada media (RMSE) por sus siglas en inglés *Root-Mean Square Error* entre los pixeles en el eje *x* y en *y* con las Ecuaciones (11) y (12), este experimento se repetirá en 10 escenarios distintos.

$$\mathrm{RMSE_x} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ideal} - x_{planificada}\right)^2} \tag{11}$$

$$\mathrm{RMSE_y} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_{ideal} - y_{planificada}\right)^2} \tag{12}$$

Las Ecuaciones (11) y (12) se pueden traducir a un código en Matlab para una mayor facilidad de procesamiento de los datos como se muestra a continuación:

```
RMSEX=sqrt(median((xIdeal-xPlanificada).^2))
RMSEY=sqrt(median((yIdeal-yPlanificada).^2))
```

5.1.4 Evaluación para la Optimización de algoritmo de planificación

El tercer experimento que se va a realizar es probar al algoritmo de planificación con diferentes valores de delta, este valor de delta lo que realiza es modificar la distancia entre cada punto que compone el árbol del RRT, el objetivo es obtener el mejor tiempo para encontrar la ruta de la solución y que exista una distancia aceptable entre cada punto que compone la trayectoria, lo que permitirá que el robot se mueva con facilidad. Con esta variación se hará un análisis de los resultados obtenidos para optimizar el algoritmo de planificación de movimiento.

Para este experimento, se variará la constante delta entre los siguientes valores: delta igual a 40, 30, 25, 20 y finalmente 10, las pruebas se realizaron en iteraciones de 10 veces con cada valor de

delta y se calcula el promedio ($T_{promedio}$) del tiempo que se demora el algoritmo (*tiempo*) en encontrar la trayectoria mediante la Ecuación (13).

$$\mathrm{T}_{\mathrm{promedio}} = \frac{1}{N}\sum_{n=1}^{N} tiempo \tag{13}$$

5.2 Pruebas de funcionamiento y Ejecución de Experimentos.

A continuación, se muestra la ejecución de cada uno de los experimentos diseñados en la Sección 5.1. y en ellos se especifican los elementos clave, identificando el factor a evaluar y los factores evaluados, analizando una relación causa-efecto entre ellos. Una vez recolectada y procesada toda la información, los datos obtenidos se muestran en tablas, además de imágenes de ciertos experimentos que resultaron más relevantes.

5.2.1 Pruebas para la Detección de Obstáculos

La ejecución del experimento se fundamenta en el análisis del algoritmo de detección de obstáculos basado en el atributo visual del objeto, tal como su figura, encontrando la ubicación de cuatro vértices que conforman un rectángulo, y poder identificar si el pixel forma parte de obstáculo o del piso. Para garantizar la efectividad del algoritmo propuesto se asume las siguientes condiciones para el espacio de desplazamiento de SpiderBot: No existen obstáculos suspendidos sobre la superficie, el suelo es relativamente plano y los obstáculos difieren sus características con el suelo. Estas condiciones se cumplen para todos los ambientes de experimentación.

5.2.1.1 Iluminación y Superficie del Espacio de Configuración

Un factor determinante en la detección es el cambio de la iluminación en el entorno que opera SpiderBot. Para la experimentación se utiliza tres niveles de iluminancia, obtenidos de una fuente de luz de 1200 lúmenes. Para un nivel bajo se usa el 10% de la fuente luz, en el nivel medio 40% y por último en el nivel alto se usa al 100% la capacidad de iluminación de la fuente.

El algoritmo de detección basado en la forma del objeto tiene una pequeña desventaja, puede ocasionar inconvenientes en la identificación de la figura al usarlo en suelos con textura, a continuación, se muestra la experimentación realizada en tres tipos de suelos, el primer suelo es de madera, el segundo de cartulina color blanca, y la última superficie de alfombra.

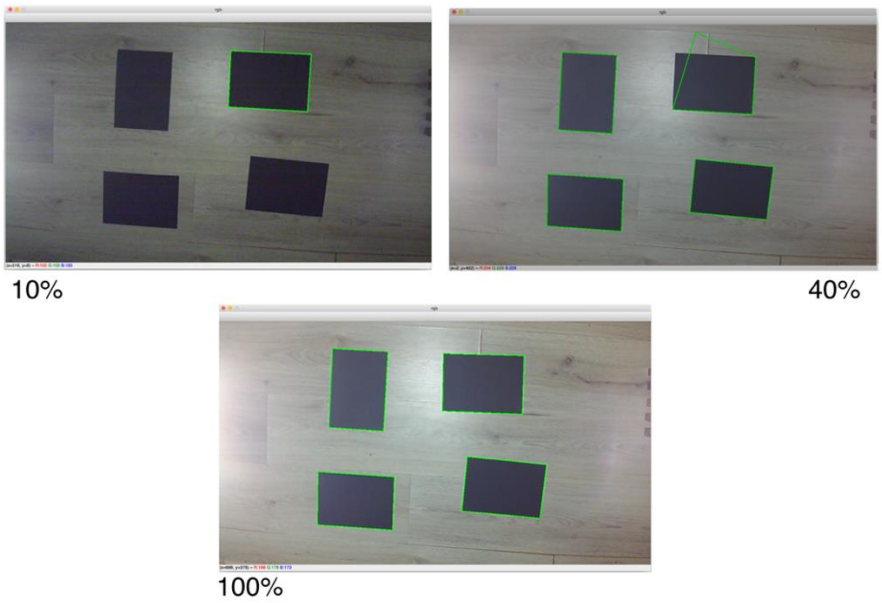


Figura 77. Detección de Obstáculos en una Superficie Madera

En la Figura 77 se muestran las imágenes que la cámara captura para realizar la detección de obstáculos en una superficie de madera, en estas imágenes se puede notar que cuando la iluminación es del 10% sólo se alcanza a localizar un obstáculo, cuando es del 40%, se detectan los cuatro obstáculos pero existe un error en uno de los vértices, y cuando es del 100% se hallaron correctamente todos los obstáculos.

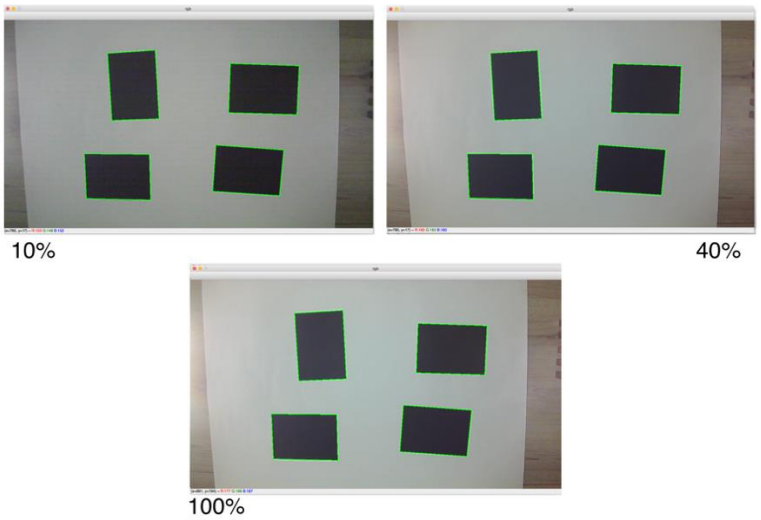


Figura 78. Detección de Obstáculos en una Superficie de Cartulina Color Blanca

La Figura 78 muestra los escenarios con una superficie de cartulina color blanco, en ella las tres imágenes parece que tienen una iluminación parecida, pero esto se debe a que la cámara ajusta automaticamente su enfoque y contraste dependiendo de la iluminación puesta pero en la realidad estos tres escenarios fueron muy distintos uno con otro, aquí en todas las configuraciones se detectan a los cuatro obstáculos sin ningún tipo de fallo.

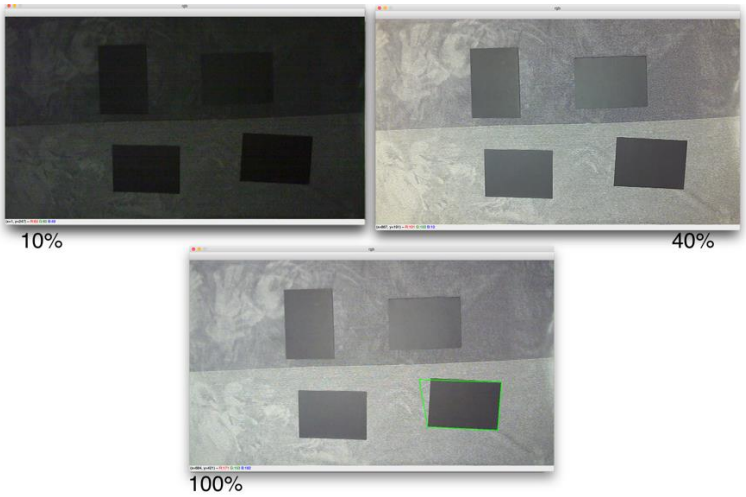


Figura 79. Detección de Obstáculos en una Superficie de Alfombra

Por último, en la Figura 79 se observan los escenarios para cada nivel diferente de iluminación en una superficie de alfombra, la cual, por su color y textura, no ayuda mucho a la detección de

obstáculos, de todas las pruebas solo un obstáculo fue detectado, y esto con un nivel de 100% de iluminación.

Un resumen de los resultados de todas las pruebas y el porcentaje de detección calculado con la Ecuación (10) de cada una de ellas, se muestran en la Tabla 21, en donde se observan que los mejores resultados se obtuvieron con la cartulina blanca de fondo, o cuando la iluminación estuvo por arriba del 40%.

Tabla 21

Escenarios y Resultados de detección de obstáculos para iluminación y superficie.

Superficie	Nivel de Luz	Número de Obstáculos detectados	**P**(%)
Madera	**Alto (100%)**	4	100 %
	Medio (40%)	4	100 %
	Bajo (10%)	1	25 %
Cartulina	**Alto (100%)**	4	100 %
	Medio (40%)	4	100 %
	Bajo (10%)	4	100 %
Alfombra	**Alto (100%)**	1	25 %
	Medio (40%)	0	0 %
	Bajo (10%)	0	0 %

5.2.1.2 Posición y tamaño de los obstáculos

Una vez encontrado el nivel de iluminación y la textura del suelo adecuados, que garanticen un buen índice de detección de obstáculos, procedemos a variar el tamaño y posición de los obstáculos. Aquí se dejará constante la superficie y la iluminación del entorno, específicamente se utilizará la cartulina de color blanco y una iluminación del 100%.

En la Figura 80 se encuentran cuatro configuraciones distintas cambiando la posición de ciertos obstáculos, en ella se logran hallar a todos, sin importar los tamaños que tienen tanto en la medida

longitudinal o en el ancho, existe un caso, en la tercera imagen en donde los vértices de dos rectángulos se unen, y el algoritmo los detecta como un solo polígono.

La Tabla 22 se muestra la recopilación de cada una de las pruebas realizadas, además del porcentaje de detección calculado con la Ecuación (10). Solo existió el caso anteriormente mencionado donde no se encontraron dos obstáculos por separado, si no como un solo obstáculo, aquí se obtuvo un 87.5%. En los demás casos se logró conseguir un 100% de detección, comprobando que el algoritmo de funciona de una manera correcta.

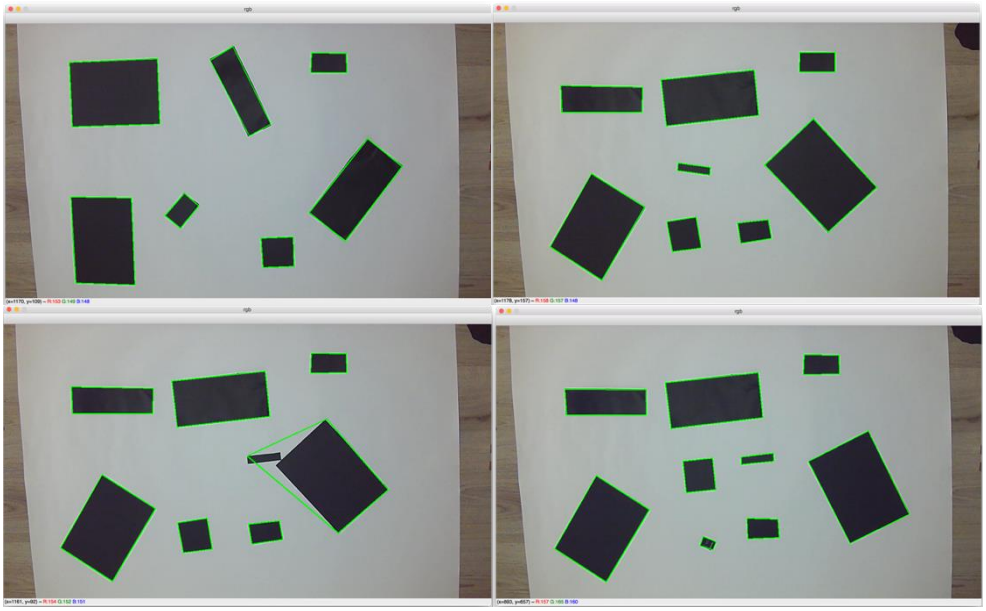


Figura 80. Detección con Diferentes Posiciones y Tamaños de Obstáculos

Tabla 22

Escenarios y Resultados de detección de obstáculos para iluminación y superficie.

Prueba	Número de Obstáculos Detectados	Número de Obstáculos Colocados	**P**(%)
1	7	7	100 %
2	8	8	100 %
3	7	8	87.5%
4	9	9	100 %
5	8	8	100 %
6	8	8	100 %

5.2.2 Pruebas para la Planificación de Trayectoria

En este experimento lo que se realiza es diez diferentes pruebas en las que se coloca al robot y obstáculos en posiciones aleatorias en la superficie de trabajo, luego se obtienen los valores de *x* y *y* de las posiciones de la trayectoria encontrada por el algoritmo y se obtiene el error de la raíz cuadrada de la media (RMSE).

En la Figura 81 se puede observar una de las pruebas realizadas, la línea continua color rojo muestra la ruta encontrada por el algoritmo de planificación de trayectoria y en morado con línea punteada se muestra la ruta ideal que SpiderBot debería seguir para llegar a su objetivo, mostrado por el punto rojo encerrado con un circulo azul. La ruta ideal, y la que tendrá una menor distancia de recorrido, es la que une el punto de inicio con el punto final mediante una línea recta, pero debido a que un obstáculo se encuentra entre estos dos puntos, esta ruta tiene que desviarse para evitarlo.

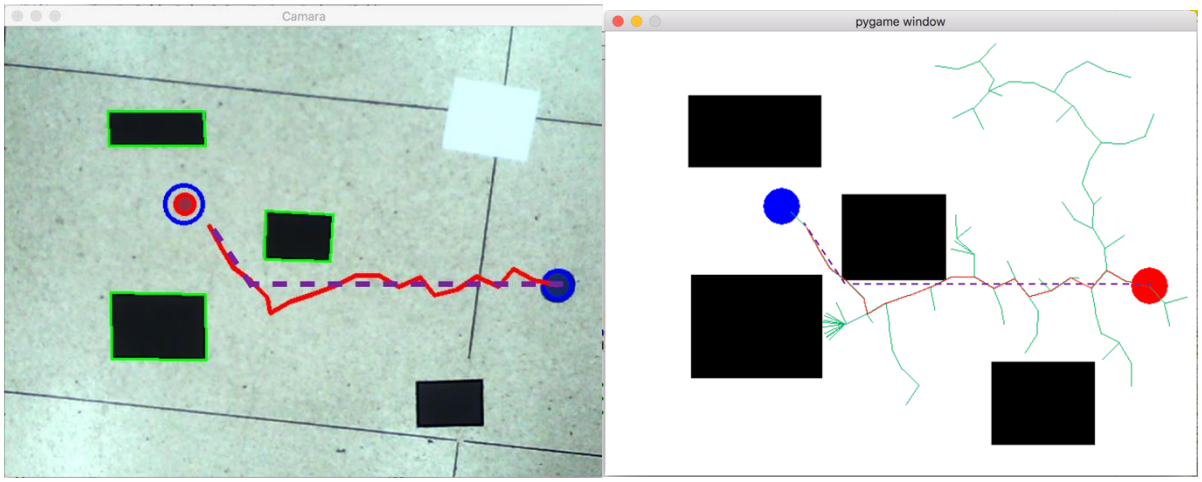


Figura 81. Ruta ideal vs Real

Luego, los datos de cada punto de la ruta son pasados a Matlab y graficados para comprobar que son correctos, obteniendo así la Figura 82. En ella se puede observar en el eje de las abscisas a las coordenadas de pixeles en x y en el eje de las ordenadas a las coordenadas de pixeles en y, pero estas necesitan ser cambiadas de signo para obtener el mismo gráfico reflejado, debido a que

en Python el punto (0,0) se encuentra en la esquina superior izquierda, a diferencia de Matlab que es en la esquina inferior izquierda.

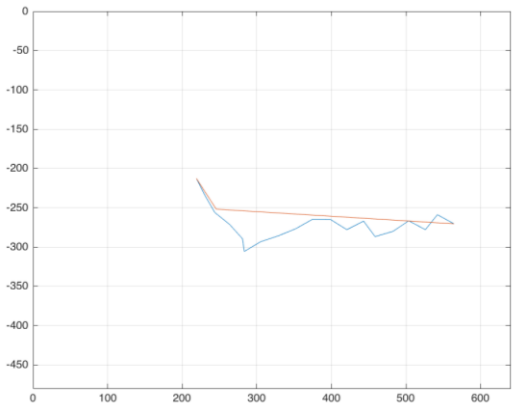


Figura 82. Ruta ideal vs Real en Matlab

En cada prueba se ha realizado un etiquetado dependiendo de que tipo de ruta tuvo que recorrer SpiderBot, con 'tipo de ruta' se refiere a qué eje tuvo un mayor recorrido en cuanto a distancia. En la Figura 83 se pueden observar estos dos estilos, si en el eje x hubo un mayor viaje, será de tipo horizontal, caso contrario, si en el eje y hubo una mayor distancia, será de tipo vertical.

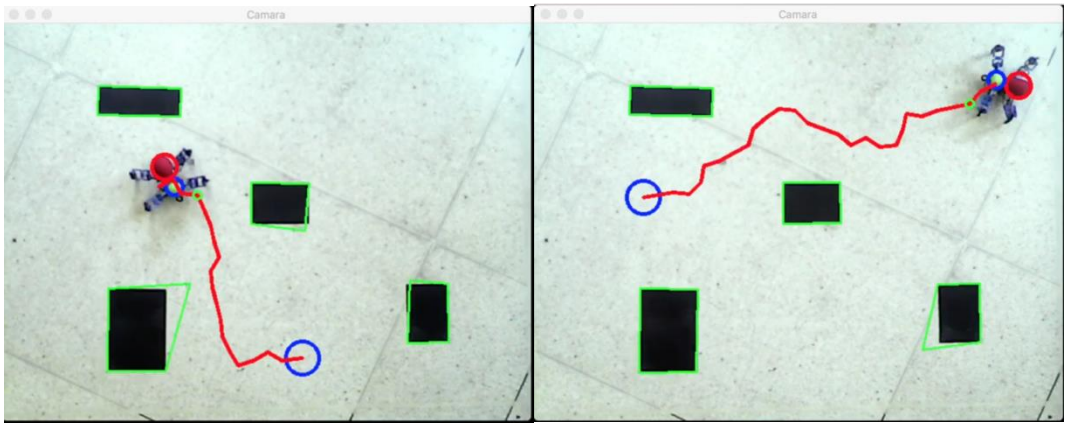


Figura 83. Tipos de Ruta: Vertical (Izquierda) y Horizontal (Derecha)

Los resultados obtenidos para el RMSE en los ejes x y y para cada trayectoria se muestran en la Tabla 23, se puede observar que en el tipo de ruta vertical, el RMSE en el eje x es mayor que el eje y, y en las rutas horizontales sucede lo contrario.

Tabla 23

RMSE Ruta ideal vs Real para diez pruebas

Prueba	RMSE Eje x (píxel)	RMSE Eje y (píxel)	Tipo de Ruta
1	2.2282	10.4122	Vertical
2	5.8883	12.2363	Vertical
3	8.1910	6.4348	Horizontal
4	1.7619	9.6615	Vertical
5	8.2770	2.4803	Horizontal
6	9.5828	4.7942	Horizontal
7	1.5852	12.9407	Vertical
8	12.1690	2.6710	Horizontal
9	4.2813	9.9026	Vertical
10	13.7013	3.7450	Horizontal

5.2.3 Pruebas para la Optimización del algoritmo de planificación

Como se mencionó anteriormente el objetivo de este experimento es obtener el mejor tiempo para encontrar la solución de la trayectoria y se va a realizar la variación de Delta entre los siguientes valores: Delta igual a 40, 30, 25, 20 y finalmente 10, se realizará la prueba con cada valor un total de diez veces y se encontrará el promedio de tiempo. El valor de Delta se lo puede cambiar directamente en el código del programa en Python.

Primero se ejecutará el experimento con Delta igual a 40, en la Figura 84, es posible observar una de las pruebas realizadas, en donde cuatro obstáculos rectangulares de diferente tamaño fueron colocados en el ambiente; un punto inicial y punto final fueron elegidos inicialmente al azar. En la imagen de la izquierda se encuentra lo que la cámara percibe, además de la ruta solución encontrada y en la derecha se pueden ver todas las ramificaciones encontradas del árbol para la planificación de trayectoria.

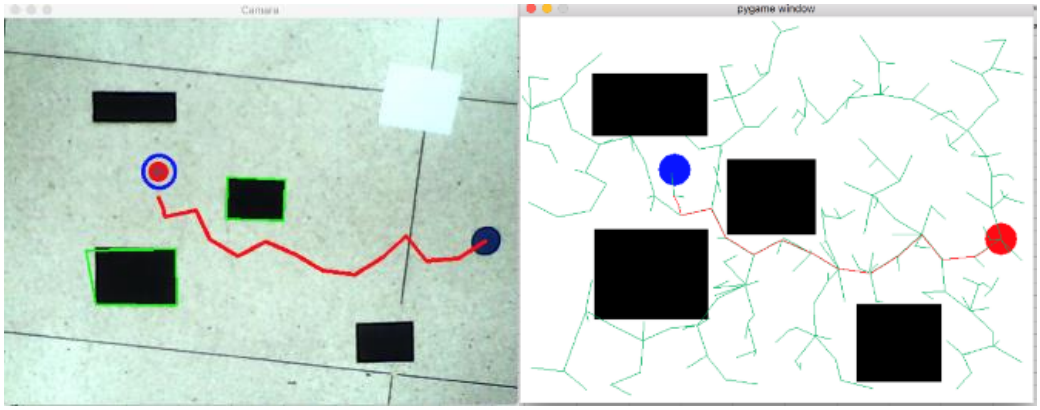


Figura 84. Ejecución del algoritmo de planificación con Delta=40.

En la Tabla 24 se encuentran todos los tiempos de los diez experimentos para este valor de Delta, además se puede ver el tiempo promedio en encontrar la ruta hacia el objetivo calculado con la Ecuación (13) el cual fue de 19.45 segundos.

Tabla 24

Tiempos Experimentales obtenidos con Delta = 40

Delta=40	
N	tiempo (seg)
1	32.95
2	16.67
3	3.75
4	11.71
5	48.47
6	6.65
7	27.03
8	29.79
9	5.48
10	11.95
$\mathbf{T_{promedio}}$	19.45

Cabe recalcar que en este experimento la posición de los obstáculos, punto inicial y punto final u objetivo no fueron cambiados en las cincuenta pruebas realizadas, esto se debe a que solo se desea ver los efectos de la variable delta en la planificación de trayectoria.

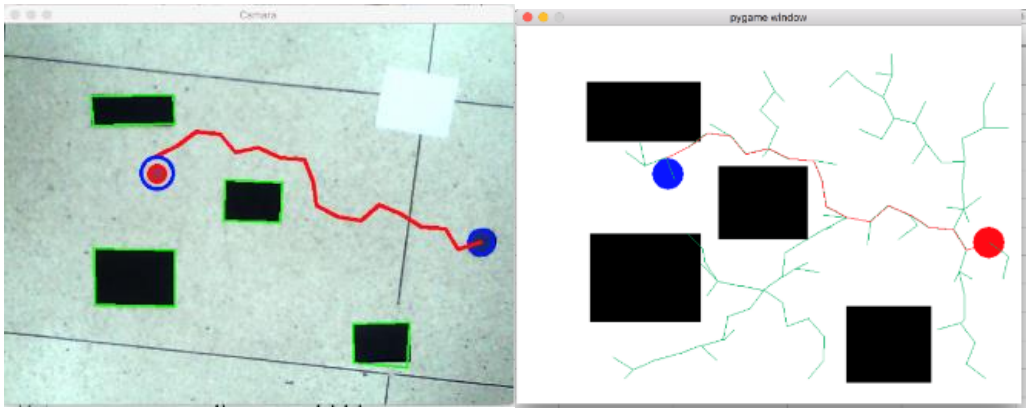


Figura 85. Ejecución del algoritmo de planificación con Delta=30.

A la derecha, en la Figura 85, se encuentra el árbol generado con Delta igual a 30 en color verde, también la trayectoria solución en rojo. Aquí se puede observar que la distacia de nodo a nodo es de un tamaño menor debido a esta variación de Delta y menos caminos fueron formados a comparación de lo que se encuentra en la Figura 84.

La Tabla 25 muestra los datos obtenidos para cada una de las diez pruebas realizadas con Delta igual a 30, aquí el tiempo promedio calculado con la Ecuación (13) fue de 15.25 segundos, es decir un tiempo promedio menor al obtenido en la Tabla 24.

Tabla 25

Tiempos Experimentales obtenidos con Delta = 30

Delta=30	
N	tiempo (seg)
1	35.32
2	15.16
3	7.58
4	**4.53**
5	24.96
6	9.33
7	22.58
8	19.55
9	10.55
10	2.91

La Figura 86 muestra el experimento realizado con Delta igual a 25, aquí se puede observar que el número de nodos obtenidos en el árbol es mucho menor que los anteriores. En todas las pruebas se utilizaron obstáculos de color negro para una mejor detección de estos, ya que así existe un buen contraste entre la superficie de fondo y se logran detectar de una mejor manera las esquinas de los rectángulos.

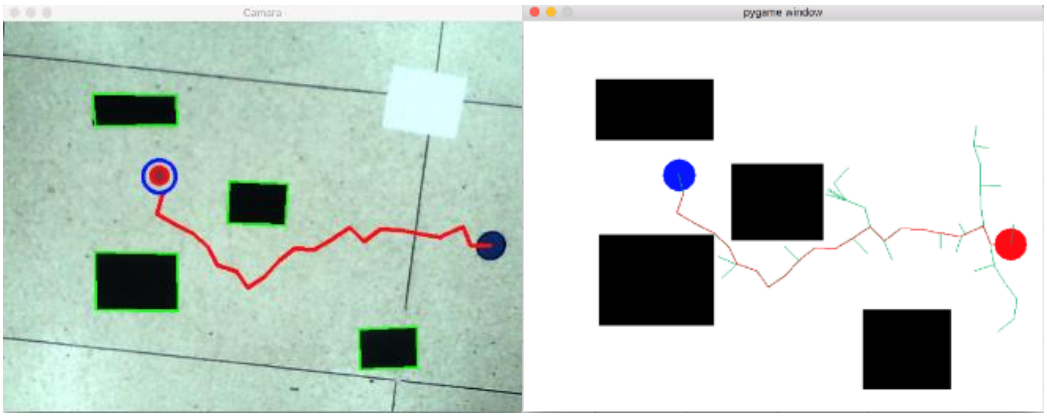


Figura 86. Ejecución del algoritmo de planificación con Delta=25.

Los tiempos experimentales obtenidos con Delta igual a 25 se pueden observar en la Tabla 26, en donde el tiempo promedio para encontrar una trayectoria solución fue de 5.4 segundos, además el menor tiempo en el que el algoritmo encontró una ruta aquí, fue de 2.51 segundos, el cual fue el mejor tiempo hallado entre todos los datos experimentales tomados.

Tabla 26

Tiempos Experimentales obtenidos con Delta = 25

Delta=25	
N	tiempo (seg)
1	7.01
2	2.51
3	7.52
4	4.28
5	3.95
6	3.63

CONTINÚA

7	**4.02**
8	5.18
9	12.06
10	3.87
$T_{promedio}$	5.40

La Figura 87 muestra uno de los experimentos realizados con Delta igual a 20, aquí se pueden observar un número menor de nodos que la Figura 84 y la Figura 85, pero un número mayor que los de la Figura 86. La configuración de obstáculos, punto inicial y punto final son las mismas.

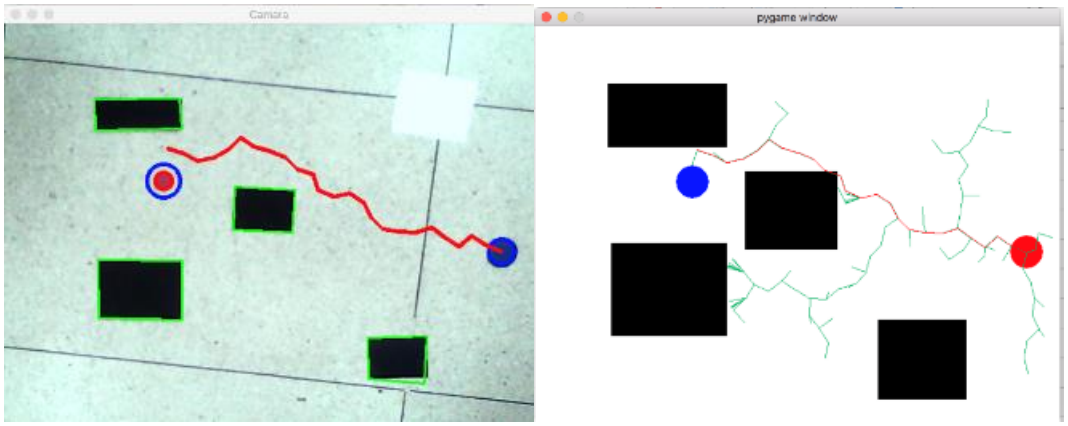


Figura 87. Ejecución del algoritmo de planificación con Delta=20.

Tabla 27 muestra los obtenidos con Delta igual a 20, en esta prueba el tiempo promedio en encontrar una trayectoria que resuelva el problema de planificación fue de 12.74 segundos, siendo este un tiempo mayor que el anterior experimento con Delta igual a 25.

Tabla 27

Tiempos Experimentales obtenidos con Delta = 20

Delta=20	
N	tiempo (seg)
1	11.82
2	13.17
3	18.84
4	15.12
5	5.97

CONTINÚA

6	4.54
7	12.2
8	22.75
9	15.49
10	7.53
$T_{promedio}$	12.74

En la última prueba con Delta igual a 10 la distancia de nodo a nodo es mucho menor que en los anteriores reflejando el cambio del valor, aquí en la Figura 88 se pueden observar todos los nodos creados por el algoritmo del árbol, aquí se necesitó un mayor número de estos para encontrar la solución.

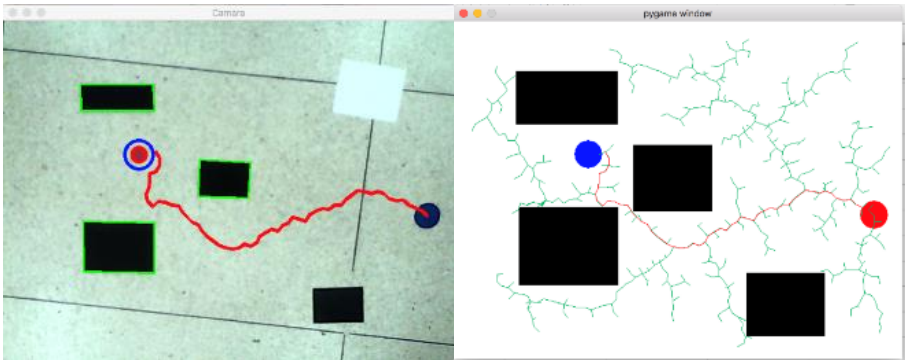


Figura 88. Ejecución del algoritmo de planificación con Delta=10.

En la Tabla 28 se pueden observar todos los tiempos de las diez pruebas realizadas para este valor de Delta, en general se obtuvieron tiempos mayores que los anteriores experimentos, esto se ve evidenciado en el tiempo promedio que se obtuvo, el cual fue de 21.41 segundos siendo este el mayor duración.

Tabla 28

Tiempos Experimentales obtenidos con Delta = 10

Delta=10	
N	tiempo (seg)
1	15.41
2	36

CONTINÚA ⇨

3	**13.54**
4	22.92
5	22.1
6	13
7	13.18
8	56.39
9	11.82
10	9.78
$T_{promedio}$	21.41

La Tabla 29 muestra un resumen de los resultados obtenidos al variar la constante Delta para poder determinar su valor óptimo, el menor tiempo obtenido fue de 5.40 segundos con Delta igual a 25, seguido de 12.74 segundos con Delta igual a 20, como se puede observar el tiempo va decreciendo mientras Delta disminuye, pero cuando se llega a valor de 25 vuelve a aumentar.

Tabla 29

Tiempo Promedio con cada valor de Delta

Delta (píxel)	**$T_{promedio}$ (seg)**
40	19.45
30	15.25
25	5.40
20	12.74
10	21.41

5.3 Análisis de Resultados de los Experimentos

5.3.1 Análisis de Resultados para la Detección de Obstáculos

Posteriormente a la etapa de experimentación, en el cual se pone a prueba el desempeño del algoritmo de detección de obstáculos, usándolo en varios escenarios para evaluar su porcentaje de detección.

Un punto clave en el uso de cámaras para la detección de objetos, es tener una buena fuente de iluminación, ya que requiere el uso de algoritmos de procesamiento de imágenes y que estas

imágenes sean lo suficientemente claras. Para la experimentación se usó los tres niveles de luminosidad, combinados con tres tipos de superficie, como se muestra en la Figura 77, Figura 78 y Figura 79. En una superficie ideal, como lo es el uso de una cartulina color blanco, la fuente de luz no vario su porcentaje de detección, identificando al 100% los obstáculos en cada nivel de luz. En la superficie de madera se notó, que la fuente de luz si influye significativamente en la detección teniendo como recomendación que al trabajar espacios no ideales es conveniente utilizar una gran cantidad de luz, como se observa en la Figura 77 al utilizar la fuente de luz a un 40% de su capacidad, la detección de los obstáculos empieza a mostrar dificultades. Por último, el índice más bajo de detección se obtuvo con la tercera prueba al trabajar sobre una superficie con textura, obteniendo un 25% de detección con la fuente de luz al 100%.

Se determinó que el algoritmo presenta desventajas para reconocer el objeto, al ponerlos sobre un suelo con textura o en los cuales no se pueda diferenciar intensamente sus atributos morfológicos, como se muestra en la Figura 79, a pesar de tener una fuente luminosa fuerte, no se pudo detectar totalmente todos los obstáculos, teniendo un porcentaje de detección de tan solo el 25%, y no siendo suficiente para tener un óptimo desempeño.

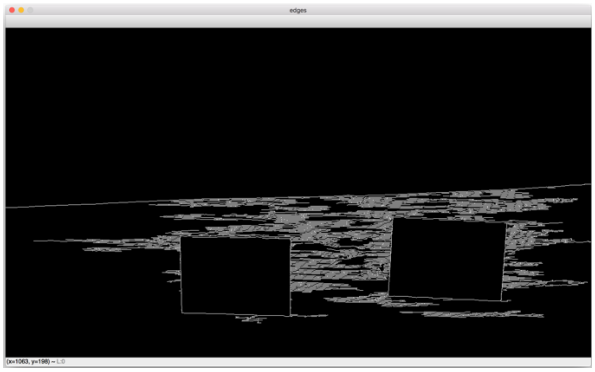


Figura 89. Detección de Bordes, con Superficie de Alfombra

Una de las cosas que se percibió es que el algoritmo no funcionó correctamente con la superficie de alfombra, a pesar de que con el ojo humano se observaban bien los obstáculos, examinando de

una manera más minuciosa, se descubrió que esto se debe al paso de detección de bordes, como se ve en la Figura 89, en los dos rectángulos inferiores el algoritmo divisa ciertas líneas del fondo como bordes e impide que se definan correctamente los vértices para su clasificación como polígonos de cuatro lados. Además, los dos rectángulos superiores se pierden con el fondo debido a su similitud en cuanto a color.

Al variar la posición de los obstáculos en el entorno como se observó en la Figura 80, no se revelaron mayores variaciones, el algoritmo de detección logró su objetivo sin problemas. Existió un caso en el que dos obstáculos se sobreponen entre sí, aquí el algoritmo tomó a estos dos obstáculos como uno solo. Este caso se podría considerar como un error que afectará a la planificación de trayectoria, pero en realidad la ayudará debido a que evitará que se formen nodos en el espacio entre estos dos polígonos, disminuyendo el área de trabajo y logrando así que el algoritmo no pierda tiempo buscando rutas en este espacio.

Utilizando un contraste alto entre los obstáculos y la superficie de trabajo, como lo son obstáculos de color negro con una superficie blanca, se consigue que el algoritmo de detección halle obstáculos de un tamaño relativamente pequeño sin ningún tipo de esfuerzo.

5.3.2 Análisis de Resultados para la Planificación de Trayectoria

Luego de terminada la etapa de ejecución del experimento, en la que se mide la calidad del algoritmo RRT para la planificación de caminos, mediante la raíz del error cuadrático medio o indicador RMSE, ya que evalúa la diferencia entre la trayectoria real y la trayectoria estimada. Nuestra trayectoria real es la que se obtuvo como resultado del algoritmo RRT y la trayectoria estimada, es la menor distancia que une el punto inicial y final.

El RMSE se calculó para cada ruta planificada y en cada eje de desplazamiento, consiguiendo un valor del error en el eje *x* y otro para el eje *y*, como se muestra en la Tabla 23 de la Sección 5.2.2,

mientras el valor RMSE se aproxime más a cero, la trayectoria planificada resultante es altamente deseable, siendo la ruta más corta y que no provoca una colisión.

Se concluye que el valor RMSE aumenta su valor en uno de los ejes, dependiendo si la trayectoria real estimada por el RRT uno dos puntos inicial y final, formando una ruta horizontal o vertical, como se puede ver en la Figura 83. Al ser una trayectoria vertical aumenta el valor RMSE en el eje de las ordenadas y si es una trayectoria horizontal aumenta su valor RMSE en el eje de las abscisas, esto se produce ya que el algoritmo RRT crea ramificaciones aleatorias con varios ángulos de rotación, solo centrado en evitar los obstáculos y no buscando una línea recta para alcanzar el objetivo.

El resultado obtenido con el cálculo de este indicador esta en pixeles y es una medida aceptable para poder decir que el algoritmo planificador de trayectorias encuentra la ruta óptima al no colisionar con algún obstáculo en todos los experimentos realizados.

5.3.3 Análisis de Resultados para la Optimización de algoritmo de planificación

Una vez concluida el proceso de experimentación para la optimización del algoritmo de planificación de trayectoria, se logró establecer como factor determinante a la variable Delta, la cual influye directamente en el tiempo de ejecución para hallar la ruta libre de colisiones, logrando un tiempo relativamente bajo al encontrar la trayectoria que seguirá SpiderBot entre su configuración inicial y la posición objetivo.

La variable Delta define el tamaño de cada ramificación aleatoria del árbol construido con el algoritmo RRT, haciendo que la distancia entre cada nodo no sobre pase el valor de esta constante, estos valores representan la medida en pixeles de cada ramificación.

En la Tabla 30 se muestra el resultado de los 10 experimentos para cada valor de Delta como variable independiente y el tiempo como variable dependiente, mostrando el tiempo más bajo y el valor promedio de ejecución del algoritmo de planificación de trayectoria.

Tabla 30

Resultado de los 10 experimentos para cada valor de tiempo en función de Delta

Tiempo (seg)	Delta=40	Delta=30	Delta=25	Delta=20	Delta=10
Menor	3.75	2.91	2.51	4.54	9.78
Promedio	19.45	15.25	5.40	12.74	9.78

Como se muestra, los tiempos obtenidos para los valores de Delta no se observa gran variación en el tiempo menor, ya que al ser un experimento aleatorio hubo casos en los cuales, si se obtuvo la ruta en tiempos relativamente bajos, pero al analizar la estabilidad del algoritmo tomando en cuenta el tiempo promedio de los experimentos, notamos que al asignar Delta=25, se logra el mejor y menor tiempo.

En la Figura 84, Figura 85, Figura 86, Figura 87 y Figura 88 de la sección anterior se puede observar como aumenta y disminuye el número de ramificaciones en la construcción del árbol aleatorio, al tener una medida muy baja en la distancia de cada rama se crean más nodos, lo cual es más probable que el algoritmo finalice con éxito, pero con mayor procesamiento de datos y más tiempo de inversión. De igual forma se consiguió resultados parecidos al usar una medida alta en la distancia de cada rama, se crean varios nodos y el procesamiento del algoritmo de seguimiento de ruta es mayor, pero tiene gran dificultad al intentar alcanzar el objetivo con un tiempo superior de ejecución.

En base a la Tabla 29 de la sección anterior, se puede observar que el tiempo de ejecución del algoritmo disminuye al encontrar la trayectoria óptima, cuando Delta toma el valor de 25, produciendo menos variación a la salida y obteniendo resultados más estables. También se pudo

observar en la experimentación que el valor de Delta afecta significativamente el tiempo de ejecución, al asignar a Delta valores mayores a 30 y valores menores a 20.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- Para diseño del robot se utilizó piezas simples que facilitaron su manufactura en impresión 3D, sin presentar inconvenientes para desarrollar el modelo matemático de cada articulación, utilizando las matrices de transformación homogénea, se obtuvo la ecuación cinemática de cada extremidad.
- Se cumple el objetivo de modelar matemáticamente cada articulación del robot cuadrúpedo SpiderBot, utilizando el método de Denavit-Hartenberg, para aplicar la cinemática directa, ya que esta propuesta permite obtener una única solución para el valor de las articulaciones, y determinar la posición y orientación del efector final.
- Para seleccionar los componentes que se usaron en el robot cuadrúpedo SpiderBot, se tomó en cuenta, sus dimensiones físicas, costo y consumo eléctrico. Los actuadores principales que se eligieron son: el microcontrolador ATMega328P smd, por su fácil operación y disponibilidad, y los servomotores SG90, principalmente por su tamaño y precisión.
- El diseño y construcción de la placa de control del sistema del Robot, inicialmente fue hecho de forma artesanal, presenciando algunos inconvenientes como su tamaño y sobre todo por no tener una cubierta que proteja los caminos, estos se empezaron a oxidar y dañar las conexiones, por la tanto, se decidió realizar nuevamente el diseño de la placa electrónica optimizando el espacio, al usar componentes más pequeños y una placa a doble lado. Una vez concluido su diseño, la placa fue hecha de forma profesional.
- El robot implementado actualmente está totalmente terminado y funcional, todo su proceso de construcción ha sido realizado con éxito, ya que cada etapa del proyecto ha sido probada por

separado. La comunicación inalámbrica de la transmisión de sus datos también ha sido verificada. El programa para el control de movimiento desarrollado en C++ ha sido terminado, los movimientos que realiza SpiderBot son fluidos y coordinados. Y los algoritmos para su navegación autónoma fueron programados en Python y su funcionalidad fue satisfactoriamente comprobada.

- Es importante conocer la orientación de SpiderBot, por lo tanto, ubicamos dos colores en el cuerpo del Robot, el color verde para reconocer el frente y el color rojo para identificar la parte posterior, con esto dos colores se obtiene dos puntos con las coordenadas cartesianas de su ubicación dentro del espacio de configuración.
- Para poder determinar el ángulo de rotación de SpiderBot respecto a la trayectoria planificada es necesario establecer un triángulo escaleno, dos de sus vértices se determinan por la orientación del robot y el ultimo vértice es establecido por los nodos del RRT, este proceso es iterativo hasta alcanzar la posición final.
- Basados en trigonometría se obtuvo el ángulo de rotación entre el robot y la trayectoria dada por el RRT, utilizando la combinación de dos teoremas, la ley de los cosenos y la pendiente de una recta, era necesario utilizar dos teoremas ya que el primero nos sirve para determinar el valor del ángulo de giro y el segundo para determinar el valor positivo o negativo de dicho ángulo.
- En este proyecto se desarrolló una extensión del algoritmo RRT, ya que proporciona una base fácil para la comprensión y programación del algoritmo mejorado, abordando directamente la planificación de trayectorias con un enfoque novedoso para poder detectar los obstáculos dentro del espacio de trabajo.

- El algoritmo RRT, árbol de exploración rápida usado para la planificación de ruta permite determinar con relativa rapidez el camino entre una configuración inicial y una configuración final, libre de colisión.
- Para la navegación autónoma del robot cuadrúpedo en un entorno no definido, se utiliza la combinación de varios algoritmos, que trabajan en conjunto para definir la trayectoria libre de colisiones que va a seguir SpiderBot. Para esto se utiliza un algoritmo de planificación de ruta RRT y también dos algoritmos de detección de obstáculos, uno usando segmentación morfología del objeto y el otro utilizando su color, el primer algoritmo es específicamente para detectar los obstáculos que debe evitar el robot, y el segundo algoritmo se lo usa para detectar la configuración inicial de SpiderBot en el espacio de trabajo, así el usuario solo debe definir la configuración final o posición objetivo que debe alcanzar.
- Al finalizar las pruebas de funcionamiento del algoritmo de planificación de trayectoria RRT, se comprobó que su tiempo de ejecución para hallar la ruta desde el punto inicial al punto final, depende directamente de la variable “Delta”, aumentando significativamente al utilizar un delta mayor a 30 y menor a 20, siendo 25 el valor más óptimo a utilizar, cumpliendo con los requisitos de tiempo y distancia.
- La ruta óptima no es la que menos distancia recorrió el robot, pero en función de los resultados obtenidos se puede decir que la mejor trayectoria a seguir es la que evita los obstáculos a una distancia pertinente para no tener colisiones por maniobra muy cerca del obstáculo, se determinó que SpiderBot es capaz de escapar sin dificultad de los obstáculos y llegar al objetivo de una forma óptima.

- Con base en los experimentos realizados, la iluminación debe considerarse como un factor determinante para el rendimiento óptimo del software, ya que una cantidad baja de luz afecta el reconocimiento de colores y objetos.
- Utilizando un contraste alto entre los obstáculos y la superficie de trabajo, como lo son obstáculos de color negro con una superficie blanca, se consigue que el algoritmo de detección halle obstáculos de un tamaño relativamente pequeño sin ningún tipo de esfuerzo.

6.2 Recomendaciones

- Debido a que la cámara debe ubicarse en una perspectiva cenital para mayor conveniencia se puede utilizar una cámara con un mayor ángulo de visión, con esto el sensor no se lo tiene que posicionar a una gran altura del suelo.
- Para una mejor detección de los obstáculos y del robot se puede usar una cámara con una mayor resolución, pero siempre tomando en cuenta que el computador donde se realice el análisis, tenga capacidad suficiente de procesamiento para realizar todo el tratamiento de imágenes.
- La tibia del robot se puede imprimir en 3D en un material de tipo flexible como SOFT PLA o FILAFLEX, puesto que esta parte de la pata es la que se encuentra en contacto directo con el suelo, esto ayudará a que SpiderBot no se resbale en suelos lisos y obtenga más tracción.
- Se pueden utilizar Leds a bordo del robot para iluminar interiormente los colores que se encuentran en la carcasa para que consiga una mejor detección de estos, ya que así la cámara divisará no solo la luz reflejada del ambiente que rodea a SpiderBot, sino también la de los Leds.
- Efectuar los experimentos en una superficie libre de imperfecciones o suciedad en vista de que estas pueden hacer que los obstáculos cambien de forma y no sean hallados correctamente por el algoritmo de detección por segmentación morfológica.

- Se recomienda tener varias baterías de repuesto dado que el consumo energético del robot es grande debido a los doce servomotores que posee, con esto, se pueden realizar un mayor número de pruebas sin tener tiempos de espera para recargar las baterías.
- Realizar una estructura metálica o de madera lo suficientemente alta, resistente y desmontable que pueda soportar a la cámara para que el experimento sea movible y no dependa del lugar o techo en donde se monta inicialmente la misma.

6.3 Trabajos Futuros

- En este proyecto el hardware y software desarrollado permiten recopilar datos del entorno y conocer la ubicación y orientación del Robot por lo tanto sirve como base para investigaciones futuras, relacionadas al desarrollo de mecanismos de programación evolutivos, los cuales permitan que aprenda a desplazarse en entorno complejos.
- En el proyecto se puede aumentar la implementación de sensores inerciales (IMU) sobre el robot para estimar la orientación y ubicación de prototipo, creando un sistema más robusto y confiable de acuerdo a la aplicación en la que se use SpiderBot.
- También, como trabajo futuro se podría implementar un controlador PID para seguir mejor el camino al recibir el ángulo de rotación en la araña en lugar de los comandos izquierdo o derecho. Del mismo modo, se puede utilizar el algoritmo RRT en un entorno desconocido utilizando un Kinect montado en una estructura grande y obtener la planificación de las trayectorias.

REFERENCIAS BIBLIOGRÁFICAS

Aguiar, A. P., & Hespanha, J. P. (2004). Logic-based switching control for trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *Proceedings of the American Control Conference*, *4*(8), 3004–3010. https://doi.org/10.1109/ACC.2004.182745

Aguilar, W., Casaliglla, V., & Pólit, J. (2017). Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles. *Electronics*, *6*(1), 10. https://doi.org/10.3390/electronics6010010

Aguilar, W. G., Abad, V., Ruiz, H., Aguilar, J., & Aguilar-Castillo, F. (2017). RRT-Based Path Planning for Virtual Bronchoscopy Simulator (pp. 155–165). Springer, Cham. https://doi.org/10.1007/978-3-319-60928-7_13

Aguilar, W. G., & Angulo, C. (2014a). Real-time video stabilization without phantom movements for micro aerial vehicles. *EURASIP Journal on Image and Video Processing*, *2014*(1), 46. https://doi.org/10.1186/1687-5281-2014-46

Aguilar, W. G., & Angulo, C. (2014b). Robust video stabilization based on motion intention for low-cost micro aerial vehicles. In *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)* (pp. 1–6). IEEE. https://doi.org/10.1109/SSD.2014.6808863

Aguilar, W. G., & Angulo, C. (2016). Real-Time Model-Based Video Stabilization for Microaerial Vehicles. *Neural Processing Letters*, *43*(2), 459–477. https://doi.org/10.1007/s11063-015-9439-0

Aguilar, W. G., Angulo, C., & Costa-Castello, R. (2017). Autonomous navigation control for quadrotors in trajectories tracking. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 10464 LNAI, pp. 287–297). Springer, Cham. https://doi.org/10.1007/978-3-319-65298-6_27

Aguilar, W. G., Angulo, C., & Pardo, J. A. (2017). Motion intention optimization for multirotor robust video stabilization. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)* (pp. 1–4). IEEE. https://doi.org/10.1109/CHILECON.2017.8229689

Aguilar, W. G., Casaliglla, V. P., & Polit, J. L. (2017). Obstacle Avoidance for Low-Cost UAVs. In *2017 IEEE 11th International Conference on Semantic Computing (ICSC)* (pp. 503–508). IEEE. https://doi.org/10.1109/ICSC.2017.96

Aguilar, W. G., Casaliglla, V. P., Pólit, J. L., Abad, V., & Ruiz, H. (2017). Obstacle Avoidance for Flight Safety xwon Unmanned Aerial Vehicles (pp. 575–584). Springer, Cham. https://doi.org/10.1007/978-3-319-59147-6_49

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Parra, H., & Ruiz, H. (2017). Pedestrian Detection for UAVs Using Cascade Classifiers with Meanshift. In *2017 IEEE 11th International Conference on Semantic Computing (ICSC)* (pp. 509–514). IEEE. https://doi.org/10.1109/ICSC.2017.83

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Angulo, C. (2017a). Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps (pp. 563–574). Springer, Cham. https://doi.org/10.1007/978-3-319-59147-6_48

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Lopez, W. (2017b). Cascade Classifiers and Saliency Maps Based People Detection (pp. 501–510). Springer, Cham. https://doi.org/10.1007/978-3-319-60928-7_42

Aguilar, W. G., Luna, M. A., Moya, J. F., Luna, M. P., Abad, V., Ruiz, H., & Parra, H. (2017). Real-Time Detection and Simulation of Abnormal Crowd Behavior (pp. 420–428). Springer, Cham. https://doi.org/10.1007/978-3-319-60928-7_36

Aguilar, W. G., Luna, M. A., Ruiz, H., Moya, J. F., Luna, M. P., Abad, V., & Parra, H. (2017). Statistical Abnormal Crowd Behavior Detection and Simulation for Real-Time Applications (pp. 671–682). Springer, Cham. https://doi.org/10.1007/978-3-319-65292-4_58

Aguilar, W. G., Manosalvas, J. F., Guillén, J. A., & Collaguazo, B. (2018). Robust Motion Estimation Based on Multiple Monocular Camera for Indoor Autonomous Navigation of Micro Aerial Vehicle (pp. 547–561). Springer, Cham. https://doi.org/10.1007/978-3-319-95282-6_39

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017a). RRT* GL Based Optimal Path Planning for Real-Time Navigation of UAVs (pp. 585–595). Springer, Cham. https://doi.org/10.1007/978-3-319-59147-6_50

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017b). RRT* GL Based Path Planning for Virtual Aerial Navigation (pp. 176–184). Springer, Cham. https://doi.org/10.1007/978-3-319-60922-5_13

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing (pp.

596–606). Springer, Cham. https://doi.org/10.1007/978-3-319-59147-6_51

Aguilar, W. G., Salcedo, V. S., Sandoval, D. S., & Cobeña, B. (2017). Developing of a Video-Based Model for UAV Autonomous Navigation (pp. 94–105). Springer, Cham. https://doi.org/10.1007/978-3-319-71011-2_8

Aguilar, W., & Morales, S. (2016). 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms. *Electronics*, *5*(4), 70. https://doi.org/10.3390/electronics5040070

Altafini, C. (2002). Following a path of varying curvature as an output regulation problem. *IEEE Transactions on Automatic Control*, *47*(9), 1551–1556. https://doi.org/10.1109/TAC.2002.802750

Alvarez, P. Q., Antonio, J., Estrada, R., Fernández, A. A., & Torres, J. G. R. (2010). Técnicas para evasión de obstáculos en Robótica Móvil, *1*.

Antipov, V., Kokovkina, V., Kirnos, V., & Priorov, A. (2017). Computer vision system for recognition and detection of color patterns in real-time task of robot control. *2017 Systems of Signal Synchronization, Generating and Processing in Telecommunications, SINKHROINFO 2017*. https://doi.org/10.1109/SINKHROINFO.2017.7997496

Barraquand, J., Kavraki, L., Latombe, J.-C., Motwani, R., Li, T.-Y., & Raghavan, P. (1997). A Random Sampling Scheme for Planning, *16*(The International Journal of Robotic Research), 759–774.

Barraquand, J., Langlois, B., & Latombe, J.-C. (1992). Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(2), 224–241.

Barraquand, J., & Latombe, J.-C. (1993). Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, *10*(2–4), 121. https://doi.org/10.1007/BF01891837

Barrientos Antonio , Peñín Luis F., Balaguer Carlos, A. R. (1997). *Fundamentos De Robotica* (2º). McGraw-Hill.

Bertozzi, M., Broggi, A., & Fascioli, A. (1996). A stereo vision system for real-time automotive obstacle detection. *International Conference on Image Processing*, *1*, 681–684. https://doi.org/10.1109/ICIP.1996.560970

Bohlin, R., & Kavraki, L. E. (2000). Path planning using lazy PRM. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (Vol. 1, pp. 521–528). IEEE.

Borenstein, J., & Koren, Y. (1989). Real-Time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man and Cybernetics*, *19*(5), 1179–1187. https://doi.org/10.1109/21.44033

Borenstein, J., & Koren, Y. (1991). The Vector Field Histogram: Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, *7*(3), 278–288. https://doi.org/10.1109/70.88137

Bradski, G. R. (1998). Computer Vision Face Tracking For Use in a Perceptual User Interface (cité 1923 fois). *Intel Technology Journal*, *2*(2), 12–21. https://doi.org/10.1.1.14.7673

Brooks, R. A., & Lozano-Pérez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(2), 224–233. https://doi.org/10.1109/TSMC.1985.6313352

Cardoso, E., Fernández, A., Marrero, S. A., & Guardado, P. F. (2017). Modelos cinemático y dinámico de un robot de cuatro grados de libertad. *Ingeniería Electrónica, Automática y Comunicaciones*, *38*(3), 56–75.

Chinnaiah, M. C., Dubey, S., Vineela, L., Bindu, K., & Babu, E. B. (2016). An unveiling path planning algorithm with minimal sensing using embedded based robots. *2016 International Conference on Advances in Human Machine Interaction, HMI 2016*, 48–52. https://doi.org/10.1109/HMI.2016.7449171

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603–619. https://doi.org/10.1109/34.1000236

Cortes, J., Siméon, T., & Laumond, J.-P. (2002). A random loop generator for planning the motions of closed kinematic chains using PRM methods. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on* (Vol. 2, pp. 2141–2146). IEEE.

Crabbe, F. L. (2001). Unifying artificial intelligence robotics: An undergraduate textbook. *Adaptive Behavior*, *9*(2), 119–121.

de Wit, C. C. (1998). Trends in mobile robot and vehicle control. *Control Problems in Robotics and Automation*, 151–175. https://doi.org/10.1007/BFb0015082

de Wit, C. C., Siciliano, B., & Bastin, G. (1997). *Theory of Robot Control (Communications and Control Engineering)*. Retrieved from http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/3540760547

Donald, B. R. (1987). A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, *31*(3), 295–353. https://doi.org/10.1016/0004-3702(87)90069-5

Dong, Y. (2015). What's the difference between RRT and RRT* and which one should we use. Retrieved from https://www.youtube.com/watch?v=JeEk_CWcRFI

Druzhkov, P. N., Erukhimov, V. L., Zolotykh, N. Y., Kozinov, E. A., Kustikova, V. D., Meerov, I. B., & Polovinkin, A. N. (2011). New object detection features in the OpenCV library. *Pattern Recognition and Image Analysis*, *21*(3), 384–386. https://doi.org/10.1134/S1054661811020271

EECI Institute. (2011). *Trajectory tracking & Path-following control*. *EECI Graduate School on Control Supélec*. Retrieved from http://www.eeci-institute.eu/GSC2011/Photos-EECI/EECI-GSC-2011-M6/EECI-GSC-2011-M6/06_TrackPFcontrol.pdf

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, *59*(2), 167–181. https://doi.org/10.1023/B:VISI.0000022288.19776.77

Fernández, J. F., & Barrientos, T. A. (2016). Análisis , desarrollo y evaluación de modos de marcha para un robot hexápodo.

Firmanda, D., & Pramadihanto, D. (2014). Computer Vision Based Analysis for Cursor Control Using Object Tracking and Color Detection. *2014 Seventh International Symposium on Computational Intelligence and Design*, *2*, 525–528. https://doi.org/10.1109/ISCID.2014.277

Garcia, E., Estremera, J., & de Santos, P. G. (2003). A control architecture for humanitarian-demining legged robots. In *Proc. Int. Conf. Climbing and Walking Robots* (pp. 383–390).

Gouzènes, L. (1984). Strategies for Solving Collision-free Trajectories Problems for Mobile and Manipulator Robots. *The International Journal of Robotics Research*, *3*(4), 51–65. https://doi.org/10.1177/027836498400300403

Goyal, K., Kartikey, A., & Kumar, R. (2011). Face detection and tracking. *International Conference on Electronics, Communication and Aerospace Technology*, *2*(May), 1–17. https://doi.org/10.1109/ICECA.2017.8203730

Grupo Aurova. (2016). Cinemática Inversa. Retrieved January 18, 2018, from http://www.aurova.ua.es/robolab/EJS2/RRR_Intro_3.html

Guennouni, S., Ahaitouf, A., & Mansouri, A. (2015). Multiple object detection using OpenCV on an embedded platform. *Colloquium in Information Science and Technology, CIST*, *2015*–

Janua(January), 374–377. https://doi.org/10.1109/CIST.2014.7016649

Hauser, K., Bretl, T., Latombe, J. C., Harada, K., & Wilcox, B. (2008). Motion planning for legged robots on varied terrain. *International Journal of Robotics Research*, *27*(11–12), 1325–1349. https://doi.org/10.1177/0278364908098447

Hirose, S., & Kato, K. (1998). Development of quadruped walking robot with the mission of mine detection and removal-proposal of shape-feedback master-slave arm. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on* (Vol. 2, pp. 1713–1718). IEEE.

Iagnemma, K., & Dubowsky, S. (2000). Mobile robot rough-terrain control (RTC) for planetary exploration. In *Proceedings of the 26th ASME Biennial Mechanisms and Robotics Conference, DETC* (Vol. 2000).

Iida, F., Dravid, R., & Paul, C. (2002). Design and control of a pendulum driven hopping robot. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on* (Vol. 3, pp. 2141–2146). IEEE.

Kala, R., Shukla, A., & Tiwari, R. (2010). Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning. *Artificial Intelligence Review*, *33*(4), 307–327. https://doi.org/10.1007/s10462-010-9157-y

Karaman, S., & Frazzoli, E. (2010). Optimal kinodynamic motion planning using incremental sampling-based methods. *Proceedings of the IEEE Conference on Decision and Control*, 7681–7687. https://doi.org/10.1109/CDC.2010.5717430

Katic, D., & Vukobratovic, M. (2002). Intelligent soft-computing paradigms for humanoid robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on* (Vol. 3, pp. 2533–2538). IEEE.

Kavraki, L. E., Latombe, J.-C., & kolountzakis, M. (1998). Analysis of Probablistic Roadmaps for Path Planing (Vol. 14, pp. 166–171). IEEE.

Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces, 566–580. https://doi.org/10.1109/70.508439

Khatib, O. (1986). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, *5*(1), 90–98. https://doi.org/10.1177/027836498600500106

Koditschek, D. (1989). Robot Planning and Control Via Potential Functions. *The Robotics Review*, 349–367.

Krishna, A., Nandanan, K., Kumar, S. P., Srihari, K. S., & Sivraj, P. (2014). Design and fabrication of a hexapod robot. In *Embedded Systems (ICES), 2014 International Conference on* (pp. 225–230). IEEE.

Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (Vol. 2, pp. 995–1001). IEEE.

LaValle, S. M. (1998). Rapidly-Exploring Random Trees: A New Tool for Path Planning. *In*, *129*, 98–11. https://doi.org/10.1.1.35.1853

Lozano-Perez, T. (1987). A simple motion-planning algorithm for general robot manipulators. *IEEE Journal on Robotics and Automation*, *3*(3), 224–238. https://doi.org/10.1109/JRA.1987.1087095

Madrigal, R. I., & Idiarte, E. V. (2002). *Robots industriales manipuladores*. Univ. Politèc. de Catalunya.

Matthies, L., Maimone, M., Johnson, A., Cheng, Y., Willson, R., Villalpando, C., … Angelova, A. (2007). Computer vision on Mars. *International Journal of Computer Vision*, *75*(1), 67–92. https://doi.org/10.1007/s11263-007-0046-z

Menéndez, C. (2012). Navegación de robots autónomos en entronos dinámicos.

Michalko, M., Onuška, J., Lavrin, A., Cymbalák, D., & Kainz, O. (2016). Tracking the object features in video based on OpenCV. *ICETA 2016 - 14th IEEE International Conference on Emerging ELearning Technologies and Applications, Proceedings*, *1*, 223–226. https://doi.org/10.1109/ICETA.2016.7802081

Mills-Tettey, A., Lee-Shue, V. J., Prasad, N., & Tantisevi, K. (n.d.). *Robotic Motion Planning: A* and D* Search*.

Moeslund, T. B., Thomas, G., Hilton, A., Carr, P., & Essa, I. (2017). Computer Vision in Sports. *Computer Vision and Image Understanding*, *159*, 1–2. https://doi.org/10.1016/j.cviu.2017.05.006

Murphy, R. (2000). *Introduction to AI robotics*. Cambridge, Mass: MIT Press.

ó' Dúnlaing, C., Sharir, M., & Yap, C. K. (1983). Retraction: A New Approach to Motion-planning. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (pp. 207–

220). New York, NY, USA: ACM. https://doi.org/10.1145/800061.808750

OpenCV Team. (2018). About - OpenCV library. Retrieved January 20, 2018, from https://opencv.org/about.html

Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., León, G., & Jara-Olmedo, A. (2017). Math Model of UAV Multi Rotor Prototype with Fixed Wing Aerodynamic Structure for a Flight Simulator (pp. 199–211). Springer, Cham. https://doi.org/10.1007/978-3-319-60922-5_15

Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., Reyes, R. P., & Montoya, L. (2017). Vertical take off and landing with fixed rotor. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)* (pp. 1–6). IEEE. https://doi.org/10.1109/CHILECON.2017.8229691

Ortiz, O., & Santana, A. (2017). *Planificación de caminos para Robots en Realidad Virtual*.

Pantofaru, C., & Hebert, M. (2005). A Comparison of Image Segmentation Algorithms. *Carnegie Mellon University*, 336. Retrieved from http://repository.cmu.edu/cgi/viewcontent.cgi?article=1335&context=robotics%5Cnpapers2://publication/uuid/219D0A8C-2FB1-4CB4-AC9B-7E612F2CEA55

Parker, J. R. (Jim R. ., & Terzidis, K. (2011). *Algorithms for image processing and computer vision* (2nd ed.). Wiley Pub. Retrieved from https://books.google.es/books?hl=es&lr=&id=BK3oXzpxC44C&oi=fnd&pg=PR1&dq=computer+vision&ots=ISiJapxCMM&sig=R9lIR_q_7NEC9i1nNvqsmGJNbNc#v=onepage&q&f=false

Plustech.fi. (n.d.).

Rodríguez, S., Tang, X., Lien, J. M., & Amato, N. M. (2006). An obstacle-based rapidly-exploring random tree. *Proceedings - IEEE International Conference on Robotics and Automation*, *2006*(May), 895–900. https://doi.org/10.1109/ROBOT.2006.1641823

Saha, M. (2006, August). *Motion Planning with Probabilistic roadmaps*. Stanford.

Sampei, M., Tamura, T., Itoh, T., & Nakamichi, M. (1991). Path tracking control of trailer-like mobile robot. *Proceedings IROS 91IEEERSJ International Workshop on Intelligent Robots and Systems 91*, (91), 193–198. https://doi.org/10.1109/IROS.1991.174448

Samson, C. (1992). Path Following And Time-Varying Feedback Stabilization of a Wheeled Mobile Robot. *Second International Conference on Automation, Robotics and Computer Vision*, 1–14.

Samson, C. (1995). Control of Chained Systems Application to Path Following and Time-Varying Point-Stabilization of Mobile Robots. *IEEE Transactions on Automatic Control*, *40*(1), 64–77. https://doi.org/10.1109/9.362899

Schwartz, J. T., & Sharir, M. (1983). On the "piano movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, *4*(3), 298–351. https://doi.org/10.1016/0196-8858(83)90014-3

Sharma, T., Kumar, S., Yadav, N., Sharma, K., & Bhardwaj, P. (2014). Air-Swipe Gesture Recognition Using OpenCV in Android Devices. https://doi.org/10.1109/ICAMMAET.2017.8186632

Shimoda, S., Wingert, A., Takahashi, K., Kubota, T., & Nakatani, I. (2004). Hopping direction controllability for small body exploration robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (Vol. 3, pp. 2987–2992). IEEE.

Siegwart, R., & Nourbakhsh, I. R. (2004). *Introduction to autonomous mobile robots*. Cambridge, Mass: MIT Press.

Spong MW, Hutchinson S, V. M. (2004). *Robot dynamics and control.* (2º).

Sunada, W. H., & Dubowsky, S. (1983). On the dynamic analysis and behavior of industrial robotic manipulators with elastic members. *Journal of Mechanisms, Transmissions, and Automation in Design*, *105*(1), 42–51.

Szeliski, R. (2011). *Computer vision : algorithms and applications*. Springer.

Szeliski, R., & Stockman, G. C. (2011). *Computer Vision*. *Computer Vision* (Vol. 5). London: Springer London. https://doi.org/10.1007/978-1-84882-935-0

Takita, K., Hodoshima, R., & Hirose, S. (2000). Fundamental mechanism of dinosaur-like robot TITRUS-II utilizing coupled drive. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on* (Vol. 3, pp. 1670–1675). IEEE.

Tenreiro Machado, J., & Silva, M. (2006). *An Overview of Legged Robots*.

Weichselbaum, J., Zinner, C., Gebauer, O., & Pree, W. (2013). Accurate 3D-vision-based obstacle detection for an autonomous train. *Computers in Industry*, *64*(9), 1209–1220. https://doi.org/10.1016/j.compind.2013.03.015

Williamson, T. (1997). Obstacle Detection and Pedestrian Recognition Using A 3DPMD Camera. *Computer Vision: A Reference Guide*, 307–313. https://doi.org/10.1109/IVS.2006.1689632

Yamaguchi, J., Soga, E., Inoue, S., & Takanishi, A. (1999). Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* (Vol. 1, pp. 368–374). IEEE.

Yin, J., & Yang, X. (2016). 3D Facial Reconstruction of Based on Opencv and Directx. *ICALIP*, 341–344. https://doi.org/10.1109/ICALIP.2016.7846562

Zhou, Y., & Dong, Z. (2017). A vision-based autonomous detection scheme for obstacles on the runway. In *2017 Chinese Automation Congress (CAC)* (pp. 832–838). IEEE. https://doi.org/10.1109/CAC.2017.8242881

Zielinska, T., & Heng, J. (2003). Mechanical design of multifunctional quadruped. *Mechanism and Machine Theory*, *38*(5), 463–478. https://doi.org/10.1016/S0094-114X(03)00004-1