UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

MAESTRÍA EN INFORMÁTICA

TESIS

ALGORITMOS DE SIMULACIÓN DE LA TRAYECTORIA ÓPTIMA DE ROBOTS REALIZADO EN LOS LABORATORIOS DE INGENIERÍA ELECTRÓNICA DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO, 2017

PRESENTADA POR:

JESUS VIDAL LOPEZ FLORES

PARA OPTAR EL GRADO ACADÉMICO DE:

MAGÍSTER SCIENTIAE EN INFORMÁTICA

MENCIÓN EN GERENCIA DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES

PUNO, PERÚ

2018

DEDICATORIA

A Con infinita gratitud a mis queridos padres Gerardo López y la Ruth Flores, por todo el cariño, apoyo, comprensión, motivación y enseñanzas que me dieron. Ejemplos de dignidad y sacrificio, que supusieron guiarme por la senda del bien con su apoyo incondicional en el logro de mi profesión.

A mis colegas de la E.P. de Ing. Electrónica, Ingenieros: José, Luis, Ferdinand y Eudes, por haber compartido conmigo grandes momentos y por su apoyo incondicional, que sin ellos no sería posible este proyecto de investigación.

A mi Familia mis hijos Dylan y Jahcob con quienes comparto, los mejores momentos de mi vida y siempre los tengo presente.

AGRADECIMIENTOS

A la Universidad Nacional del Altiplano - Puno, ente formador de profesionales al servicio de la sociedad.

Al Ing. José Emmanuel Cruz de la Cruz, Asesor del presente proyecto de investigación, por las acertadas contribuciones, consejos e ideas que me proporcionaron, a quienes agradezco su sincera amistad.

A los miembros de mi jurado, por sus valiosas contribuciones y observaciones, las cuales permitieron mejorar el trabajo y mi agradecimiento más sincero por valorar mi trabajo.

Finalmente, mi reconocimiento general a todas aquellas personas que de una u otra forma cooperaron en la realización de esta tesis.

ÍNDICE GENERAL

CAPÍTULO III

MATERIALES Y MÉTODOS

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

Puno, 13 de abril de 2018

ÁREA: Diseño y Evaluación de algoritmos evolutivos.
TEMA: Algoritmos de Simulación.

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ANEXOS

Pág.

RESUMEN

En la presente tesis se buscó y se simuló los algoritmos que respondan a una mejor respuesta en la evasión de obstáculos para el seguimiento de un punto fijo de llegada y así lograr cumplir con el objetivo final, determinar algoritmos de simulación para una mejor trayectoria de la locomoción de un robot en los laboratorios de Electrónica, para ello se trabajara con los recursos de memoria y tiempo en función a la dimensión del tamaño de nuestro espacio a trabajar, para ello se está optando por dos tipos de algoritmos: Walk Random y Tree Guarding Walk; los cuales ayudará a analizar cuál de estos dos tipos de algoritmos nos puede cumplir con la optimización de nuestro problema de evasión de obstáculos. Lo primero a realizar fue, la programación de estos algoritmos y ver los tiempos de respuesta que demorará en llegar al objetivo, cuya trayectoria incluirá algunos obstáculos a evadir, la dimensión del recorrido estará incluido para obtener los tiempos de respuesta de cada uno de los algoritmos. Se ha utilizado en el proceso de la investigación una metodología OpenUP. Según Pressman (2005) la metodología Open Unified Process, o también conocida en castellano como Proceso Unificado, como instrumento se utilizó el lenguaje de programación PHYTON; se concluyó: El marco de integración de algoritmos de seguimiento y planificación de trayectorias propuesto, muestra bajos errores en el seguimiento de las trayectorias, debido a la optimización de los parámetros del controlador, para cada trayectoria. Es decir, el controlador, genera la mejor solución para cada trayectoria. Por otro lado, los parámetros del controlador obtenidos mediante algoritmos garantizan una solución casi óptima, ya que evita caer en mínimos locales a diferencia de otros métodos de optimización. El ampliamente probado algoritmo de planificación modelo Walk Random y Tree Guarding Walk ha probado su eficacia en la generación de trayectorias tanto locales como globales y su eficiencia en términos de tiempo de cómputo. Lo que hace adecuado para la trayectoria de un robot. La evasión local de obstáculos detectados por el sensor Kinect, resulta adecuado para la detección de objetos en el camino del robot. Pero se debe tener en cuenta en su implementación en terreno, el efecto de la luz sobre el sensor de profundidad y los efectos de la vibración.

Palabras clave: Algoritmos, locomoción, robot.

ABSTRACT

In the research developed and simulated the algorithms that respond to a better response in the evasion of obstacles to follow a fixed point of arrival and thus achieve the final objective, determine algorithms for a better trajectory of the locomotion of a robot in electronics laboratories, for this we will work with the resources of memory and time depending on the size dimension of our space to work, for this we are opting for two types of algorithms: Walk Random and Tree Guarding Walk; which will help to analyze which of these two types of algorithms can meet us with the optimization of our problem of obstacle avoidance. The first thing to do was, the programming of these algorithms and see the response times that will take to reach the objective, whose trajectory will include some obstacles to evade, the dimension of the travel space will be included to obtain the response times of each one of them. the algorithms. An OpenUP methodology has been used in the research process. According to Pressman (2005) the methodology Open Unified Process, or also known in Castilian like Unified Process, like instrument the programming language was used Phyton; it was concluded: The proposed trajectory tracking and planning algorithm integration framework, shows low errors in trajectory tracking, due to the optimization of the driver parameters, for each trajectory. In other words, the controller generates the best solution for each path. On the other hand, the parameters of the controller obtained through algorithms guarantee an almost optimal solution, since it avoids falling into local minimums unlike other optimization methods. The extensively tested model planning algorithm Walk Random and Tree Guarding Walk has proven its effectiveness in generating both local and global trajectories and its efficiency in terms of computation time. What makes it suitable for the trajectory of a robot? The local evasion of obstacles detected by the Kinect sensor is suitable for detecting objects in the path of the robot, even if they are small. But it should be taken into account in its implementation in the field, the effect of light on the depth sensor and the effects of vibration in the determination of the heights of the objects detected.

Keywords: Algorithms, locomotion, robot.

INTRODUCCIÓN

En la presente tesis se buscó y se simuló los algoritmos que respondan a una mejor respuesta en la evasión de obstáculos para el seguimiento de un punto fijo de llegada y así lograr cumplir con el objetivo final, determinar algoritmos de simulación para una mejor trayectoria de la locomoción de un robot en los laboratorios de Electrónica, para ello se trabajó con los recursos de memoria y tiempo en función a la dimensión del tamaño de nuestro espacio a trabajar, se opta por dos tipos de algoritmos: Walk Random y Tree Guarding Walk; los cuales ayudó a analizar cuál de estos dos tipos de algoritmos nos puede cumplir con la optimización de nuestro problema de evasión de obstáculos. Se utilizó en la investigación una metodología Open Unified Process, o Proceso Unificado, como instrumento se utilizó el lenguaje de programación PHYTON; se concluyó: El marco de integración de algoritmos de seguimiento y planificación de trayectorias propuesto, muestra bajos errores en el seguimiento de las trayectorias, debido a la optimización de los parámetros del controlador, para cada trayectoria; el controlador, genera la mejor solución para cada trayectoria; el ampliamente probado algoritmo de planificación modelo Walk Random y Tree Guarding Walk ha probado su eficacia en la generación de trayectorias tanto locales como globales y su eficiencia en términos de tiempo de cómputo. Lo que hace adecuado para la trayectoria de un robot.

El presente estudio consta de cuatro capítulos que se presenta a continuación:

Capítulo I: Revisión de literatura; en él se presenta el marco teórico, en el que se presenta todo el acervo teórico que corresponde a las variables motivo de estudio, los antecedentes de investigación que sirve de orientación y guía de investigación.

Capítulo II: Planteamiento del problema; consta de la identificación del problema, que el investigador asume como reto para la solución del problema; por otra parte, el enunciado del problema, son preguntas de investigación; la justificación, son las razones por el cual el investigador hace referencia el motivo de investigación; Los objetivos de investigación, son los que orientan el proceso de investigación hacer cumplidas; las hipótesis de investigación, son supuestos de la tesis a comprobar y ser demostrados.

Capítulo III: Materiales y métodos; están compuestos por el lugar de estudio, la población, muestra de estudio, el método de investigación y la descripción detallada de métodos por objetivos específicos.

Capítulo IV: Resultados y discusión; en este apartado se llega a la conclusión del estudio, a las recomendaciones pertinentes, bibliografía y anexos.

CAPÍTULO I

REVISIÓN DE LITERATURA

2.1. Marco teórico

1.1.1. Algoritmo

En su libro Fundamentos de programación, Luis Joyanes Aguilar, define al algoritmo como un método para resolver un problema. Aunque la popularización del término ha llegado con el advenimiento de la era informática, algoritmo proviene de Mohammed al-KhoWârizmi, matemático persa que vivió durante el siglo IX y alcanzó gran reputación por el enunciado de las reglas paso a paso para sumar, restar, multiplicar y dividir números decimales; la traducción al latín del apellido en la palabra algorismus derivó posteriormente en algoritmo. Señala, Joyanes Aguilar, que Euclides, matemático griego (del siglo IV a.C.) quien inventó un método para encontrar el máximo común divisor de dos números, se considera con Al-KhoWârizmi el otro gran padre de la algoritmia (ciencia que trata de los algoritmos). Joyanes Aguilar, hace un señalamiento histórico con respecto a Niklaus Wirth, inventor de Pascal, Modula-2 y Oberon, profesor quien tituló uno de sus más famosos libros, Algoritmos+Estructuras de datos=Programas, señalándonos que sólo se puede llegar a realizar un buen programa con el diseño de un algoritmo y una correcta estructura de datos.

La resolución de un problema exige el diseño de un algoritmo que resuelva el mismo. La propuesta para la resolución de un problema es la siguiente:

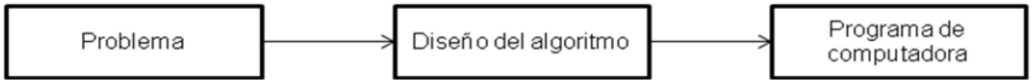

Figura 1. Resolución de un problema.
Fuente: Vázquez (2012) Análisis Y Diseño De Algoritmos.

Los pasos para la resolución de un problema son:

1.- Diseño del algoritmo, describe la secuencia ordenada de pasos, sin ambigüedades, que conducen a la solución de un problema dado. (Análisis del problema y desarrollo del algoritmo).

2.- Expresar el algoritmo como un programa en un lenguaje de programación adecuado. (Fase de codificación).

3.- Ejecución y validación del programa por computadora.

Para llegar a la resolución de un problema es necesario el diseño previo de un algoritmo, de modo que sin algoritmo no puede existir un programa.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo. Así, por ejemplo, en una analogía con la vida diaria, la receta de un platillo de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración, del mismo se realizarán sin importar el idioma del cocinero.

En la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

Dada la importancia del algoritmo en la ciencia de la computación, un aspecto importante será el diseño de algoritmos.

Joyanes Aguilar, enfatiza que el diseño de la mayoría de los algoritmos requiere creatividad y conocimientos profundos de la técnica de programación. En esencia, la solución de un problema se puede expresar mediante un algoritmo.

Características de los algoritmos

Las características fundamentales que debe cumplir todo algoritmo son:

- Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.
- Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.

Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

La definición de un algoritmo debe describir tres partes: Entrada, Proceso y Salida. Por ejemplo, en el cálculo de la edad de una persona, conociendo su año de nacimiento, la definición del algoritmo, quedaría de la siguiente manera: Entrada: la edad de la persona, información del año de nacimiento y el actual.

Proceso: realizar la diferencia del año actual menos el año de nacimiento.

Salida: visualización del resultado generado. Es decir, el resultado es la edad.

1.1.2. Aproximaciones basadas en modelos Matemáticos

Los métodos clásicos con los que se calcula la trayectoria de un robot serial se basan principalmente en los modelos matemáticos de cada estructura. En Chevallereau, (2010) se desarrollan las trayectorias viables de un manipulador no redundante dentro de un campo de restricciones a partir de las matrices de velocidad asociadas al mecanismo. Por otra parte, en Leng y Chen se desarrolla la planeación aproximada de trayectorias basada en las ecuaciones dinámicas Lagrange-Euler asociadas a la estructura del robot. Otro caso de aproximación exitosa de planeación de trayectorias está basado en teorías de optimización, por ejemplo, en Lin la planeación de trayectorias para un manipulador redundante se logra aplicando una teoría de optimización de minimización de costo en el modelo cinemático inverso, logrando que el manipulador evada singularidades y evite colisiones dentro del espacio de trabajo del robot. Por otro lado, en Yao y Gupta la planeación se logra

proyectando un grupo de restricciones al efecto final del manipulador y aplicando el método de optimización de gradientes descendentes. Los cálculos y optimización de trayectorias basados en los métodos geométricos y matemáticos suelen volverse mucho más complejos en la medida en que la configuración de los manipuladores cambia o se incluyen variables adicionales, como obstáculos o regiones singulares dentro del espacio de trabajo del manipulador, lo cual no les permite ser escalados a problemas de planeación off-line de trayectorias más elaboradas.

1.1.3. Planeación de trayectorias asistidas

De manera paralela con los métodos basados en la deducción matemática, se han desarrollado métodos alternativos para planeación de trayectorias que dependen tanto de los modelos matemáticos como de mecanismos auxiliares para calcular trayectorias. Por ejemplo, en De Luca & Oriolo (2010) se propone una planeación de trayectorias asistida por un control retroalimentado, lo que para su implementación sugiere requerir elementos esenciales tales como sensores, actuadores (del robot) y un controlador independiente. En otro caso, en que la planeación es asistida, son implementadas técnicas de visión artificial para realizar la programación de trayectorias por simulación, siendo esta una de las que más recursos demandan para su utilización dado el costo de la iluminación y cámaras de video necesarias. En Kunz se usan sensores como herramienta de percepción añadida al robot, lo cual contribuye a desarrollar un mapeo del espacio de trabajo y a la vez llevar a cabo el seguimiento de la ruta del manipulador al ejecutar tareas en un entorno dinámico.

Por último, esta aproximación más económica pero que también supone recursos adicionales es propuesta en Harada, en ella la simulación gráfica es usada para la planeación off-line de trayectorias y los sensores actúan como medio de comprobación de las posiciones y orientaciones logradas en el entorno on-line, mientras la generación de trayectorias es primero realizada mediante simulación. Este tipo de aproximaciones se basan en elementos de moldeamiento matemático y herramientas que permiten un mayor control y percepción del espacio en el que el manipulador debe interactuar; lo cual genera un mayor costo de implementación debido al aumento de los recursos usados, pero aun así prometen ser una solución

bastante robusta para el problema de planeación de trayectorias gracias a los datos dados por la percepción del entorno de trabajo.

1.1.4. Robótica cooperativa

Otro de los problemas que es abordado mediante la planeación de trayectorias es la robótica cooperativa, en la cual dos o más manipuladores comparten un espacio de trabajo y desarrollan tareas en conjunto como posicionamiento de piezas, soldadura, inspección y ensamble, entre otras. En Shin y Bien se establecen tres temas de vital importancia para manipuladores cooperativos; primero, el análisis de trayectorias libres de colisión con la inclusión de un obstáculo dinámico dentro del espacio de trabajo compartido por dos manipuladores; segundo, el análisis de trayectorias libres de colisiones entre dos manipuladores sin obstáculos dentro del espacio de trabajo; y por último, se trata el problema de optimización de tiempos de trayectorias basado en restricciones dinámicas en las estructuras.

De igual manera, en Zurawski y Phang (2011) se establece un esquema maestro-esclavo en el que el espacio de trabajo es subdivido en regiones discretas, lo que indica que se planean trayectorias no simultaneas, lo cual es de gran utilidad posteriormente en tareas asíncronas como se observa en Cheng. Un ejemplo de la coordinación de trayectorias entre tres manipuladores es logrado en Tsuji, en el cual, por medio del estudio de dinámica virtual se logra no solo el posicionamiento en común de un elemento dentro del espacio de trabajo compartido, sino también descifrar las posiciones relativas entre dos manipuladores, lo que lo convierte en un análisis paralelo de estructura para los manipuladores involucrados. Este tipo de resultados ha dado paso a un nuevo campo en la planeación e implementación de trayectorias, ya que con el objetivo de aumentar el volumen de producción en el ámbito industrial la robótica cooperativa se está convirtiendo en una necesidad.

1.1.5. Algoritmo de Aproximación

En esta sección, se presenta el algoritmo principal para maximizar el número de objetivos.

Seguimiento, o maximizar la calidad de seguimiento. Dividimos el tiempo en rondas de Duración. Consideramos el escenario en el que se utilizan medidas de rondas anteriores, Los robots son capaces de predecir el movimiento de los

objetivos de la ronda actual. Para cada Robot, creamos un conjunto de m trayectorias candidatas que se pueden seguir para la corriente redonda. Por ejemplo, estas trayectorias se pueden generar utilizando Métodos basados en el muestreo (Colin & Kelly, 2010). Nuestro objetivo es elegir una trayectoria para cada uno de los robots

Sea Rj (x) el conjunto de los objetivos previstos para ser cubiertos por x Continuación de la trayectoria Por j th robot. Creamos un sistema de conjuntos (X, R) donde X es el conjunto de todos los objetivos y R es una colección de todos los conjuntos Rj (x). Agruparemos conjuntos en R en k colecciones, una por robot. Cada Grupo contiene m conjuntos cada uno. Es decir:

$$R = \left\{ \underbrace{R_1(1), \dots, R_1(m)}_{candidate\ trajectories\ for\ r_1}, \dots, \underbrace{R_k(1), \dots, R_k(m)}_{candidate\ trajectories\ for\ r_k} \right\}$$

Una asignación válida de trayectorias puede ser representada por un mapa, σ: [1,. . . , K] → [1,. . . , M], indicando la trayectoria σ (j) (es decir, se elige el conjunto Rj (σ (j))) para el j-ésimo robot.

Podemos eliminar un objetivo del conjunto Rj (x) si no satisface una calidad mínima determinada del requisito de seguimiento

1.1.6. Trayectoria deseada

Una de las situaciones más comprometidas para el planificador, es la obtención de caminos suaves en el momento de rodear un obstáculo en la necesidad de estar en una mejor posición para adquirir información de zonas no vistas. A continuación, se propone una metodología en 2D (ejes x, y), para la obtención del planificador de trayectorias. En la Figura 01, se muestra la simulación de un ambiente en el cual el móvil se encuentre en la posición (0,0). La circunferencia representa el obstáculo, en donde están circunscritos los puntos adquiridos del objeto, y por ´ultimo la posición hacia donde nos dirigimos. Como se puede observar no es posible llegar al objetivo por medio de una línea recta, de tal forma que se debe encontrar la manera de rodear la circunferencia, se encuentran los dos puntos de intersección de la trayectoria con la circunferencia.

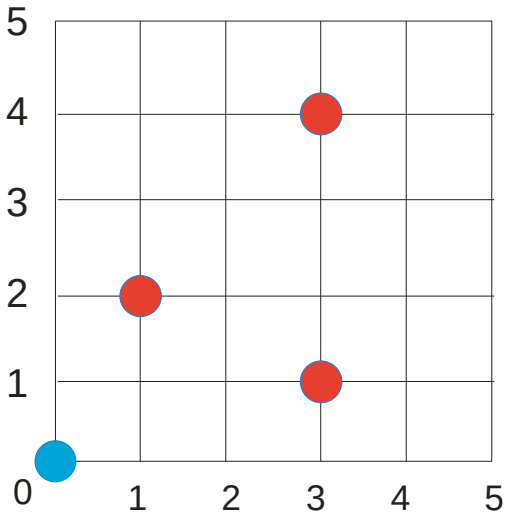

Figura 2. Posición inicial del robot y los tres obstáculos.

El proceso de planificación de trayectorias de robots móviles orientados a la reconstrucción eficiente del entorno a partir de técnicas de visión en estéreo, consta fundamentalmente de las siguientes etapas: Adquisición de los puntos en 3D de la primera escena a reconstruir por medio de visión estéreo. Reconstrucción de la escena usando los puntos en 3D hallados en el paso anterior. Calcular la posición y orientación que debe seguir el robot, para ganar la mayor información del entorno, a partir de la primera escena reconstruida. Generar una trayectoria entre la posición actual y la posición deseada para el robot. Calcular el modelo cinemático del robot móvil, con el cual se pueda implementar un controlador, de tal manera que el robot se desplace por la trayectoria deseada. Realizar iterativamente estos pasos, y de esta forma lograr una reconstrucción incremental del entorno que se desea explorar.

2.2. Antecedentes

Por el tipo de estudio desarrollado, se lograron representar los antecedentes mas importantes:

a) Internacional

Lorenzo (2012) realizó un estudio en la Universidad de Concepción, Chile, que lleva como título: “Algoritmos para la planificación y seguimiento de trayectorias en robots agrícolas”, el objetivo de esta investigación es desarrollar e implementar algoritmos de seguimiento y planificación (global y local) de trayectorias de robots agrícolas. La planificación global se realizó mediante el algoritmo A* aplicado sobre mapas de cultivo y la planificación local se realizó aplicando A* sobre un mapa 2D obtenido a partir de

imágenes 3D de los obstáculos encontrados en el camino. En cuanto el seguimiento de trayectorias, esta se realizó implementando una aproximación numérica de la trayectoria mediante el método de Euler. Los parámetros correspondientes a la dinámica del controlador de la trayectoria del robot fueron obtenidos mediante algoritmos genéticos. El mapa 3D fue generado a partir del sensor Kinect de Microsoft y sus datos procesados usando Matlab 2010b. Los resultados preliminares muestran que es posible implementar estos algoritmos en pequeños robots diseñados para cultivos hilerados. Proveyendo así, una metodología robusta que permite seguir las rutas asignadas con errores inferiores a RMSE=0.1m en trayectorias de 30m.

Fernández & Fernández (2011) presentó un estudio en la Universidad Complutense Madrid. Facultad de Informática, titulada: "Planificación de Trayectorias para un robot móvil" En el presente proyecto hemos desarrollado un sistema que controla los desplazamientos de un robot móvil SRV-1 que dispone de un módulo de visión estereoscópica. La aplicación se ejecuta en un computador que se comunica con el robot a través de Wi-Fi. Para facilitar el diseño, nuestro primer paso fue desarrollar un simulador 3D. En la memoria explicamos el proceso seguido en el desarrollo del simulador y el diseño de las trayectorias del robot. Dado el carácter abierto de la especificación del proyecto y la necesidad de coordinación con otro grupo encargado del procesamiento de las imágenes estereoscópicas, optamos por utilizar en el desarrollo del sistema una metodología de tipo OpenUp1 que facilita la incorporación de prestaciones surgidas en curso del desarrollo y afrontar problemáticas similares en futuros proyectos.

Guzmán *et al.* (2012) en su artículo presenta el estudio de: "Búsqueda de la ruta óptima mediante los algoritmos: Genético y Dijkstra utilizando mapas de visibilidad" Este artículo presenta el estudio de la generación de trayectorias entre dos puntos y una cantidad cualquiera de obstáculos entre ellos mediante el mapa de visibilidad teniendo en cuenta la geometría del robot, de igual manera en éste se plantea la comparación entre el algoritmo genético y el algoritmo Dijkstra al encontrar la ruta óptima entre las trayectorias ya generadas con el mapa de visibilidad. El algoritmo es implementado en Java, en Java Lejos versión 0.9 se desarrolla un algoritmo que envía al robot los puntos de navegación mediante bluetooth corrigiendo el error de su trayectoria y sus giros mediante el uso de los sensores como el compás, el tacómetro y el concepto de la odometría; las pruebas para obtener los resultados son aplicadas sobre el robot LEGO NXT 2.0.

Moreno (2010) Realizó un estudio en la Universidad Nacional de Colombia Bogotá titulada: "Diseño de simulador de un algoritmo para el control de un Robot Modular tipo Cadena" Este documento de tesis presenta una revisión de los conceptos básicos que definen a los robots modulares, su clasificación y los proyectos más importantes que han sido desarrollados. Concluye: El modelo se construye a partir de dos formas de control inspiradas en la naturaleza, los generadores centrales de patrones (CPG), que proveen un sistema de generación de primitivas de movimiento, y el control inspirado en mensajes hormonales, el cual funciona a manera de controlador de alto nivel integrando los diferentes movimientos generados por los CPG en base a información sensorial. El modelo híbrido de control y el modelo simple de robot modular son implementados en un ambiente simulado obteniendo como resultado el aumento en la capacidad de las cadenas de módulos para atravesar terrenos difíciles con un desempeño que depende del grado de integración de las diferentes primitivas de movimiento y que es superior al de un sistema de control basado solamente en CPG. Finalmente se describe un primer prototipo físico de robot modular tipo cadena que logra generar movimientos coordinados de locomoción gracias a la aplicación del algoritmo híbrido de control.

b) Nacional

Uriol, R. (2016) efectuó una investigación en la Pontificia Universidad Católica del Perú. Escuela de Posgrado, titulada: "Planeamiento de trayectoria y control de un robot móvil marítimo aplicando optimización por colonia de hormigas" El problema a resolver consiste en encontrar el camino más corto entre dos puntos dentro de un entorno (mapa) con obstáculos. Ambos algoritmos, tanto el de planeamiento de trayectoria como el de control óptimo, se implementaron en el entorno MatLab. Para poner a prueba el funcionamiento conjunto de estos dos algoritmos se usaron seis mapas distintos que buscan explorar el comportamiento de ambos algoritmos ante diversas variaciones de un caso base de comparación. El tiempo de convergencia de los algoritmos y los parámetros que ajustan sus desempeños fueron analizados.

Jaramillo, Matta y Correa (2016) presentaron un estudio en la Pontificia Universidad Católica del Perú. Escuela de Posgrado, titulada: "Plataforma de software para modelado y simulación de robots manipuladores" Robomosp está organizado por subsistemas jerárquicos: interfaz gráfica de usuario, subsistema de modelado 3D, subsistema de simulación gráfica 3D, subsistema robótico, subsistema de programación y API

(comandos IOCI: Input Output Command Interface) que sirve de interfaz para intercambiar información con aplicaciones externas. Concluyen: Dado que el proceso de construcción de Robomosp ha sido de tipo evolutivo, se ha utilizado el modelo en espiral basado en componentes como metodología de desarrollo. Esta metodología permite el diseño e implementación de sistemas cuyas funcionalidades son mejoradas o adicionadas al avanzar en ciclos, previamente definidos, de la espiral central de desarrollo. Conjuntamente se deben seguir los lineamientos de conexión y organización entre los módulos del sistema que se especifican en la arquitectura del software.

Melillanca & Burgos (2016) presentaron un estudio en la Universidad, Facultad de Ciencias de Ingeniería, titulada: “Diseño y programación de algoritmos de control en robots móviles. Estudio y aplicación a Robotino – Festo” Trabajan y diseñan algoritmos basado en el software de control Robotino View, para el Robot omnidireccional Robotino de la empresa de automatización alemana Festo S.A, de su división Didactic. La finalidad de diseñar algoritmos de control, se basa en la necesidad de llevar al máximo la potencialidad del software, como el hardware construido por el fabricante, de esa manera, explorar campos como el posicionamiento, visión artificial, el uso de sensores y la simulación de algoritmos basados en lógica digital. De esta manera se realiza un estudio de su software y hardware, como de sus limitaciones, para ello se genera una serie de explicaciones, set de ejemplos de diferentes funciones y especificaciones del equipo. Detallando varias medidas a considerar durante la utilización y puesta en marcha, como también problemas de conectividad, precisión y diferencias de valores en los ejemplos realizados.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1. Identificación del problema

Los avances tecnológicos recientes permiten construir equipos de sensores y Robots que pueden detectar datos de lugares difíciles de alcanzar en un espacio inaccesible para el ser humano tanto por su tamaño como por el riesgo de su vida.

Los sistemas de detección robótica tienen el potencial de revolucionar una colección de diversas aplicaciones como la agricultura, monitoreo ambiental, estudios climáticos, seguridad, a vigilancia en un futuro próximo responderá a las comodidades del hogar, el trabajo haciendo más placentera la calidad de vida de la humanidad. Para aprovechar plenamente esta tecnología, es crucial para complementarla con algoritmos eficientes que planifiquen la detección en estos sistemas. En esta investigación, simulamos y desarrollamos nuevos algoritmos de planificación de sensores y presentamos resultados óptimos para la trayectoria de un robot.

En la primera parte de esta investigación de tesis, buscamos los algoritmos que se tiene en diversos trabajos similares los mismos que los catalogamos y los precisamos mediante una representación adecuada.

En la segunda parte de esta investigación, estudiamos los problemas que pueden resolver estos algoritmos de planificación de ruta en robots, viendo los casos más difíciles y más fácil y el caso promedio. Comenzamos por investigar cómo planificar el movimiento de un equipo de Robots encargados de rastrear objetivos que se mueven en el suelo. Entonces estudiamos varios problemas de cobertura que surgen en aplicaciones de monitoreo.

Esta investigación hace progresos hacia la construcción de sistemas de detección robótica, en dos direcciones. Presentamos algoritmos con fuertes garantías de rendimiento teórico, probando que nuestros algoritmos son óptimos o que sus costes son como máximo constante de los valores óptimos. También demostraremos la viabilidad y aplicabilidad de nuestros resultados a través de la simulación en la implementación del sistema.

Pretendemos responder las interrogantes:

2.2. Enunciados del problema

a) Pregunta general

- ¿En qué medida es posible simular algoritmos óptimos y eficientes que permitan una óptima ruta para la locomoción de un Robot?

2.3. Justificación

Los sistemas de detección robótica tienen el potencial de revolucionar una colección de diversas aplicaciones como la agricultura, monitoreo ambiental, estudios climáticos, seguridad, a vigilancia en un futuro próximo responderá a las comodidades del hogar, el trabajo haciendo más placentera la calidad de vida de la humanidad. Para aprovechar plenamente esta tecnología, es crucial para complementarla con algoritmos eficientes que planifiquen la detección en estos sistemas. En esta investigación, simulamos y desarrollamos nuevos algoritmos de planificación de sensores y presentamos resultados óptimos para la trayectoria de un robot.

2.4. Objetivos

2.4.1. Objetivo general

Simular algoritmos para una mejor trayectoria de la locomoción de un robot en los laboratorios de Electrónica, 2018

2.4.2. Objetivos específicos

1. Identificar los principales algoritmos de planificación en la trayectoria de un Robot en 2D.
2. Construir la prueba de los algoritmos de trayectoria en el mejor y peor caso.
3. Evaluar la simulación de los algoritmos de la trayectoria óptima.

2.5. Hipótesis

2.5.1. Hipótesis general

Con la simulación de los algoritmos Walk Random y Tree Guarding Walk y aproximación es posible optimizar la trayectoria de un robot en los laboratorios de Electrónica, 2018

2.5.2. Hipótesis específicas

Para la ejecución de esta investigación científica se tendrá en cuenta de acuerdo al modelo y tipo de investigación la comprobación de una hipótesis debiendo esta ser afirmada o negada en relación a la comprobación de los datos recabados y la información que fue recolectada, tomando en cuenta la funcionalidad del instrumento y la practicidad de los mismos, la misma que se plantea en hipótesis general.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. Lugar de estudio

La presente investigación se realizará en la Universidad Nacional del Altiplano la cual está ubicada en el distrito, provincia y departamento de Puno, específicamente en el Barrio de San José con dirección Avenida Sesquicentenario 1121, en donde a la fecha viene funcionando sus instalaciones, del mismo modo cuenta con la sus oficinas administrativas la cual se encuentra ubicada en la Avenida El Sol 570; para el caso nuestro nos enfocaremos en el campus universitario donde están las instalaciones de la EP. de Ing. Electrónica, específicamente en los laboratorios de la escuela.

Figura 3. Universidad Nacional del Altiplano Puno
Fuente: Revista Institucional de la UNA Puno

3.2. Población

Se La población está constituida por todos los algoritmos de planificación de la trayectoria de un robot. En este caso nos hemos basado en los modelos de algoritmos Model Walk Random y Tree Guarding Walk, para hacer la búsqueda de la trayectoria a través del grafo. Escogimos éstos por ser eficiente y flexible, aparte de tener como propiedad ser

unos algoritmos completos, es decir, si existe una solución siempre dará con ella. Esto es importante puesto que nos da fiabilidad y estabilidad a la hora de calcular una trayectoria.

3.3. Muestra

La muestra está constituida por la totalidad de la población, los que son los algoritmos que tengan facilidad en su implementación y simulación.

Tipo de muestra no Probabilística aleatoria simple, según Carrasco (2005) donde destaca, que cada miembro de la población tiene una igual e independiente de ser seleccionado como parte de la muestra. Las palabras claves son aquí igual e independientes. Igual porque no existe una predisposición a escoger una persona por otra.

3.4. Método de investigación

La metodología de la investigación científica es encargada de orientar el proceso de investigación, proporcionando información detallada sobre las acciones que permitieron describir y analizar el problema planteado. Para ello se formulará los distintos componentes que lo integran:

Enfoque de Investigación

Según Hernandez *et al.* (2006), el enfoque de la investigación utilizado en la presente investigación es Cuantitativa, debido a que se usa recolección de datos para probar la Hipótesis, con base en la medición numérica y el análisis estadístico para establecer patrones de comportamiento y probar teorías. Por lo cual se trabajará en base a los siguientes objetivos planteados de la investigación.

La metodología aplicada en el proceso de la investigación como se mencionó en diferentes apartados decidimos utilizar para el desarrollo de este proyecto una metodología OpenUP. Según Pressman (2005) la metodología Open Unified Process, o también conocida en castellano como Proceso Unificado, entiende el proyecto como un ciclo de vida del que los componentes del equipo son parte orgánica del mismo.

3.5. Descripción detallada de métodos por objetivos específicos

Para el Objetivo:

Identificar los principales algoritmos de planificación de la trayectoria de un Robot en 3D:

Se realizará un estudio del estado del arte recurriendo la búsqueda en otros trabajos presentados en casos de movimiento de robot, se documentará y se precisará los algoritmos en la investigación.

Se analizará las variables de las coordenadas iniciales, las coordenadas finales, el tiempo de ejecución del algoritmo, se utilizará un equipo portátil para registrar los algoritmos.

Para el Objetivo:

Prueba de los algoritmos de trayectoria en el mejor y peor caso:

Se codificará el algoritmo mediante un lenguaje de programación adecuado, utilizando unos compiladores open source, utilizaremos también un simulador para contrastar los datos de entrada y salida de las trayectorias.

Para el Objetivo:

Simular los algoritmos de trayectoria:

Utilizando el simulador y los programas open source se realizará una simulación con datos que podría presentarse en situaciones reales para evaluar la performance del algoritmo.

3.5.1. Tipo de investigación

La investigación es de tipo aplicativo – Experimental, considerando que se realizan en condiciones controladas, en las cuales el efecto de las fuentes de invalidación interna es eliminado, así como el de otras posibles variables independientes que no son manipuladas o no interesan (Hernández- Sampieri *et al.,* 2013 y Crano, 2003), generalmente logran un control más riguroso (Festinger, 1993), pero estos últimos suelen tener mayor validez externa. Ambos tipos de experimento son deseables.

3.5.2. Diseño De Investigación

Debido a que analizan las relaciones entre una o más variables independientes y una o más dependientes, así como los efectos causales de las primeras sobre las segundas, son estudios explicativos. Se trata de diseños que se fundamentan en el

enfoque cuantitativo y en el paradigma deductivo. Se basan en hipótesis preestablecidas, miden variables y su aplicación debe sujetarse al diseño concebido con antelación; al desarrollarse, el investigador está centrado en la validez, el rigor y el control de la situación de investigación. Asimismo, el análisis estadístico resulta fundamental para lograr los objetivos de conocimiento. Como señalan Feuer, Towne & Shavelson (2002), su fin es estimar efectos causales.

3.5.3. Métodos de investigación

Se tomó en cuenta el método deductivo. Según Bernal (2010), el método deductivo que consiste en tomar conclusiones generales para explicaciones particulares. El método se inicia con el análisis de los postulados, teoremas, leyes, principios de aplicación universal, para aplicarlos a soluciones o hechos particulares.

3.5.4. Técnicas e instrumentos

Para la programación de la aplicación presentada en la presente investigación se utilizó LabVIEW, software desarrollado por National Instruments, debido a que proporciona todas las herramientas requeridas para el desarrollo del Informe de tesis.

La investigación tuvo un entorno de programación gráfica usado para desarrollar sistemas sofisticados de medida, pruebas y control a través de íconos gráficos y cables que parecen un diagrama de flujo. Ofrece una integración con miles de dispositivos de hardware y brinda cientos de bibliotecas integradas para análisis avanzado y visualización de datos. El desarrollo del programa tiene varias etapas, como primer paso se tiene la adquisición de datos, es decir el ingreso del mapa de entorno del robot, después se tiene una etapa de procesamiento de estos datos donde se obtienen los puntos que conforman los bordes de los obstáculos pertenecientes al mapa de entorno. Se halla el diagrama en base a los puntos hallados y seguido se procedió a discriminar los puntos del diagrama generado que no corresponden a lo requerido, por último, en base al diagrama final se obtiene la trayectoria óptima a través del algoritmo y la misma es descargada en la plataforma móvil de Festo, (Diagrama de bloques de la programación)

3.5.5. Para recolección de datos

Que consiste en el acopio de la información requerida en la investigación, tales como revisión de trabajos de referencia a nuestro tema de investigación, artículos, que fueron realizados con anterioridad para determinar el nivel de ejecución del presupuesto en relación al cumplimiento de las metas, Así mismo se justifica los errores cometidos debido a la posible inexactitud de la posición de partida del robot, así como la toma de datos respecto a un sistema de referencia elegido.

Prueba 1: El mapa de entorno ingresado, contiene 5 obstáculos de varias formas y tamaños, el diagrama y la trayectoria obtenida a seguir.

Prueba 2: El mapa de entorno ingresado, es exactamente el mismo de la prueba anterior.

Prueba 3: El mapa de entorno ingresado para esta prueba es el del laboratorio de Microprocesadores, el cual comprende un contorno rectangular de 6,97x7,6 (m), posee 6 obstáculos de forma rectangular mismos que son de diferentes tamaños, el diagrama y trayectoria generada por la HMI a seguir por el Robot en base a los puntos de partida y llegada deseados se muestran.

Para análisis de errores, se tomaron 9 puntos a lo largo de la trayectoria, tanto teóricos como prácticos y se obtuvieron los resultados mostrados en las Tablas de resultados.

Tabla 1

Validación del Instrumento

Alfa de Cronbach	Alfa de Cronbach Basada en elementos estandarizados	Número de elementos
,806	,808	20

El coeficiente Alfa de Cronbach 0,806 como resultado, indica que los instrumentos de prueba formulada es consistente como instrumento y por ende es Fiable, dando así su validez respectiva para su aplicación en esta investigación desarrollada; cabe

resaltar que el valor resultante está por encima de los valores estándar resultantes de otras investigaciones.

Para Procesamiento y análisis de datos

Los datos clasificados, ordenados y presentados son analizados, a fin de descubrir las causas y sus efectos y la relación que existe entre ellos. La investigación tendrá un análisis, expresado en conclusiones referenciales. Mediante la aplicación de las distintas técnicas que nos brinda (Bernal C. , 2010) y (Hernandez, Fernandez, & Baptista, 2006), en sus publicaciones referentes a la Metodología de la Investigación Científica, las cuales procederemos a desarrollar en los siguientes párrafos de manera que se tenga una mejor comprensión y entendimiento.

- Tabulación de datos: Los datos obtenidos de la aplicación de la investigación fueron tabulados mecánicamente es decir mediante las TIC's, para poder procesar posteriormente con un software estadístico.
- Procesamiento estadístico: El análisis y procesamiento de los datos se hizo utilizando el software Statistical Package for the Social Sciences. Este es uno de los programas estadísticos más conocidos teniendo en cuenta su capacidad para trabajar con grades bases de datos y un sencillo interface para la mayoría de los análisis.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. Aspectos generales

Se han efectuado una por una las diferentes pruebas para probar el correcto funcionamiento de todos los algoritmos integrados en la aplicación, a continuación, mostramos unas cuantas pruebas relacionadas con las principales funcionalidades del sistema y la información recopilada por cada una de ellas.

Partimos desde la prueba más sencilla con la cual se cubría el primer objetivo hasta las distintas funcionalidades que se han ido añadiendo a continuación.

Cumpliendo el propósito de la investigación sobre la prueba de algoritmos, podemos señalar el entorno gráfico de la herramienta, destacar el simulador tridimensional, el cual nos permite ver con claridad y sencillez el escenario, la situación del robot y evaluar la calidad de la trayectoria generada. Hay que destacar que el simulador se comporta como el robot real, es decir, no tiene ninguna referencia de orientación. Los giros se hacen por tiempos de la misma forma que el robot, lo que nos permite más realismo a la hora de la ejecución de la simulación. Esto es importante porque dependiendo del ajuste que se dé al robot en cuanto a fuerza de ruedas, peso, fricción…podríamos ajustarlo para cualquier tipo de terreno y tener una precisión simulador-realidad mayor.

La herramienta también está preparada para tener un sistema de referencia, mediante GPS que en este caso se simularía con un sistema de cámaras que nos proporcionaría la posición del robot en coordenadas relativas enviadas a través del protocolo de transmisión UDP. Un sistema como éste nos proporciona mayor precisión puesto que al no tenerlo estamos moviéndonos con una navegación estimada.

A parte de que la herramienta tenga la posibilidad de la planificación de caminos con un entorno conocido o sin conocer, damos la posibilidad de comunicar directamente con el robot y enviarle directamente instrucciones de movimiento, captura de imágenes y video en tiempo real.

Debido a la envergadura del proyecto, éste fue divido en dos partes. La parte correspondiente a nuestro grupo se centra en la estrategia de planificación de rutas mientras que la del otro grupo se encargaría del tratamiento de imágenes. Para no depender de éste se decidió crear un algoritmo de detección de objetos por colores mediante la disparidad.

En los siguientes apartados veremos cómo se ha ido desarrollando el proyecto, las fases y objetivos que se han ido superando, los problemas que han ido surgiendo a lo largo de éste y cómo se han ido solventando. Con esto daremos una idea más específica de la dificultad a la que puede llevar desarrollar una herramienta de este tipo.

4.2. Identificar los principales algoritmos de planificación de la trayectoria de un robot en 3D

En este caso nos hemos basado en los modelos de algoritmos Model Walk Random y Tree Guarding Walk, para hacer la búsqueda de la trayectoria a través del grafo. Escogimos éstos por ser eficiente y flexible, aparte de tener como propiedad ser unos algoritmos completos, es decir, si existe una solución siempre dará con ella. Esto es importante puesto que nos da fiabilidad y estabilidad a la hora de calcular una trayectoria.

Se analizará las variables de las coordenadas iniciales, las coordenadas finales, el tiempo de ejecución del algoritmo, se utilizará un equipo portátil para registrar los algoritmos.

4.3. Prueba de los algoritmos de trayectoria

Se codificó el algoritmo mediante un lenguaje de programación adecuado, compiladores open source, utilizaremos también un simulador para contrastar los datos de entrada y salida de las trayectorias.

Módulo de planificación de trayectorias

El módulo de planificación de trayectorias se encarga de encontrar una ruta transitable entre una posición de origen y una posición objetivo. Para el cálculo de la trayectoria

utilizamos el algoritmo A* (Model Walk Random) B* (Model Tree Guarding Wal) por su eficiencia. A los algoritmos le debemos proporcionar información del entorno a través de una matriz de pesos (costes). Esta matriz se calcula en las coordenadas a partir de la lista de obstáculos.

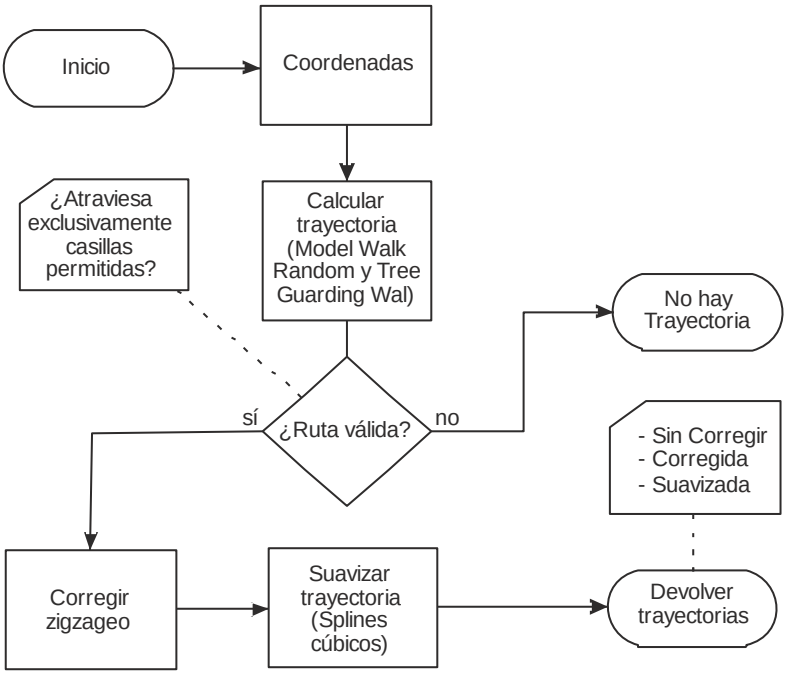

Figura 4. Diagrama de Bloques de Corrección Zigzageo

Las coordenadas se llevan a cabo a partir de la lista de obstáculos. El primer paso, es crear coordenadas o en su defecto una cuadrícula. Inicialmente, todas las coordenadas son permitidas. A continuación, se prohíben las coordenadas defectuosas o los recorridos que contengan total o parcialmente un obstáculo. Por último, se establecen como coordenadas de seguridad las contiguas a las prohibidas. El proceso se detalla en las trayectorias de los primeros 10, 20, 30 de las 100 corridas, el que visualiza en las siguientes ilustraciones:

TRAYECTORIAS: 100 corridas, algoritmo A* (Model Walk Random)

Se codificó el algoritmo mediante un lenguaje de programación adecuado, compiladores open source, utilizaremos también un simulador para contrastar los datos de entrada y salida de las trayectorias.

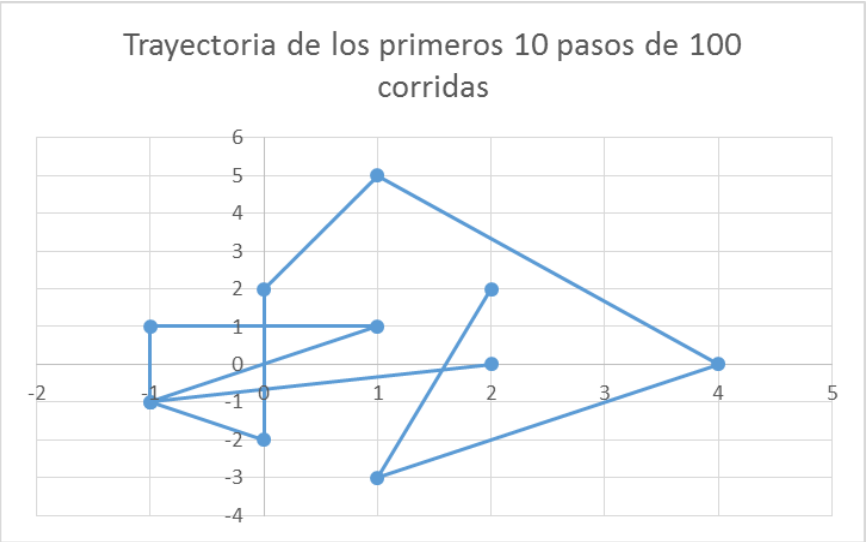

Figura 5. Primeros 10 Pasos del 1er Algoritmo

Interpretación

En la tabla se observa la trayectoria de 100 corridas, del algoritmo A según el modelo Model Walk Random. Que se observa que en la primera trayectoria que tiene como inicio se tuvo (2,0) con una distancia a (-1,-1), el recorrido ha seguido su trayectoria a (1,1) que posterior a éste el recorrido (1.-1), que se desplaza al mismo punto de la segundo distancia (-1,-1), para seguir una trayectoria de un cuadrante a (-2,0), para lo cual recorre cuatro cuadrantes hasta (2.0) posteriormente seguir a la intersección (5,1), el recorrido continua al (4,0) y como penúltimo recorrido tienen una trayectoria hasta (-3,1) y por último la trayectoria se desplaza a (2,2).

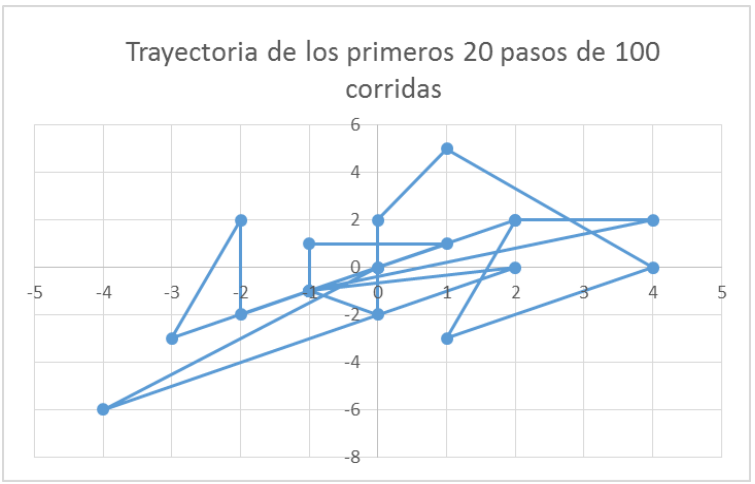

Figura 6. Primeros 20 Pasos del 1er Algoritmo

Interpretación

Considerando las trayectorias anteriores según los pasos ya demostradas de trayectorias 10 de 100 corridas, se continúa con las trayectorias del algoritmo A según el modelo Model Walk Random. Que explica que la décimo primer recorrido se ubica en el punto (2,2) con una distancia a (2,4), el recorrido ha seguido su trayectoria a (-1,-2) que posterior a éste el recorrido (0.2), que se desplaza al mismo punto de la segundo distancia de (-4,-6), para seguir una trayectoria de un cuadrante a (0,0), para lo cual recorre hasta (-3,-2.5) posteriormente seguir a la intersección (2,-2), el recorrido continua al punto final de (-2,-2). Como se aprecia, las trayectorias de los primeros 20 pasos de 100 corridas no es muy dificultoso, visualizándose según la trayectoria que el robot buscará el objeto extraño en un tiempo de: 0.12080425 segundos, sin considerar obstáculos que ya son conocidos por el robot. Que lo que ocurren las trayectorias de 10 y 20… de las 100 corridas.

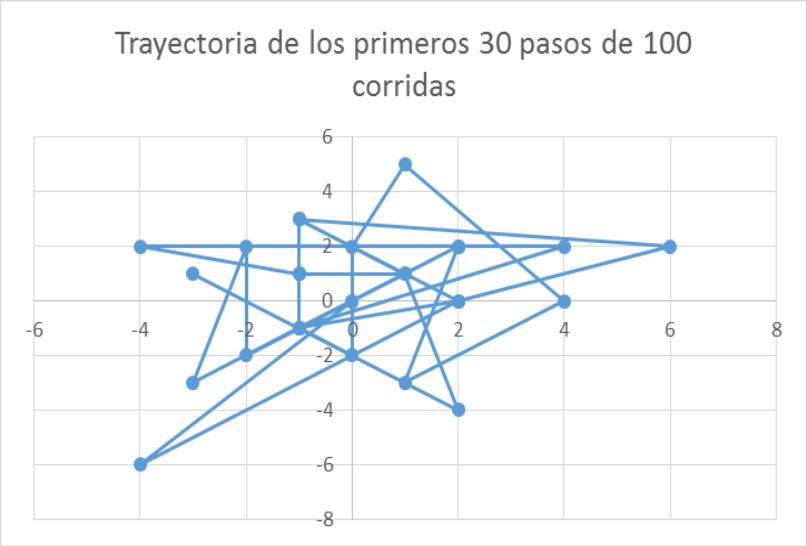

Figura 7. Primeros 30 Pasos del 1er Algoritmo

Interpretación

Como se aprecia, la trayectoria de los primeros 30 pasos de 100 corridas no es muy dificultoso, visualizándose según la trayectoria que el robot buscará el objeto extraño en un tiempo de: 0.240170956 segundos, sin considerar obstáculos que ya son conocidos por el robot. Que lo que ocurren las trayectorias de 10 a 30… de las 100 corridas.

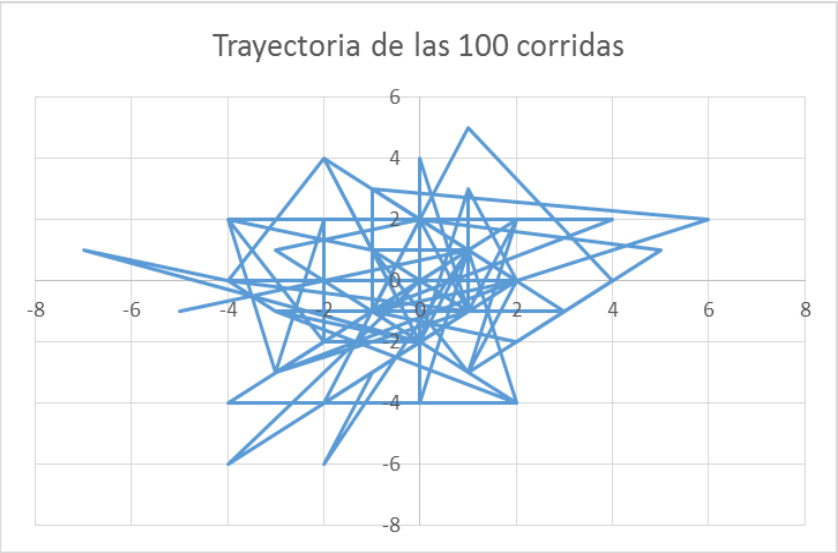

Figura 8. Primeros 100 Pasos del 1er Algoritmo

Interpretación

Como cabría esperar, los obstáculos lejanos no influyen en las coordenadas. Por obstáculo lejano se considera cualquier obstáculo que no está localizado ni siquiera parcialmente en la trayectoria.

Como se aprecia, la trayectoria de las 100 corridas no es muy embarazoso, visualizándose según la trayectoria que el robot buscará el objeto extraño en un tiempo de: 0.480214189 segundos, sin considerar obstáculos que ya son conocidos por el robot. Que lo que ocurre en las 100 corridas.

Trayectorias: 1000 corridas, B* (Model Tree Guarding Wal)

Se codificó el algoritmo mediante un lenguaje de programación adecuado, compiladores open source, utilizaremos también un simulador para contrastar los datos de entrada y salida de las trayectorias.

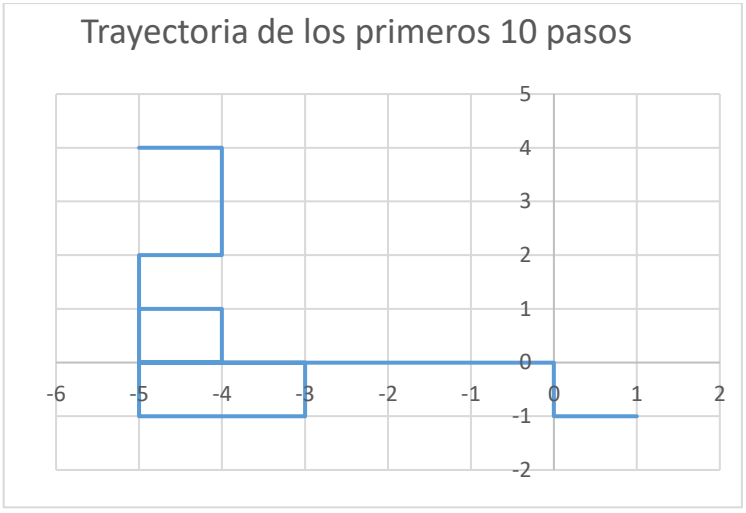

Figura 9. Primeros 10 Pasos del 2do Algoritmo

Interpretación

En la tabla se observa la trayectoria de 1000 corridas, del algoritmo B según el modelo Model Tree Guarding Wal, en la primera trayectoria de 10 pasos que tiene como inicio; se tuvo (1,-1) con una distancia a (-1,0), el recorrido ha seguido su trayectoria a (-5,0) que posterior a éste el recorrido (-5,-1), que se desplaza al mismo punto de la segundo distancia (-3,0), para seguir una trayectoria de un cuadrante a (-5,2), para lo cual recorre un cuadrante hasta (-4.2) posteriormente seguir a la intersección (-4,4), el recorrido continua al (-5,4)

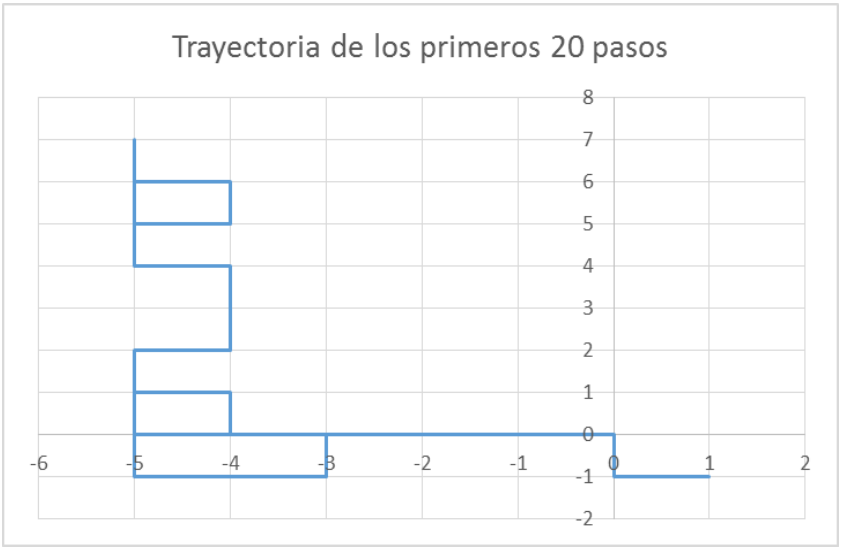

Figura 10. Primeros 20 Pasos del 2do Algoritmo

Interpretación

En la tabla se observa la trayectoria de 1000 corridas, del algoritmo B según el modelo Model Tree Guarding Wal, seguida en la primera trayectoria de 10 pasos que tiene como inicio; se tuvo (-5,4) con una distancia a (-5,5), el recorrido ha seguido su trayectoria a (-4,6) que posterior a éste el recorrido llega al punto (-5,-6), que se desplaza al mismo punto de (-5,5), para seguir una trayectoria de un cuadrante a (-5,7).

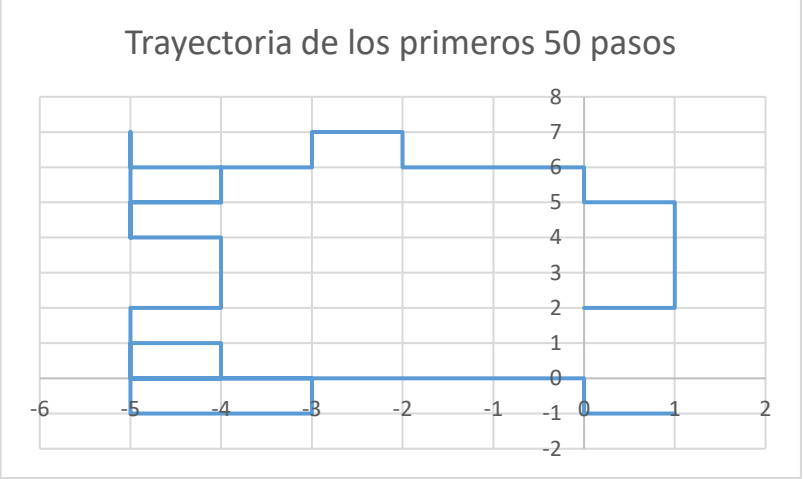

Figura 11. Primeros 50 Pasos del 2do Algoritmo

Interpretación

Como se puede observar en las trayectorias de las coordenadas de los 10, 20, 50… primeros pasos de 1000 corridas, efectuándose en un tiempo de: 0.8876290321 segundos; son trayectorias ya reconocidas por el robot, sin embargo, que haya obstáculos fuera de las coordenadas, es habitual. Asimismo, aunque menos frecuente, el objetivo marcado podría exceder los límites de las coordenadas. En ese caso, se tomará como objetivo real de la planificación aquella coordenada que siendo permitida esté más cerca del objetivo marcado (cuerpo extraño). Hay que recordar que las dimensiones de las trayectorias son constantes. Razón por la cual, adaptando las dimensiones de las coordenadas a las necesidades de la sesión de trabajo se evita esta situación. No obstante, ha sido prevista.

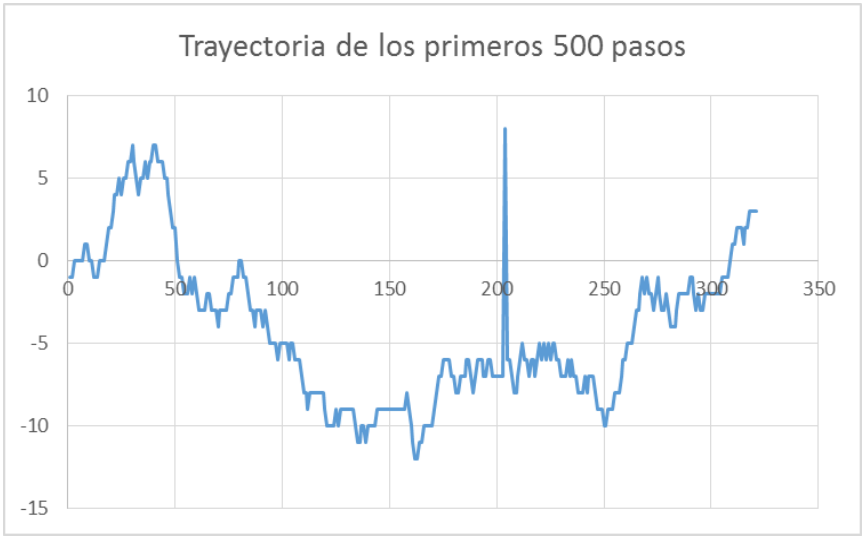

Figura 12. Primeros 500 Pasos del 2do Algoritmo

Interpretación

Por último, apreciemos un último contratiempo: la finitud del escenario en la simulación 2D. El escenario es extenso, pero no ilimitado. Debemos, por consiguiente, impedir que el robot de la simulación se precipite al vacío.

Consideramos, por tal razón, a toda coordenada que excede los límites del escenario, para evitar riesgos innecesarios, estas coordenadas prohibidas estarán rodeadas por los correspondientes recorridos de seguridad.

La trayectoria devuelta por los algoritmos A* (Model Walk Random) y B* (Model Tree Guarding Wal) contendrá zigzagueos. Este efecto es indeseado pues conlleva que el robot realice más giros de los necesarios para recorrerla. Se corrigió, por tanto, la trayectoria quitando los zigzagueos.

El algoritmo que corrige la trayectoria es sencillo, se basa en eliminar todos los puntos intermedios posibles. Dados dos puntos A y B de una ruta, si la línea recta que los une atraviesa exclusivamente coordenadas permitidas, entonces los puntos intermedios pueden omitirse, de forma que queda A directamente unido con B en línea recta y, por tanto, sin zigzagueo.

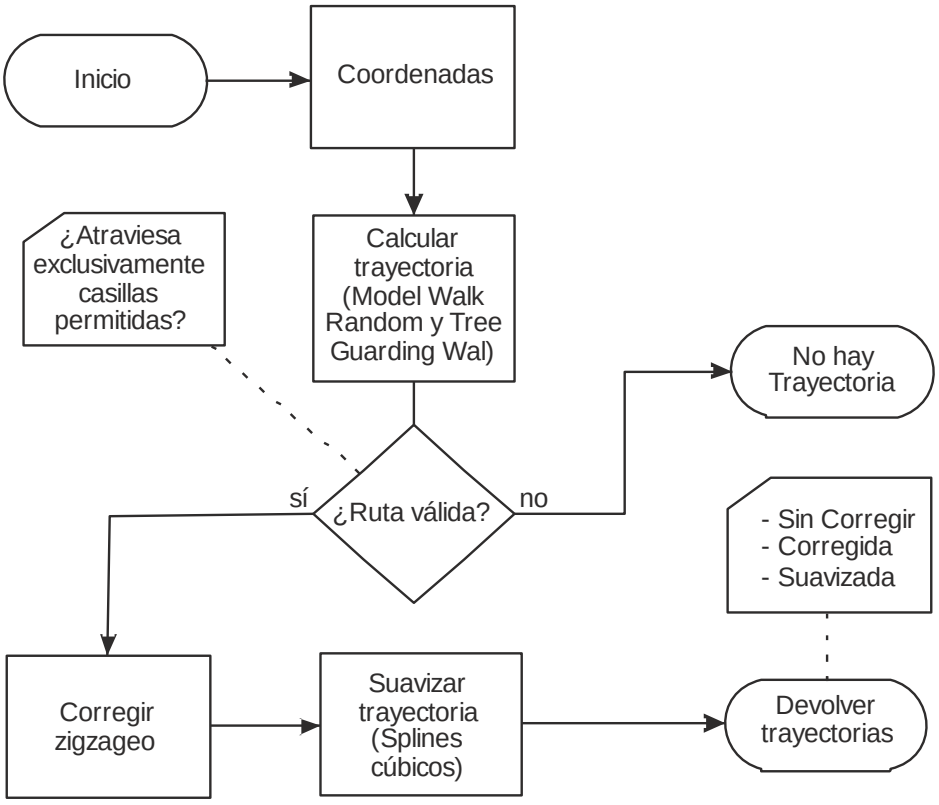

Figura 13. Diagrama de flujo de corrección de zigzageo

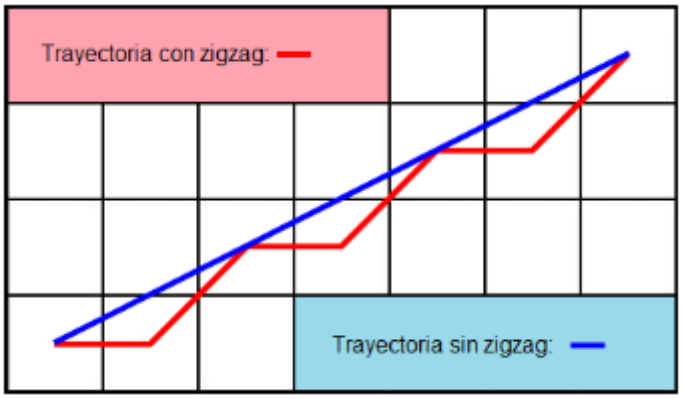

Figura 14. Corrección del zigzagueo de la trayectoria

Una vez corregida la trayectoria, se puede suavizar mediante splines cúbicos. El módulo de planificación aplica los splines cúbicos, aunque luego no se utiliza.

Cálculo de Acciones

Este módulo se encarga de obtener las acciones necesarias para la ejecución de la ruta por parte del robot. Estas acciones siguen un patrón establecido al que llamamos paso. En

cada uno de los pasos el robot ha de realizar dos acciones esenciales: un giro y un avance. La totalidad de los pasos se agrupan a su vez en una estructura denominada lista de pasos. En la implementación de este módulo aparecen tres clases: Lista de pasos. cs, Paso. cs y Acción. Cs.

Lista de pasos. cs es la utilizada desde el módulo principal. Su constructora recibe como parámetros la rotación del robot y la trayectoria.

En la ilustración de la siguiente página, aparece la estructura de la lista de pasos y su forma de ejecución.

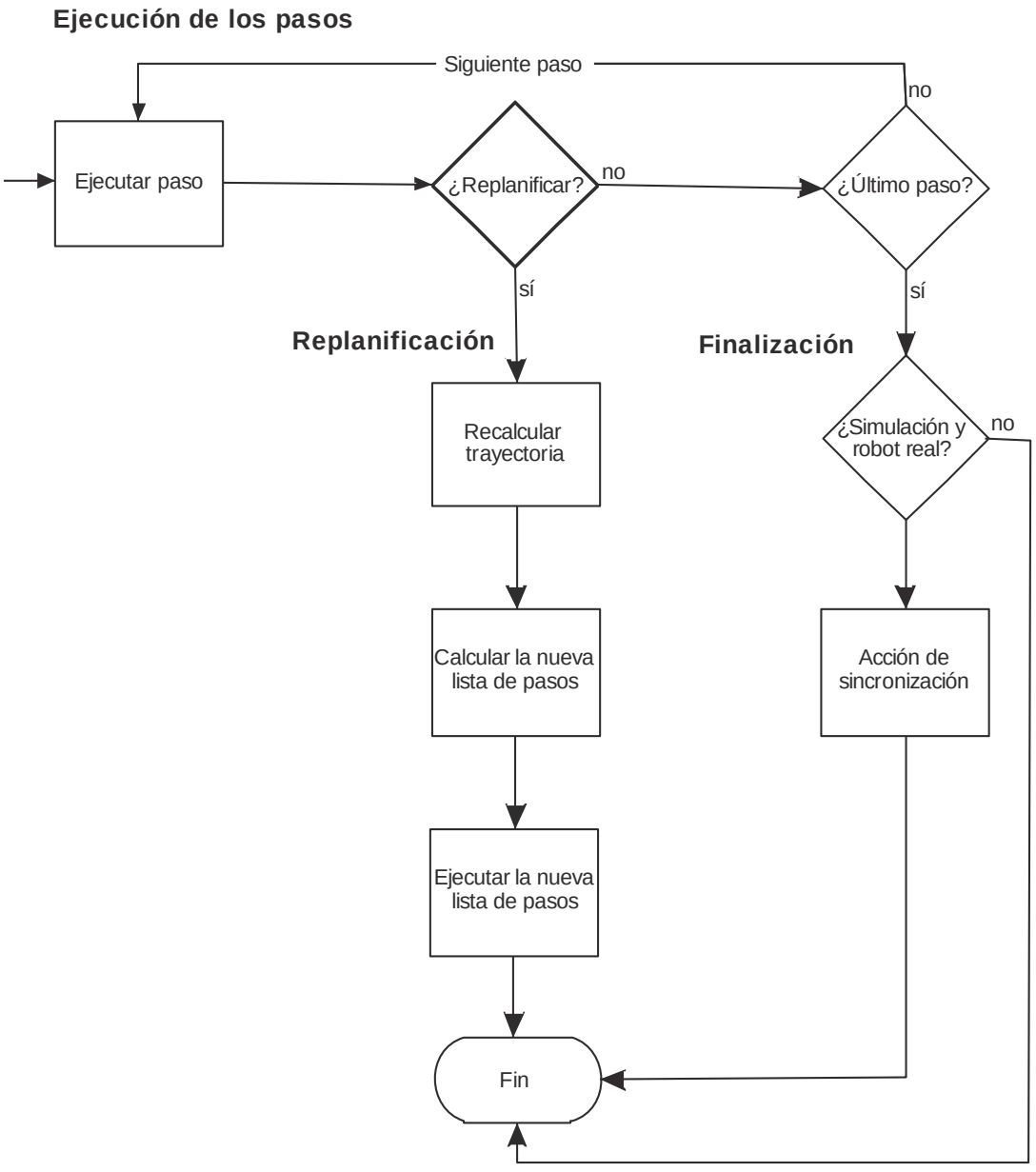

Figura 15. Diagrama de Flujo de los Pasos a Seguir del Robot

Como ya hemos comentado, las dos acciones principales de un paso son girar y avanzar. En una acción de giro el robot rota sobre su eje vertical el número requerido de grados. Este número jamás supera 180º puesto que disponemos de ambos giros: a derecha y a izquierda. Por su parte, en el avance, el robot se desplaza una cantidad señalada en la dirección y sentido indicados por su rotación actual.

Para que las acciones anteriores se traduzcan en la ejecución exitosa de la ruta, son necesarias unas acciones complementarias que comprueben la posición del robot. En este tipo de acciones se compara lo que se esperaba a priori que realizara el robot con lo que realmente ha realizado. En concreto, hay dos acciones de comprobación. En una sólo se comprueba el giro, en la otra se comprueba además la posición. La primera se incluye tras el giro. La segunda, detrás del avance.

Las acciones de comprobación están presentes en todas las listas de pasos. Aunque, si el GPS no está disponible, se ignoran.

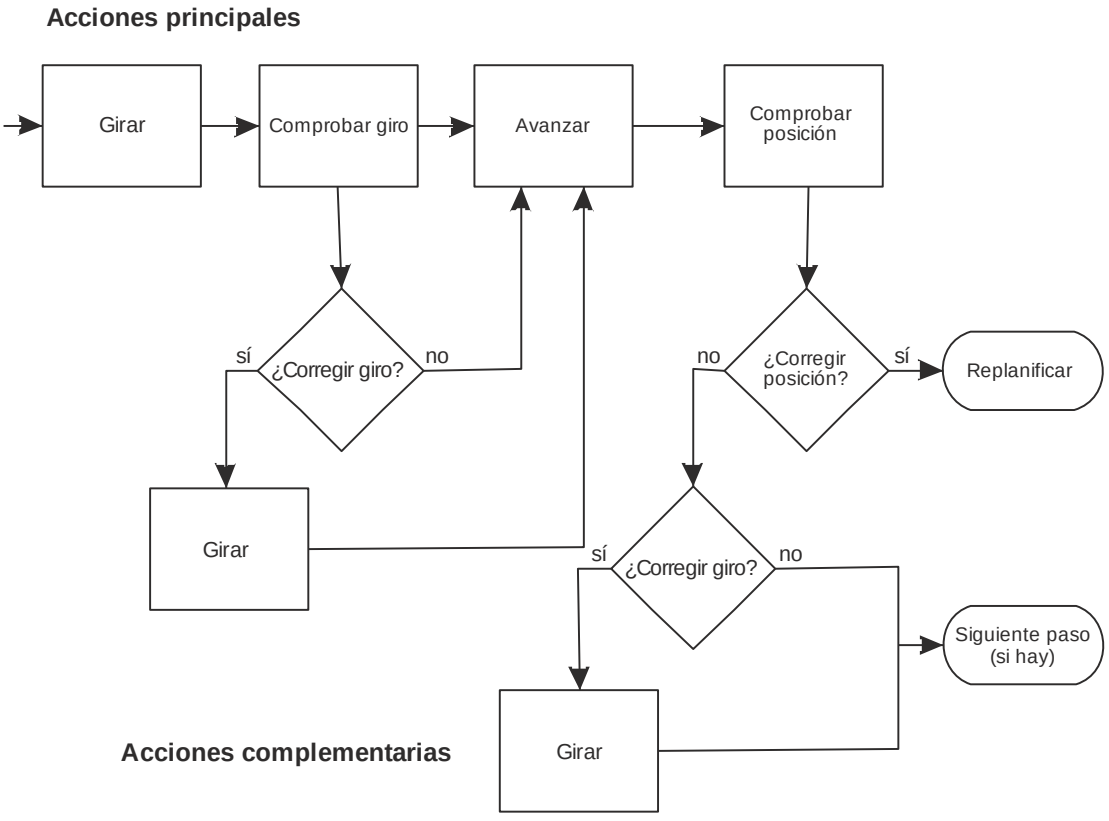

Figura 16. Acciones Principales a Seguir

Se muestra en la siguiente ilustración la estructura de un paso y su ejecución.

Además de las acciones comentadas hasta ahora, existen dos acciones más: visión y sincronismo. Estas acciones no están presentes en todas las listas de pasos.

La acción de visión estará presente si el usuario la ha pedido a través de la interfaz gráfica. En este caso, la acción de visión se añade en todos los pasos tras la comprobación de giro. Esta acción se encarga de capturar imágenes y comprobar si hay elementos extraños o nuevos.

La acción de sincronismo se introduce únicamente al final del último paso de la lista de pasos si ésta va a ser ejecutada simultáneamente por el robot real y el robot de la simulación. El propósito de esta acción es que el robot de la simulación tenga la misma rotación que el robot real al finalizar la ruta.

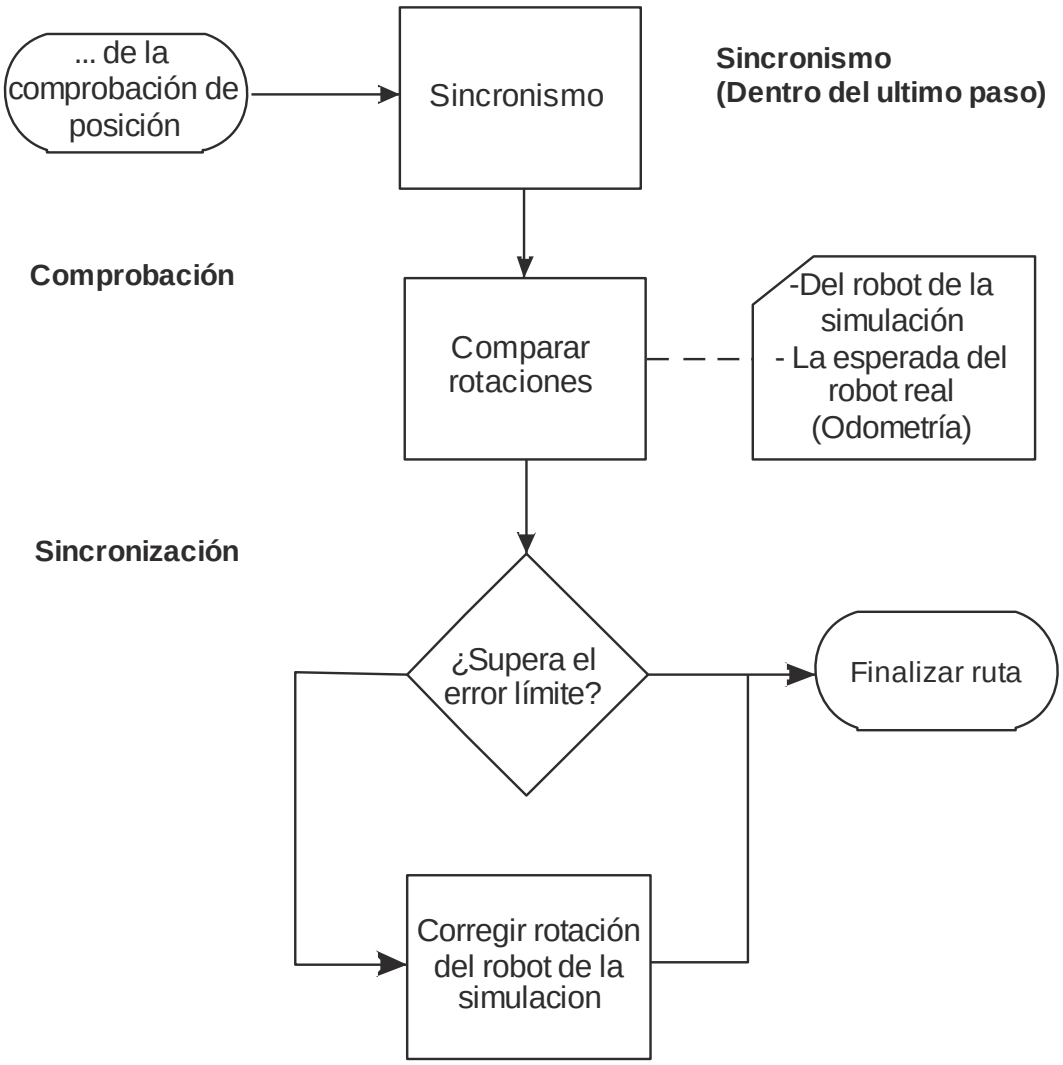

Figura 17. Pasos de Sincronismo del Robot

Búsqueda del objetivo

Este módulo se encarga de la ejecución de un grupo extraño en la trayectoria de los algoritmos. Simulamos con una bola roja. En este modo de ejecución el robot ha de buscar una bola roja y aproximarse a ella.

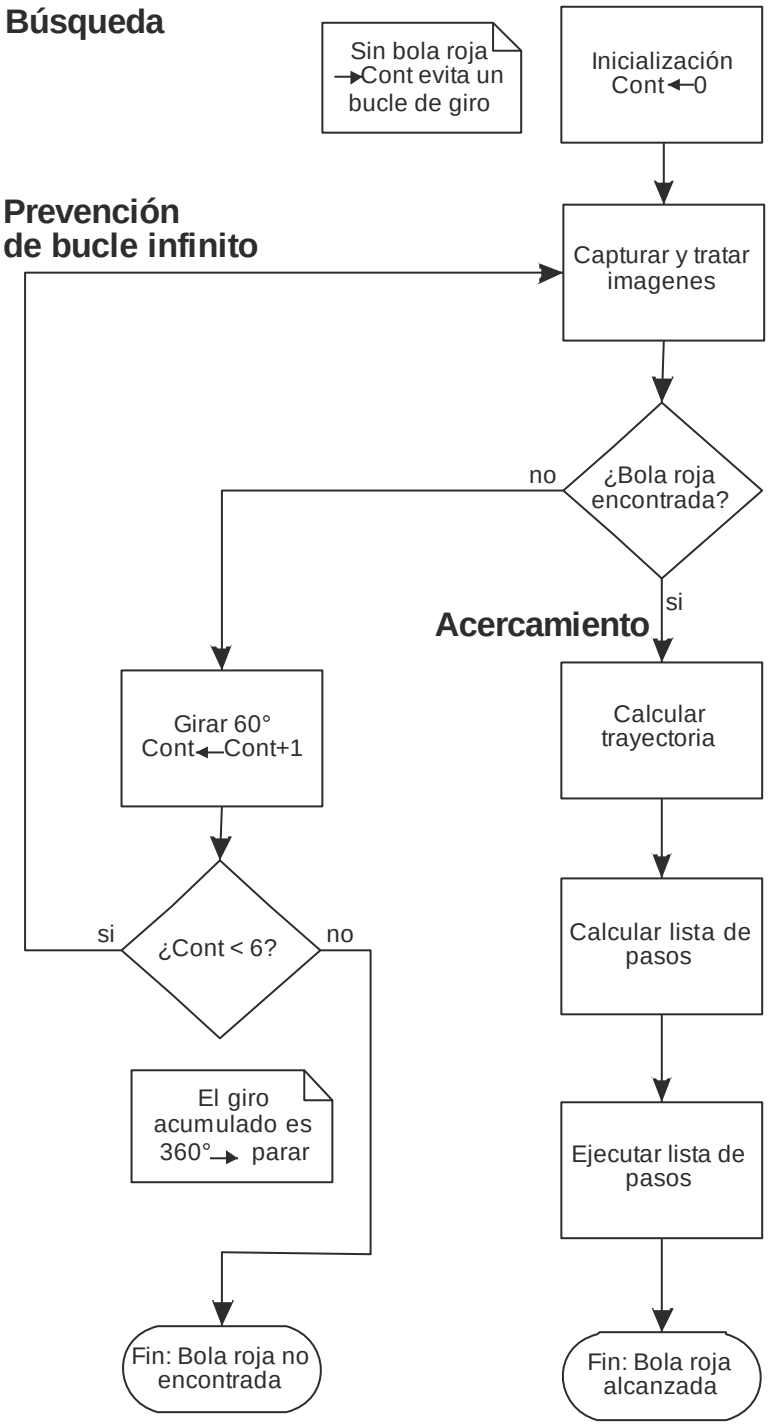

Figura 18. Búsqueda del objetivo

Simular los algoritmos de trayectoria

Utilizando el simulador y los programas open source se realizará una simulación con datos que podría presentarse en situaciones reales para evaluar la performance del algoritmo.

Modo de ejecución: Ambos con visión

Dificultad: Análisis de visión estereoscópica tras cada giro, detección de obstáculos ya analizados antes, detección de obstáculos nuevos y re-planificación del camino en caso que sea necesario.

Otra dificultad a destacar es el sincronismo entre simulación y SRV-1.

Demostración de la evolución al objetivo principal del proyecto. Visión estereoscópica en funcionamiento tras cada giro, lo que permite que podamos trazar cualquier camino de una coordenada a otra sin problema de colisión debido a la detección de obstáculos nuevos en cada giro. Trazaremos un camino, tras cada giro haremos una foto para poder analizar nuevos obstáculos no detectados anteriormente, en el caso de que los haya, re-planificaremos el camino de tal forma que evitemos esos obstáculos hasta llegar a nuestro objetivo.

Tabla 2

Resultados Obtenidos

Resultados de la prueba	
Detalles de ejecución	
Modo de ejecución	Ambos con visión
GPS activado:	No
Algoritmo de detección de obstáculos:	Algoritmo basado en colores
Alcanza el objetivo	Sí
Descripción textual	
Colocamos enfrente del robot dos obstáculos: una bola verde y una roja. La bola verde está encima de una caja que oculta la bola roja al principio. Así el robot, verá primero la bola verde y más tarde la bola roja. Esto hará que replanifique en dos ocasiones. Ponemos la posición objetivo de la ruta detrás del último obstáculo. La posición objetivo la marcamos con una bola azul.	
Datos dela ruta	
Posición origen:	(X: 0, Y: 0)
Rotación Inicial:	179,9853
Posición objetivo:	(X: -420, Y: -20)
Distancia al objetivo (distancia simulador)	420,42
Lista de pasos original	
Paso 1	

N° de acción	**Tipo**	**Cantidad**	**Observación**

1	Girar	2,742004	
2	Consultar giro	-	GPS no activado No consulta
3	Visión	-	Nuevo obstáculo => No Replanifica
4	Avanzar	420,4497	Replanificación => no se ejecuta.
5	Consultar	-	Replanificación => no se ejecuta.
6	Sincronismo	-	Replanificación => no se ejecuta.

Lista de pasos tras la primera replanificación

Paso 1

N° de acción	Tipo	Cantidad	Observación
1	Girar	33,69008	
2	Consultar giro	-	GPS no activado => No consulta
3	Visión		No replanifica
4	Avanzar	115,3777	
5	Consultar	-	GPS no activado => no consulta

Paso 2

N° de acción	Tipo	Cantidad	Observación
1	Girar	-33,69006	
2	Consultar giro	-	GPS no activado => no consulta
3	Visión	-	Nuevo obstáculo => replanifica
4	Avanzar	256,0001	Replanifica => no se ejecuta
5	Consultar	-	

Paso 3			
N° de acción	**Tipo**	**Cantidad**	**Observación**
1	Girar	-44,66165	Replanificación => No se ejecuta
2	Consultar giro	-	Replanificación => No se ejecuta
3	Visión	-	Replanificación => No se ejecuta
4	Avanzar	96,18596	Replanificación => No se ejecuta
5	Consultar	-	Replanificación => No se ejecuta
6	Sincronismo	-	Replanificación => No se ejecuta

De los resultados obtenidos en él, todos los escenarios, podemos decir que, la generación de trayectorias resulta robusta y eficiente en términos de tiempo de cómputo. Además, para la evasión local de obstáculos resulta una alternativa fiable ya que de haber espacio para que el robot pase, el algoritmo, encontrara la ruta de menor costo. Pero, sin embargo, esta ruta tiende a ser muy rectilínea, es decir con aristas pronunciadas, por lo que se hace necesario reducir la velocidad del robot a la mitad (0.1 m s-1) para que el controlador pueda seguir la trayectoria de forma ajustada y evitar así las colisiones en espacios reducidos.

Por otra parte, la evasión de obstáculos resulta exitosa, siempre que la altura del objeto sea menor que la altura a la que se encuentra el sensor Kinect. Esto, debido a que si no ve su punto de destino (punto intermedio de la trayectoria global P), el robot necesita generar una nueva trayectoria al siguiente punto intermedio Pk+2, k+1, lo que no siempre resulta ser la trayectoria de menor costo para evadir un obstáculo.

En cuanto al controlador, este resulta adecuado, ya que considera el modelo cinemático del robot, pero dicha cinemática cambia si es que se le añade peso adicional, como por ejemplo un pequeño estanque con herbicida. Por lo que los parámetros θ deben ser recalculados.

4.4. Resultados

Después de analizada el recorrido de nuestro robot los datos de uno de los algoritmos, a pesar de que el tiempo de respuesta son casi similares, se pudo optar por considerar ambos algoritmos, puesto que las corridas obtenidas van a variar cuando el robot móvil esté construido, es decir el tiempo de respuesta de un servomecanismo tiene un ligero retardo con los resultados de las corridas en laboratorio, provocando así en nuestro móvil una lentitud de comunicación

Luego de analizar estos retrasos, y viendo que nuestro robot móvil dependerá de que nuestro algoritmo realice correctamente su trabajo principal que es el de ubicar nuestro objetivo a buscar

4.5. Prueba estadística

Para la prueba estadística se necesitó realizar algunas operaciones para obtener la información necesaria, las cuales eran calcular los tiempos de ejecución de los dos algoritmos que se estaban usando datos de los caminos (1er y 2do algoritmo) y para otros casos

Uno de los avances más destacados en los últimos años en lo que se refiere al bienestar de las personas en el hogar es la aparición del robot de limpieza. Gracias al trabajo de innumerables ingenieros, se ha conseguido eliminar la necesidad de barrer y aspirar manualmente.

Si analizamos un poco la arquitectura de un robot de limpieza, podemos diferenciar diversas partes que suelen ser comunes en todos ellos:

- El depósito que es el recipiente donde es almacenada toda la suciedad que ha recogido el robot durante el tiempo que ha estado funcionando. En este caso, no suele tratarse de una bolsa como los aspiradores de toda la vida, sino un cajón que se debe de retirar y limpiar cada cierto tiempo.

- La aspiradora cuya función no es otra que la de atrapar el polvo y la suciedad mediante succión al mismo tiempo que el robot se mueve por el suelo.

- El cepillo lateral rotatorio que se encarga de recoger la suciedad que la aspiradora no puede llegar.

- El agitador que se trata de un cepillo situado en la parte baja del dispositivo y que elimina la suciedad más difícil de extraer.

Estos robots pequeños e inteligentes que pueden costar desde 40€ hasta más de 1000€, son capaces de moverse por las diferentes partes de la casa evitando chocar contra muebles o paredes, pero ¿cómo funcionan?

Aunque el comportamiento exacto de dichos aparatos depende del fabricante, lo que hacen, generalmente, es limpiar hasta donde pueden en una dirección hasta que chocan contra un objeto, luego giran y continúan limpiando aleatoriamente por otras partes de la habitación.

Los robots de limpieza utilizan sensores para ayudarles a determinar dónde limpiar, así como qué zonas evitar (una escalera, por ejemplo).

Los robots más inteligentes pueden incluso trazar un mapa virtual y planear una trayectoria óptima de limpieza para así evitar obstáculos.

Algunos lo hacen utilizando cámaras, mientras que otros utilizan láseres. Estos robots detectan reflexiones en el láser para determinar dónde se encuentran los obstáculos en entornos de 360 grados, lo que permite al robot crear un mapa y saber dónde se encuentra dentro de él.

Por otro lado, rastrean lo que han limpiado y lo que les queda por limpiar en cada una de las diferentes partes de la casa.

También algunos cuentan con funciones de detección de suciedad que les permite tomar nota de cuanta suciedad está siendo arrastrada por sus cepillos y así prestar más atención a la limpieza de esas áreas.

Muchos robots también tienen la capacidad de seguir una ruta lógica tomando como referencia una pared. Utilizando este método, el robot continúa limpiando junto a un obstáculo gracias a un sensor infrarrojo para detectar qué tan cerca del obstáculo puede llegar a estar el robot sin chocar contra él.

Sin duda alguna nos encontramos ante uno de los aparatos que más ha revolucionado la limpieza del hogar, principalmente. Quizá nos encontremos ante un elemento

imprescindible para aquellos que son demasiado vagos o simplemente para los que no tienen suficiente tiempo para estar limpiando.

CONCLUSIONES

- El marco de integración de algoritmos de seguimiento y planificación de trayectorias propuesto, muestra bajos errores en el seguimiento de las trayectorias, debido a la optimización de los parámetros del controlador, para cada trayectoria. Es decir, el controlador genera la mejor solución para cada trayectoria.

- Los parámetros del controlador obtenidos mediante algoritmos garantizan una solución casi óptima, ya que evita caer en mínimos locales a diferencia de otros métodos de optimización.

- El ampliamente probado algoritmo de planificación modelo Walk Random y Tree Guarding Walk ha probado su eficacia en la generación de trayectorias tanto locales como globales y su eficiencia en términos de tiempo de cómputo. Lo que hace adecuado para la trayectoria de un robot.

- La evasión local de obstáculos detectados por el sensor Kinect, resulta adecuado para la detección de objetos en el camino del robot, aunque estos sean de pequeño tamaño. Pero se debe tener en cuenta en su implementación en terreno, el efecto de la luz sobre el sensor de profundidad y los efectos de la vibración en la determinación de las alturas de los objetos detectados.

- La herramienta con un algoritmo correcto de reconocimiento de objetos extraños y un sistema de posicionamiento global podría dar lugar a un sistema completo de navegación no tripulado adecuado para cualquier tipo de propósito.

RECOMENDACIONES

- Realizar de manera permanente una evaluación en las áreas involucradas, en el área de logística y el área de infraestructura, debido a que al abastecimiento de materiales un es factor indispensable en la ejecución de obras, con el propósito de optimizar el proceso, implementando mejoras permanentes y oportunas, verificando que cada implementación contribuya eficientemente con el cumplimiento de objetivos en las obras y por ende en la Universidad Nacional del Altiplano.

- Contemplar los cronogramas de ejecución física, durante el periodo de ejecución, optimizando la administración de los recursos asignados que se verán reflejados en la ejecución física y financiera, cumpliendo así con los plazos establecidos de manera eficiente.

- Incidir en los mecanismos de coordinación y comunicación, ya que estos son generadores de mejores resultados, a efectos de mantener y seguir mejorando las acciones de gestión en cuanto a la ejecución de obras por administración directa en la Universidad Nacional del Altiplano.

- Considerar los lineamientos de mejora propuestos en el trabajo de investigación dentro de los planes de gestión previstos por la Universidad Nacional del Altiplano.

BIBLIOGRAFÍA

Bañón, J. (2000) *Sobre algoritmos de planificación del movimiento de robots*, Ingeniería y Competitividad, vol. 2, n.° 2, pp. 18-25.

Barrientos J. (1996) *Fundamentos de Robótica*. Madrid: Universidad Politécnica de Madrid.

Bessonnet, G. Lallemand, J. (1990) *Optimal trajectories of robot arms minimizing constrained actuators and traveling time*, in International Conference on Robotics and Automation. IEEE. Proceedings of the IEEE International. Cincinnati, Ohio.

Chevallereau, C. (1996) *Feasible trajectories for non redundant robot at a singularity*, in International Conference on Robotics and Automation. IEEE. Minnesota.

Choset, H. (2005) P*rinciples of robot motion: theory, algorithms and implementation*. Cambridge: Massachusetts Institute of Technology MIT.

Colin, Kellin Alonzo. (2010) *Toward optimal sampling in the space of paths.* Springer Tracts in Advanced Robotics, 66:281 – 292.

De Luca, A. Oriolo, G. (2000) *Motion Planning and Trajectory Control of an Underactuated Three-Link Robot Vía Dynamic Feedback Linearization*, in International Conference on Robotics and Automation. IEEE. San Francisco, California.

González, J. (2004) *Control bilateral con retroalimentación de esfuerzos aplicado a la cirugía cardiaca*, Tesis de Doctorado en Computación Avanzada para Ciencias e Ingenierías. Universidad Politécnica de Madrid, España.

Harada, K. (2012) *Pick and place planning for dual-arm manipulators*, in International Conference on Robotics and Automation. IEEE. Minnesota.

Huang, G. (2011) *Inverse kinematics analysis trajectory planning for a robot arm*, in Asian Control Conference ASCC. Kaohsiung, Taiwán.

Ismael, A. (2009) *Robot trajectory planning with semi-infinite programming*, European Journal of Operational Research, vol. 153, n.° 1, pp. 607-617.

Kubota, N. (1998) *Hierarchical trajectory planning of redundant manipulators with structured intelligence*", Advanced Robotics Journal, vol. 12, n.° 3, pp. 209-225.

Kunz, T. (2010) *Time path planning for a robot arm in changing enviroments*, in International Conference on Intelligent Robots and Systems. IEEE. Taipei.

Leng, D. Chen, M. (1997) *Robot trajectory planning using simulation*", Robotics & Computer-Integrated Manufacturing Journal, vol. 13, n.° 2, pp. 121-129.

Lin, C. (2004) *Motion plannning of redundant robots by perturbation method*, Mechatronics Journal, vol. 14, n.° 3, pp. 237-339.

Mitsi, S. (2005) *Off-line programming of an industrial robot for manufacturing*, International Journal of Advanced Manufacturing Technology, vol. 26, n.° 5, pp. 262-267.

Pan, Z. (2012) R*ecent progress on programming methods for industrial robots*, Robotics and Computer-Integrated Manufacturing, vol. 28, n.° 1, pp. 87-94.

Ramabalan, S. (2008) *Multi-objetive dynamic optimal trajectory planning of robot manipulators in the presence of obstacles*, International Journal of Advances in Manufacturing Technology, vol. 41, n.° 5, pp. 580-594.

Reeves, C. Wright, C. (1995) *Genetic algorithms and statistical methods: a comparison*, in International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications. Galesia. Sheffield.

Shin, Y. Bien, Z. (1998) *A novel method ofcollision-free trajectory planning for two robot arms*", in International Conference on Systems, Man, and Cybernetics. IEEE. Seul. Conference on Robotics and Automation. IEEE. Nagoya, 1995.

Tsuji, T. (1997) *Distributed trajectory generation for coopertive multi-arm robots via virtual force interactions*, IEEE transactions on systems, man, and cybernetics Journal, vol. 27, n.° 5, pp. 862-867.

Ubillos, J., Xu, M., Ming, Z., & Vogt, C. (2011). Name-Based Sockets Architecture draft-ubillos-name-based-sockets-03. *IETF*. Obtenido de http://tools.ietf.org/pdf/draft-ubillos-name-based-sockets-03.pdf

Ubillos., J. (2011). namebasedsockets (Name-based sockets). *Online*. Obtenido de https://github.com/namebasedsockets

UCLouvain. (2017). *Multipath TCP - Linux Kernel Implementation*. Retrieved from http://multipath-tcp.org

Universidad Nacional del Altiplano de Puno. (2017 - 2019). *Plan Estratégico Institucional*. Puno: UNA.

Vera., M. (2012). *Evolución histórica del aprovisionamiento de bienes y servicios de la facultad de medicina de la UNMSM*. Lima: Universidad Nacional Mayor de San Marcos.

Vidal, J. (2017). *El Control Simultáneo como Nuevo Enfoque de la Auditoría Gubernamental, en la Gestión de Proyectos de Inversión Pública, en la Gerencia Regional de Infraestructura del Gobierno Regional del Callao.* Lima: Universidad Inca Garcilaso de la Vega.

Wang, F. (2010) *Genetic algorithms with a new repair operator for assembly job shop scheduling*", International Journal of Industrial Engineering, vol. 18, n.° 7, pp. 377-385.

Wunderlich, J. (2004) *Simulating a robotic arm in a box: redundant kinematics, path planning, and rapid prototyping for enclosed spaces*, Simulation: Journal of Society for Computer Simulation, vol. 80, n.° 6, pp. 301-316.

Xiaoshu, J. Xichen, Y. (2009) O*ff-line programming of a robot for laser re-manufacturing*, Tsinghua Science and Technology Journal, vol. 14, n.° 1, pp. 186-191.

Xu, W. (2009) *Study on non-holonomic cartesian path planning of free-floating space robotic system*, Advanced Robotics Journal, vol. 23, n.° 1-2, pp. 113-143.

Yahya, S. (2009) *A geometrical motion planning approach for redundant plannar manipulators*, Australian Journal of Basic and Applied Sciences, vol. 3, n.° 4, pp. 3757-3770.

Yao, Z. Gupta, K. (2007) *Path planning with general end-effector constrains*, Robotics And Autonomous Systems, vol. 55, n.° 1, pp. 316-327.

Zha, X. Chen, X. (2004) *Trajectory coordination planning and control for robot manipulators in automated material handling and procesing*", International Journal of Advanced Manufacturing Technology, vol. 23, n.° 2, 831-845.

Zhenyu, L.(2009) *Motion navigation for arc welding robots based on feature mapping in a simulation enviroment*, Robotic And Computer-Integrated Manufacturing Journal, vol. 26, n.° 1, pp. 137-144.

ANEXOS

ANEXO 1. APLICACIONES Y AVANCE TECNOLÓGICO: LOS ROBOTS DE LIMPIEZA

Uno de los avances más destacados en los últimos años en lo que se refiere al **bienestar** de las personas en el hogar es la aparición del robot de limpieza. Gracias al trabajo de innumerables ingenieros, se ha conseguido eliminar la necesidad de barrer y aspirar manualmente.

Si analizamos un poco la arquitectura de un robot de limpieza, podemos diferenciar **diversas partes** que suelen ser comunes en todos ellos:

- El **depósito** que es el recipiente donde es almacenada toda la suciedad que ha recogido el robot durante el tiempo que ha estado funcionando. En este caso, no suele tratarse de una bolsa como los aspiradores de toda la vida, sino un cajón que se debe de retirar y limpiar cada cierto tiempo.
- La **aspiradora** cuya función no es otra que la de atrapar el polvo y la suciedad mediante succión al mismo tiempo que el robot se mueve por el suelo.
- El **cepillo lateral rotatorio** que se encarga de recoger la suciedad que la aspiradora no puede llegar.
- El **agitador** que se trata de un cepillo situado en la parte baja del dispositivo y que elimina la suciedad más difícil de extraer.

Estos robots pequeños e inteligentes que pueden costar desde 40€ hasta más de 1000€, son capaces de moverse por las diferentes partes de la casa **evitando chocar** contra muebles o paredes, pero **¿cómo funcionan?**

Aunque el comportamiento exacto de dichos aparatos depende del fabricante, lo que hacen, generalmente, es limpiar hasta donde pueden en **una dirección** hasta

que chocan contra un objeto, luego giran y continúan limpiando aleatoriamente por otras partes de la habitación.

Los robots de limpieza **utilizan sensores** para ayudarles a determinar dónde limpiar, así como qué zonas evitar (una escalera, por ejemplo).

Los robots más inteligentes pueden incluso trazar **un mapa virtual** y planear una trayectoria óptima de limpieza para así evitar obstáculos.

Algunos lo hacen utilizando **cámaras**, mientras que otros utilizan láseres. Estos robots detectan reflexiones en el láser para determinar dónde se encuentran los obstáculos en entornos de 360 grados, lo que permite al robot crear un mapa y saber dónde se encuentra dentro de él.

Por otro lado, rastrean lo que han limpiado y lo que les queda por limpiar en cada una de las diferentes partes de la casa.

También algunos cuentan con funciones de **detección de suciedad** que les permite tomar nota de cuanta suciedad está siendo arrastrada por sus cepillos y así prestar más atención a la limpieza de esas áreas.

Muchos robots también tienen la capacidad de seguir una ruta lógica tomando como referencia una pared. Utilizando este método, el robot continúa limpiando junto a un obstáculo gracias a un **sensor infrarrojo** para detectar qué tan cerca del obstáculo puede llegar a estar el robot sin chocar contra él.

Sin duda alguna nos encontramos ante uno de los aparatos que más **ha revolucionado la limpieza del hogar**, principalmente. Quizá nos encontremos ante un elemento imprescindible para aquellos que son demasiado vagos o simplemente para los que no tienen suficiente tiempo para estar limpiando.

ANEXO 2. CAPTURA DE LA PROGRAMACIÓN DEL ALGORITMO MODEL WALK RAMDOM

```
# CAMINOS ALEATORIOS
# Model Walk Random
import random
from time import time
def random_walk(n):
    """RETORNA LAS COORDENADAS DESPUES DE n NUMERO DE CAMINOS ALEATORIOS """
    x=0
    y=0
    for i in range(n):
        step=random.choice(['N','S','E','O'])
        if step == 'N':
            y=y+1
        elif step =='S':
            y = y-1
        elif step == 'E':
            x=x+1
        else:
            x=x-1
    return (x,y)

timesi = 0
for i in range(100):
    start_time = time()
    walk = random_walk(10)
    elapsed_time = time() - start_time
    print(i, "::", walk, "Siguiente Coordenada = ", abs(walk[0])+abs(walk[1]), " / ", elapsed_time)
    timesi = elapsed_time + timesi
print("Tiempo de Recorrido time: %.10f seconds." % timesi)
```

ANEXO 3. CAPTURA DE LA PROGRAMACIÓN DEL ALGORITMO MODEL TREEGUARDING WALK

```
# CAMINOS ALEATORIOS
# Model TreeGuarding Walk
import random
from time import time

def walkit(x,y):
    step=random.choice(['N','S','E','O'])
    if step == 'N':
        y=y+1
    elif step =='S':
        y = y-1
    elif step == 'E':
        x=x+1
    else:
        x=x-1
    return (x,y)

timesi = 0
#Obstaculos
obs= [(2,3),(4,7),(2,5)]
#Punto de Inicio de Recorrido
xi=0
yi=0
#Punto final de recorrido
xf=500
yf=100

for i in range(100):
    # Iniciar Camino
    walk = walkit(xi,yi)
    i=1
    while walk[0]!=xf & walk[1]!=yf:
        start_time = time()
        walk = walkit(walk[0],walk[1])
        print(i, "::", walk, "Siguiente Coordenada = ", abs(walk[0])+abs(walk[1]))
        elapsed_time = time() - start_time
        timesi = elapsed_time + timesi
        i=i+1
print("Tiempo de Recorrido time: %.10f seconds." % timesi)
```

ANEXO 4. CODIGO FUENTE MODELO WALK RAMDOM

CASO 1:

Código fuente del algoritmo **Model Walk Random,** no se aprecia bien por lo que se ha pegado del Python (Es el programa usado para correr el algoritmo y simularlo)

```
# CAMINOS ALEATORIOS
# Model Walk Random
import random
from time import time
def random_walk(n):
    """RETORNA LAS COORDENADAS DESPUES DE n NUMERO DE CAMINOS ALEATORIOS """
    x=0
    y=0
    for i in range(n):
        step=random.choice(['N','S','E','O'])
        if step == 'N':
            y=y+1
        elif step =='S':
            y = y-1
        elif step == 'E':
            x=x+1
        else:
            x=x-1
    return (x,y)

timesi = 0
for i in range(100):
    start_time = time()
    walk = random_walk(10)
```

```
    elapsed_time = time() - start_time

    print(i, "::", walk, "Siguiente Coordenada = ", abs(walk[0])+abs(walk[1]), " / ",
elapsed_time)

    timesi = elapsed_time + timesi

print("Tiempo de Recorrido time: %.10f seconds." % timesi)
```

ANEXO 5. CODIGO FUENTE MODELO TREEGUARDING WALK

CASO 2,

El segundo código fuente del algoritmo **Model TreeGuarding Walk** al igual que el anterior a continuación se presenta el código fuente de Python

```
# CAMINOS ALEATORIOS
# Model TreeGuarding Walk
import random
from time import time

def walkit(x,y):
    step=random.choice(['N','S','E','O'])
    if step == 'N':
        y=y+1
    elif step =='S':
        y = y-1
    elif step == 'E':
        x=x+1
    else:
        x=x-1
    return (x,y)

timesi = 0
#Obstaculos
obs= [(2,3),(4,7),(2,5)]
#Punto de Inicio de Recorrido
xi=0
yi=0
#Punto final de recorrido
xf=500
```

```
yf=100


for i in range(100):
    # Iniciar Camino
    walk = walkit(xi,yi)
    i=1
    while walk[0]!=xf & walk[1]!=yf:
        start_time = time()
        walk = walkit(walk[0],walk[1])
        print(i, "::", walk, "Siguiente Coordenada = ", abs(walk[0])+abs(walk[1]))
        elapsed_time = time() - start_time
        timesi = elapsed_time + timesi
        i=i+1
print("Tiempo de Recorrido time: %.10f seconds." % timesi)
```