ESTUDIO COMPARATIVO ENTRE MÉTODOS QUE SOLUCIONAN EL PROBLEMA DE PLANEACIÓN DE TRAYECTORIAS EN ROBÓTICA MÓVIL

EDWIN DAVID ERAZO CAICEDO

JESÚS ALBERTO GETIAL BARRAGÁN

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2018

ESTUDIO COMPARATIVO ENTRE MÉTODOS QUE SOLUCIONAN EL PROBLEMA DE PLANEACIÓN DE TRAYECTORIAS EN ROBÓTICA MÓVIL

EDWIN DAVID ERAZO CAICEDO

JESÚS ALBERTO GETIAL BARRAGÁN

TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE

INGENIERO ELECTRÓNICO

DIRECTOR

ANDRES DARIO PANTOJA BUCHELI. PhD

INGENIERO ELECTRÓNICO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2018

NOTA DE RESPONSABILIDAD

“La Universidad de Nariño no se hace responsable por las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor.”

Acuerdo 1. Artículo 324. Octubre 11 de 1966, emanado del honorable Consejo Directivo de la Universidad de Nariño.

NOTA DE ACEPTACIÓN:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

San Juan de Pasto, 5 de abril de 2018

AGRADECIMIENTOS

Un grato reconocimiento y agradecimiento para:

- Doctor Andrés Dario Pantoja. PhD, por su apoyo incondicional y sus sabias decisiones y orientaciones, claves para el desarrollo de este proyecto.
- Docentes y representantes de la facultad de ingeniería electrónica por su ayuda y colaboración en actividades relacionadas con esta investigación.
- A ParqueSoft Pasto, por el préstamo de equipos e instalaciones para el desarrollo de la etapa de implementación.

DEDICATORIA

A Dios por su sabiduría, sus enseñanzas y todo su amor manifiesto al darme la vida a diario.

A mis padres, por su esfuerzo y dedicación por educarme y sentar las bases sobre las que he edificado mi vida.

A Erika, por todo su apoyo y motivación.

_____

Jesus Alberto Getial Barragán.

A la Universidad de Nariño por su inmensa vocación en el desarrollo de nuestra región, manténgase su espíritu transformador y crítico siempre en sus estudiantes y sus maestros.

A mis familiares y amigos por el apoyo en el tránsito de mi vida universitaria, sus consejos y las enormes enseñanzas brindadas.

A mi madre Patricia Caicedo, no puede ser este logro más que un reflejo de su esfuerzo y su cariño.

_____

Edwin David Erazo Caicedo.

RESUMEN

En este trabajo se presenta una metodología para comparar diferentes técnicas que resuelven el problema de planeación de trayectoria con base en resultados de validaciones virtuales y experimental, considerando uno y múltiples robots en escenarios de navegación controlados. Se considera el estudio de restricciones no holonómicas a partir de interpoladores Spline, BSpline y un control de movimiento lineal para la navegación de un robot móvil de tracción diferencial. Además, se muestra una aproximación para la extensión de la formulación inicial del algoritmo de campos de potencial artificial y descomposición en celdas trapezoidales, al caso de múltiples robots cooperativos para la navegación con evitación de colisión.

Para ello se evalúa el comportamiento de las trayectorias generadas en tres escenarios de estudio según índices enfocados en el resultado de la curva que guía al móvil de un punto a otro. Se mide parámetros como tiempo y distancia recorrida tras la implementación en tiempo casi real de una de las técnicas, para así lograr un compendio teórico-práctico que permite determinar las limitaciones de cada uno de los métodos analizados. De esta manera el desarrollo de esta tesis sirve como guía en la escogencia de entre una técnica u otra para resolver problemas particulares y concluye proponiendo trabajo futuro en el área.

ABSTRACT

This work presents a methodology to compare different techniques that solve the trajectory-planning problem based on virtual and experimental validation results, considering a single and multiple robots in controlled navigation scenarios. It considers the study of non-holonomic constraints from Spline-BSpline interpolators, and a linear motion control for the navigation of a differential traction mobile robot. In addition, it shows an approximation for the extension of the initial formulation for artificial potential fields algorithm and trapezoidal cells decomposition to the multiple cooperative robots case for navigation with collision avoidance.

To do this, it evaluates the trajectories behavior generated in three study scenarios according to indexes focused on the curve result that guides the mobile from one point to another. Measure parameters such as time and distance traveled after the implementation in quasi real time of one of the techniques to achieve a theoretical-practical compendium that allows to determine the limitations of each one of the methods analyzed. In this way, the development of this thesis serves as a guide in the choice of one technique or another to solve particular problems and concludes proposing future work in the area.

CONTENIDO

1. INTRODUCCIÓN 19

2. CONCEPTOS BÁSICOS DE PLANEACIÓN DE TRAYECTORIAS EN ROBÓTICA MÓVIL 25

2.1. ROBÓTICA MÓVIL TERRESTRE 25
2.1.1. INTRODUCCIÓN 25
2.1.2. DEFINICIONES Y NOTACIÓN 25
2.1.3. SISTEMAS DE LOCOMOCIÓN DE ROBOTS MÓVILES 27
2.1.4. NAVEGACIÓN. 27
2.1.5. RESTRICCIONES CINEMÁTICAS 28
2.1.6. CINEMÁTICA DE UN ROBOT MÓVIL 29
2.1.7. GENERACIÓN DE TRAYECTORIAS 34
2.1.8. CONTROL CINEMÁTICO 40
2.2. PLANIFICACIÓN DE TRAYECTORIAS 44
2.2.1. INTRODUCCIÓN 44
2.2.2. MÉTODO DE CAMPOS POTENCIALES ARTIFICIALES. (PF) 44
2.2.3. MÉTODO DE DESCOMPOSICIÓN POR CELDAS TRAPEZOIDALES (CD) 48
2.2.4. MÉTODO DE BÚSQUEDA ALEATORIA. (RRT*) 50
2.2.5. MÉTODO DE OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS (PSO) 51
2.3. PLANEACIÓN DE TRAYECTORIAS PARA MÚLTIPLES ROBOTS 54
2.3.1. INTRODUCCIÓN 54
2.3.2. EXTENSIÓN MÉTODO CAMPOS POTENCIALES ARTIFICIALES 56
2.3.3. EXTENSIÓN MÉTODO DE DESCOMPOSICIÓN POR CELDAS TRAPEZOIDALES 57
2.3.4. OBSTÁCULOS DE VELOCIDAD RECÍPROCA (RVO) 59

3. PROCEDIMIENTOS PARA IMPLEMENTACIÓN DE ALGORITMOS 63

3.1. METODOLOGÍAS PROGRAMADAS PARA PLANEACIÓN DE TRAYECTORIAS 63
3.1.1. FUNCIÓN DE NAVEGACIÓN PARA CAMPOS POTENCIALES ARTIFICIALES CASO ÚNICO ROBOT 63
3.1.2. OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS 64
3.1.3. EXPLORACIÓN RÁPIDA DE ÁRBOL ALEATORIOS* 66
3.1.4. DESCOMPOSICIÓN POR CELDAS TRAPEZOIDALES CASO ÚNICO ROBOT 66
3.1.5. FUNCIÓN DE NAVEGACIÓN PARA CAMPOS POTENCIALES ARTIFICIALES. EXTENSIÓN CASO MÚLTIPLES ROBOTS 76
3.1.6. DESCOMPOSICIÓN POR CELDAS TRAPEZOIDALES. EXTENSIÓN CASO MÚLTIPLES ROBOTS 79
3.1.7. OBSTÁCULOS DE VELOCIDAD RECÍPROCA (RVO) 85
3.2. MARCO EXPERIMENTAL 86

3.2.1. DISEÑO DE EXPERIMENTOS EN PLATAFORMA VIRTUAL 86
3.2.2. DISEÑO DE EXPERIMENTOS EN PLATAFORMA FÍSICA REAL 86
3.2.3. PLATAFORMA "ARDUINO-ROBOT" 86
3.2.4. MÓDULOS XBEE PRO S3 89
3.2.5. POSICIONAMIENTO A PARTIR DE VISIÓN POR COMPUTADOR 89

4. METODOLOGÍA DE COMPARACIÓN ANALÍTICA Y ANÁLISIS DE RESULTADOS 91

4.1. INTRODUCCIÓN 91
4.2. METODOLOGÍA DE COMPARACIÓN 92
4.2.1. ÍNDICES CASO DE UN ÚNICO ROBOT 92
4.2.2. PROCEDIMIENTOS PARA LA ADQUISICIÓN DE MUESTRAS 94
4.2.3. PROCEDIMIENTO PARA EL ANÁLISIS DE MUESTRAS 97
4.3. APROXIMACIÓN AL TRABAJO FUTURO 98
4.3.1. CONTROL DE MOVIMIENTOS 98
4.3.2. GENERACIÓN DE TRAYECTORIAS 98
4.3.3. IMPLEMENTACIÓN EN PLATAFORMA FÍSICA 99
4.3.4. ÍNDICES CASO DE MÚLTIPLES ROBOTS 99

5. RESULTADOS 100

5.1. METODOLOGÍA DE COMPARACIÓN ANALÍTICA 100
5.1.1. RELACIÓN ENTRE PARÁMETROS E ÍNDICES 100
5.1.2. ESTUDIO DE COMPARACIÓN 126
5.2. APROXIMACIÓN AL TRABAJO FUTURO 130
5.2.1. CONTROL DE MOVIMIENTOS. 130
5.2.2. GENERACIÓN DE TRAYECTORIAS 131
5.2.3. IMPLEMENTACIÓN PLATAFORMA FÍSICA 135
5.2.4. CASO DE MÚLTIPLES ROBOTS – AMBIENTE DINÁMICO 137

6. CONCLUSIONES 141

7. TRABAJO FUTURO 142

BIBLIOGRAFÍA 143

ANEXOS 150

LISTA DE TABLAS

Tabla 1 Valor numérico de los parámetros. Extensión de PF para múltiples robots. 79
Tabla 2. Valor numérico de parámetros. Extensión de CD para múltiples robots..84
Tabla 3. Valor numérico de parámetros. RVO.....86
Tabla 4. Velocidad angular en rad/s para un valor de pwm.....88
Tabla 5. Valor numérico de parámetros. PF.....95
Tabla 6 Valor numérico de parámetros. PSO.....96
Tabla 7 Valor numérico de constantes. RRT*.....96
Tabla 8 Valor numérico de constantes. CD 97
Tabla 9. Variables de ajuste regresión polinomial para función de navegación por gradiente descendiente escenario trampa. 104
Tabla 10. Variables de ajuste regresión polinomial para función de navegación por gradiente descendiente escenario pasaje estrecho. 104
Tabla 11. Variables de ajuste regresión polinomial para función de navegación por gradiente descendiente escenario arreglo de obstáculos. 105
Tabla 12. Valores de media, σ y longitud mínima y máxima para distancias de campo local por gradiente descendiente. 106
Tabla 13. Valores de media, σ y porcentaje mínimo y máximo para distancias de función de navegación. Función de navegación para gradiente descendiente. ...106
Tabla 14. Variables de ajuste regresión polinomial para función de navegación PSO escenario arreglo trampa. 111
Tabla 15. Variables de ajuste regresión polinomial para función de navegación PSO escenario pasaje estrecho. 112
Tabla 16. Variables de ajuste regresión polinomial para función de navegación PSO escenario arreglo de obstáculos. 112
Tabla 17. Valores de media, σ y longitud mínima y máxima para distancias de campo local para PSO, escenario trampa. 113
Tabla 18. Valores de media, σ y longitud mínima y máxima para distancias de campo local para PSO, escenario arreglo de obstáculos. 114
Tabla 19. Valores de media, σ y % mínimo y máximo para distancias de función de navegación para PSO, escenario pasaje estrecho. 115
Tabla 20. Valores de media, σ y % mínimo y máximo para distancias de función de navegación para PSO, escenario arreglo. 115
Tabla 21. Valores de $\boldsymbol{Nodmax}$ que determinan aciertos y fracasos en la planeación 118
Tabla 22. Variables de ajuste regresión polinomial para función escenario trampa en RRT*. 119
Tabla 23. Variables de ajuste regresión polinomial para función escenario pasaje estrecho en RRT*. 120
Tabla 24. Variables de ajuste regresión polinomial para función escenario arreglo de obstáculos en RRT*. 120

Tabla 25. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. RRT* escenario trampa. 121
Tabla 26. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. RRT* escenario pasaje estrecho. 121
Tabla 27. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. RRT* escenario arreglo de obstáculos. 122
Tabla 28. Variables de ajuste regresión lineal en CD. 125
Tabla 29. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. CD. 126
Tabla 30. Resumen valores de regresión polinomial de las muestras. 127
Tabla 31. Longitud de distancia original y por interpolaciones. Escenario trampa. 132
Tabla 32. Longitud de distancia original y por interpolaciones. Escenario pasaje estrecho. 133
Tabla 33. Longitud de distancia original y por interpolaciones. Escenario pasaje estrecho. 134
Tabla 34. Índices de comparación implementación. 136
Tabla 35. Índices de comparación múltiples robots. 137

LISTA DE FIGURAS

Figura 1. Esquemas de locomoción terrestre. 27
Figura 2. Esquemas de navegación automática de robots. 28
Figura 3. Vehículos con restricciones holonómicas y no holonómicas. 29
Figura 4. Coordenadas para especificar posición y orientación de una rueda en el plano. 30
Figura 5. Sistema de referencia en navegación de robots móviles. 31
Figura 6. Contacto osculador. 31
Figura 7. Velocidades de vehículo de tracción diferencial respecto centro instantáneo de rotación CIR. 34
Figura 8. Una interpolación de los vértices de un polígono de control V, por el método Spline. 36
Figura 9. Variaciones de la interpolación de los vértices de un polígono de control V, por el método B-Spline, variando en orden k. 39
Figura 10. Vectores gradiente para el campo de potencial atractivo y repulsivo. 44
Figura 11. Funciones de atracción para objetivo en coordenada (5,5). 45
Figura 12. Función de navegación con cinco obstáculos estáticos. 47
Figura 13. Definición de los objetos y el espacio de trabajo en CD. 48
Figura 14. Descomposición trapezoidal, mapa de conexiones y definición de un segmento de línea. 49
Figura 15. Esquema de expansión de RRT. 50
Figura 16. Ruta planeada por método RRT* 51
Figura 17. Ruta planeada por método PSO. 54
Figura 18. Planificación de trayectoria para robot $r1$ en un campo de potencial variable debido al robot $r2$. 56
Figura 19. Ejemplo en el que el robot $r2$, es tomado como un objeto más en la descomposición trapezoidal. 58
Figura 20. Descomposición del grafo para el Objeto Camino, referente al espacio tomado por $r2$, al desplazarse por una trayectoria. 58
Figura 21. Objetos Camino que se traslapan. 59
Figura 22. Representación geométrica del obstáculo de velocidad del robot A impuesto por el robot B. 60
Figura 23. Ruta planeada por método RVO para tres robots homogéneos de tracción diferencial. 62
Figura 24. Representación de escenarios de prueba para el método de campos potenciales artificiales. 64
Figura 25. Representación de escenarios de prueba para método PSO 65
Figura 26. Representación de escenarios de prueba para el método RRT*. 66
Figura 27. Estructura de búsqueda para el mapa trapezoidal, generado por los segmentos de línea $s1$ y $s2$ 67
Figura 28. Definición del trapezoide y cuatro de los cinco casos posibles para el punto left(Δ) en el caso general. 69
Figura 29. Grafos de posición general y no general. 70

Figura 30. Ejemplo para la identificación de trapezoides afectados por un segmento de línea y la creación posterior del nuevo mapa trapezoidal. 71
Figura 31. Ejemplos de la modificación de la estructura de búsqueda. 74
Figura 32. Representación de escenarios de prueba para el método CD.....76
Figura 33. Fuerza de repulsión que $r1$ experimenta al entrar al campo generado por $r2$. 77
Figura 34. Representación del escenario de navegación para robot 1,2 y 3 78
Figura 35. Nodos creados por el Objeto Camino generado por $r1$ cuando no se toma en cuenta su forma, una trayectoria generada para el robot $r2$ en el nuevo mapa de ruta y la formación de un nuevo Objeto Camino debido a esta.....80
Figura 36. Mapa de ruta generado para la alternativa de añadir únicamente 2 nodos en las extensiones verticales. 81
Figura 37. Ejemplo de formación de Objetos Camino y zonas de intersección con el método simplificado. 82
Figura 38. Robot $r1$ entrando al campo generado por $r2$ y su interacción con el mismo. 83
Figura 39. Representación de nodos para el escenario arreglo de obstáculos modificado para extensión de CD. 84
Figura 40. Representación del escenario de navegación arreglo de obstáculos modificado para el método RVO 85
Figura 41. Arduino-Robot. 87
Figura 42. Modelo matemático motor DC que posee el Arduino-Robot. 88
Figura 43. Conexiones y disposición final de la plataforma robótica. 89
Figura 44. Imagen segmentada aplicando K-medias e identificador de entrada para la selección de grupo representativo que caracteriza al robot. 90
Figura 45. Rutas planeadas dependiendo de k. Función de navegación para gradiente descendiente.....100
Figura 46. Rutas dependido de λ y ξ. Campo local para gradiente descendiente. 102
Figura 47. Longitud total del camino dependiendo de k para función de navegación. Gradiente descendiente. 103
Figura 48. Longitud de ruta dependiendo de λ y ξ en campo local. Gradiente descendiente.....105
Figura 49. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. Función de navegación para gradiente descendiente.....107
Figura 50. Rutas dependiendo de k. Función de navegación para PSO.....108
Figura 51. Rutas dependido de λ y ξ. Campo local para PSO 109
Figura 52. Longitud total del camino dependiendo de k. Función de navegación para PSO. 111
Figura 53. Longitud de ruta dependiendo de λ y ξ. Campo local para PSO. 113
Figura 54. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. Función de navegación para PSO. 114
Figura 55. Rutas generadas para variaciones de número de partículas y radio de dispersión. Escenario trampa PSO 117

Figura 56. Rutas variando el número de nodos 118
Figura 57. Longitud total del camino dependiendo de $Nodmax$. RRT* 119
Figura 58. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. RRT*. 121
Figura 59. Variación de parámetros caso particular RRT*. 123
Figura 60. Rutas variando la forma de los obstáculos en CD. 124
Figura 61. Longitud de ruta para variaciones de forma de los obstáculos. CD. ..124
Figura 62. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. CD.....125
Figura 63. Resultados de control aplicado al robot de tracción diferencial. 130
Figura 64. Rutas planeadas en escenario tipo trampa e interpolaciones Spline y BSpline 132
Figura 65. Rutas planeadas en escenario tipo pasaje estrecho e interpolaciones Spline y BSpline.....133
Figura 66. Rutas planeadas en escenario tipo arreglo de obstáculos e interpolaciones Spline y BSpline. 134
Figura 67. Resultados de planeación de trayectoria implementando PF. 136
Figura 68. Rutas planeadas para múltiples robots en un escenario de navegación controlado. 138

LISTA DE ANEXOS

ANEXO 1. CÁLCULO DE GRADIENTE DESCENDIENTE……..150

ANEXO 2. PSEUDO-CÓDIGO ALGORITMO A*……150

ANEXO 3. PSEUDO-CÓDIGO ALGORITMO DIJKSTRA……151

ANEXO 4. PSEUDO-CÓDIGO OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS (PSO) Y COLONIA DE ABEJAS (ABC)……152

ANEXO 5. PSEUDO-CÓDIGO AGRUPAMIENTO K-MEDIAS……153

ANEXO 6. ALGORITMO PARA RASTERO DEL ROBOT ……..153

ANEXO 7. ARTÍCULO CCAC 2017……..……155

ANEXO 8. PÁGINA WEB……...162

GLOSARIO

ABC: abreviatura del algoritmo de colonia artificial de abejas cuyo término en inglés es: "*Artifitial Bee Colony".*

AGENTE: entidad física o virtual, capaz de percibir su entorno, procesarlo y actuar de manera racional, buscando optimizar el resultado.

CAMPO: es un campo vectorial que refiere a la fuerza eléctrica por unidad de carga.

CD: abreviatura del algoritmo de descomposición por celdas cuyo término en inglés es: "*Cell Decomposition".*

GRADIENTE: el vector gradiente indica la dirección en la cual el campo de una función varía más rápidamente y su módulo representa el ritmo de variación de la función en la dirección de dicho vector gradiente.

GRAFO: dato abstracto que consiste en un conjunto de vértices y un conjunto de aristas que establecen relaciones entre los nodos.

ORCA: abreviatura del algoritmo de evitación óptima de colisión recíproca cuyo término en inglés es: "*Optimal Reciprocal Collision Avoidance*".

PF: abreviatura del algoritmo de función de planificación cuyo término en inglés es: "*Planning Function".*

PRM*: abreviatura del algoritmo modificación de PRM llamado mapas de ruta probabilístico asterisco cuyo término en inglés es: "*Probabilistic RoadMaps*".*

PRM: abreviatura del algoritmo mapas de ruta probabilísticos cuyo término en inglés es: "*Probabilistic RoadMaps".*

PSO: abreviatura del algoritmo de optimización de enjambre de partículas cuyo término en inglés es: "*Particle Swarm Optimization".*

PWM: abreviatura de modulación por ancho de pulso cuyo término en inglés es: "*Pulse-Width Modulation".*

RESTRICCIÓN HOLONÓMICA: aquella en las que no interviene la velocidad y ocurre cuando los grados de libertad del robot están desacoplados. Si el número de grados de libertad controlables son todos los del robot, este es un robot holónomo.

RESTRICCIÓN NO HOLONÓMICA: aquella que depende de la velocidad y demanda de una trayectoria integrable que no puede deducirse derivando una restricción holónoma. Si el número de grados de libertad controlables no son todos los del robot, este es un robot no holónomo.

RRT*: abreviatura del algoritmo modificación de RRT llamado árboles de exploración rápida asterisco cuyo término en inglés es: "*Rapidly-exploring RandomTrees**".

RRT: abreviatura del algoritmo de árboles aleatorios de exploración rápida cuyo término en inglés es: "*Rapidly-exploring RandomTrees".*

RVO: abreviatura del algoritmo de obstáculos de velocidad recíproca cuyo término en inglés es: "*Reciprocal Velocity Obstacles".*

TRAPEZOIDE: figura geométrica de cuatro lados, de los cuales no hay ninguno paralelo a otro.

VO: abreviatura de obstáculo de velocidad cuyo término en inglés es: "*Velocity Obstacle".*

1. INTRODUCCIÓN

El estudio de planeación de trayectorias es de gran importancia dentro de la robótica móvil, ya que es un componente indispensable dentro de la navegación autónoma, no solo de vehículos terrestres, sino de todo tipo de robot que requiera realizar movimiento controlado para desarrollar una tarea, como por ejemplo los manipuladores robóticos, los robots aéreos, etc.

La planificación de trayectorias define cómo lograr que un robot móvil encuentre una ruta óptima y libre de obstáculos, desde un punto inicial, hasta un punto deseado, en diferentes escenarios. La trayectoria debe definirse mediante puntos de control que han de seguirse considerando la cinemática del sistema, el entorno de navegación, la tarea a desarrollar y la determinación de tiempo para cumplir dicha tarea.

Este problema ha sido resuelto por técnicas y algoritmos que utilizan principios diversos tales como la minimización de funciones, teoría de grafos o métodos heurísticos, entre otros [52], [63],[73].

Considerando trabajos en el área [74], [44], [4], que abarcan estudios en entornos bidimensionales y tridimensionales, se ha extraído la siguiente clasificación de métodos representativos:

- Campos potenciales artificiales
- Probabilísticos y aleatorios
    - Árboles aleatorios de exploración rápida (RRT, por sus siglas en inglés)
- Basados en grafos
    - Descomposición en celdas
- Basados en optimización estocástica
    - Optimización por enjambre de partículas (PSO, por sus siglas en inglés)

Cada una de las técnicas desarrolladas tiene ventajas y desventajas al compararse en diferentes casos debido a la naturaleza de su algoritmo. Por ejemplo, los métodos reactivos o de respuesta rápida representan el entorno como funciones tridimensionales, en donde los obstáculos tienen valores altos y el vehículo debe minimizar el camino entre un punto elevado de inicio y un punto bajo de llegada. Entre estos métodos se encuentran los denominados campos potenciales artificiales [69], [36], [49]. Este es una solución elegante e intuitiva que se basa en la minimización de funciones mediante el cálculo de gradiente descendiente.
Propuesto en 1986 por Kahtib [46], se asume al robot como una partícula con carga eléctrica opuesta a la de las funciones que representan a los obstáculos. El entorno de navegación es la suma de funciones de potencial atrayentes para el punto final deseado y repulsivo para el punto inicial y los obstáculos. Bajo esta estructura ha sido implementado con éxito para el caso de un único robot [42], [71].

Cabe resaltar que debido a la limitación de la creación de una función multidimensional para cada entorno de navegación, es muy probable que se generen mínimos locales que eviten al robot cumplir la tarea de hallar el mínimo global. Algunas de las investigaciones en busca de eliminar este problema, se han enfocado a la modificación del diseño de las funciones de potencial, proponiendo distintos tipos de funciones. Entre estas se destacan las denominadas super-quadraticas [35] que usan un conjunto de formas rectangulares o elipsoides aplicando fórmulas cuadráticas recursivas, las funciones armónicas, propuestas en [34], basadas en la solución de la ecuación de Laplace para la transferencia de calor [12] y aquellas que proponen S.S. Ge y Y.J. Cui en [18] para el caso de obstáculos dinámicos. Dichos métodos, por centrarse en la representación de los obstáculos, son conocidos como enfoques de campo potencial local.

Existe también el enfoque global, que considera la totalidad de los obstáculos y la meta para la creación de una función de potencial total y de esta manera eliminar la presencia de mínimos locales mediante la sintonización de parámetros. El ejemplo más claro es el propuesto en [54], que asegura la creación de un único mínimo local en el objetivo, rellenado los demás puntos de equilibrio. No obstante, su desventaja radica en que debe conocerse completamente el espacio de trabajo. Una aplicación de esta técnica se aprecia en [36], desarrollada para la cooperación entre varios robots en formación.

Otras orientaciones recientes para solventar el grave inconveniente de los mínimos locales consideran el comportamiento estocástico y de enjambre como se puede estudiar en [45], donde se añade a la propuesta inicial, algoritmos como el genético y los evolutivos bacteriológicos. Por su parte, [14] da a conocer una propuesta innovadora, que añade el comportamiento de enjambres de partículas activas brownianas para escapar de los mínimos locales.

Por otra parte, se encuentran los métodos heurísticos como el de optimización por enjambres de partículas (PSO, por sus siglas en inglés), que emulan comportamientos naturales para optimizar una función de bienestar con un campo de búsqueda amplio [33], [53], [40]. Esta búsqueda resulta útil, debido a la cantidad de agentes que exploran y evalúan una función de aptitud de una forma bioinspirada. Por la misma razón se asocia el beneficio de evitar caída en mínimos locales. Un análisis general de esta metodología se encuentra en [9], donde se demuestra la pertinencia del enfoque orientado a la búsqueda de rutas de un vehículo autónomo dentro de un entorno con obstáculos estáticos y dinámicos.
Existen ejemplos de la aplicación de distintas técnicas, como el caso del algoritmo genético presentado en [19], las técnicas basadas en colonias de hormigas como se muestra en [50] y [8], aquellas basadas en quimio-taxis bacteriana [62] y el algoritmo de optimización de enjambre de partículas (PSO, por sus siglas en inglés) [53],[33].
Por su parte, existen algunos métodos que hacen uso de la teoría de grafos para establecer trayectos entre espacios característicos del medio, uniendo sus

centroides con caminos definidos [4], o bien sus vértices como los grafos de visibilidad [22].
En este enfoque se busca la construcción de un grafo para conectar el punto de partida con el de llegada. Posteriormente, se halla el camino más apropiado satisfaciendo algún índice de desempeño, que generalmente es la distancia más corta [14].

Esta planificación tiene un enfoque geométrico y se destacan las técnicas de grafos de visibilidad, diagramas de Voronoi, descomposición en celdas, descomposición rectangular, descomposición vertical y descomposición circular, ampliamente estudiadas en [74] y [4]. La primera técnica representa los obstáculos con polígonos e identifica que puntos del espacio se pueden unir con un segmento de recta sin interferir con las posiciones de colisión. La segunda técnica, forma segmentos rectilíneos tal que estén lo más alejados posible de los obstáculos. Finalmente, el tercer método y sus semejantes, como primera etapa descomponen el espacio libre en celdas y seguidamente construyen el grafo de conectividad de una manera similar a los grafos de visibilidad. Al final se encuentra la ruta óptima aplicando un algoritmo de búsqueda según el criterio de desempeño de menor distancia recorrida.

De entre estas técnicas se resalta la descomposición por celdas, ya que puede ser exacta o aproximada. En el enfoque exacto, la unión de las celdas corresponde al espacio libre [64] y en el enfoque aproximado se construyen celdas de una forma predeterminada y por tanto su unión no es necesariamente el espacio libre [14].

Finalmente, existen métodos basados en mapas de ruta probabilísticos, en donde la elección de la mejor trayectoria se realiza teniendo en cuenta la conformación de árboles y una búsqueda sistemática, como en la exploración rápida de árboles aleatorios (RRT, por sus siglas en inglés) [30], [31]. Cabe resaltar que son algoritmos de búsqueda local [13] porque exploran parte del espacio con movimientos aleatorios para definir una secuencia de configuraciones (posiciones) libres de colisión, de tal manera que en cada iteración se acercan más a la posición final.
Dos de los más característicos incluyen: los mapas probabilísticos (PRM, por sus siglas en inglés), en donde se crea una ruta de forma aleatoria mediante la selección de puntos al azar en el espacio, para conectar rápidamente el punto de inicio y llegada del móvil [32]. Además, se tiene los árboles aleatorios de exploración rápida (RRT, por sus siglas en inglés), propuesto en 1998 con la expansión de un árbol con el fin de generar la mayor cantidad de ramificaciones en el entorno de navegación y de esta manera buscar de entre una gran cantidad de rutas posibles, la más óptima de ellas [37]. Son muchas las variaciones realizadas a este algoritmo ya que es prometedor y tiene completitud probabilística (es decir, que la probabilidad de encontrar el mejor camino tiende a 1 exponencialmente con el número de nodos). Entre ellas se encuentra RRT_Biderectional, RRT de consulta única y RRT_ExtCon [39].

Es importante mencionar los análisis de comportamiento asintótico en [30] y [31], en donde además de presentar un sólido análisis teórico de los algoritmos PRM y RRT se proponen dos nuevos algoritmos PRM* y RRT* que añaden el beneficio de hacer que la solución converja casi seguramente al óptimo.

En cuanto se refiere a la planificación de trayectorias en entornos dinámicos, se debe encontrar una ruta libre de colisión teniendo en cuenta obstáculos móviles o la navegación de múltiples robots en el mismo entorno. Una aplicación que considera los campos potenciales artificiales para resolver esta tarea se observa en [76], en donde se propone una nueva metodología para planeación de ruta en tiempo real que permite el cálculo de maniobras para evitar obstáculos dinámicos.

El principal caso de estudio se ha centrado en la planeación orientada a los grupos de robots, ya que el problema es mucho más complejo y tiene un gran número de aplicaciones [47], y casos de estudio. Entre ellos se tiene la navegación de múltiples robots, el control de tráfico, la generación de formaciones, el seguimiento o búsqueda de un objetivo, y el despliegue de robots.

La taxonomía dentro de la navegación de múltiples robots divide los estudios en técnicas centralizadas (aquellas que planean directamente en la suma de los espacios de configuración del equipo de robots) y descentralizadas (aquellas que dividen el problema en partes y por tanto cada robot planea su propia trayectoria independientemente). En [36] se aprecia el desarrollo de ejemplos de ambas técnicas para mantener formación de un equipo de robots considerando los campos potenciales artificiales y el análisis matemático de un sistema mecánico restringido.

Desde el enfoque distribuido, se incluye el término agente para determinar al robot como un individuo capaz de percibir su ambiente, procesar sus percepciones y responder o actuar en su entorno de manera inteligente. Bajo esta premisa, resaltan los siguientes trabajos: En [25] se presenta una metodología para el análisis y diseño de sistemas multi-agente robóticos, mientras que en [26] se muestra una extensión del método de descomposición por celdas para planeación de movimiento inteligente en sistemas de múltiples robots. Por su parte, [23] muestra una extensión del método de campos de potencial aplicado a la implementación de una red de sensores móviles, en [59] una unificación de técnicas de potencial artificial, geométricas y probabilísticas para el despliegue de múltiples robots y en [7], un enfoque matemático para la coordinación de movimientos en grupos de robots. Cabe resaltar que estos desarrollos se limitan a validaciones virtuales y muchos de los casos no son óptimos cuando el número de robots crece.

Para tratar problemas con gran número de entradas se considera la robótica de enjambre. En los trabajos [41] y [48] se detallan iniciativas basadas en funciones de atracción y repulsión y comportamiento real de enjambres biológicos, con la particularidad que en [41] se da a conocer una validación experimental en plataformas físicas de distintos algoritmos distribuidos para sistemas multi-robot.

Dentro de estas iniciativas cabe resaltar los trabajos relacionados con evitación de colisión, motivados por el estudio de grupos grandes de robots en un escenario, automatizados por un método descentralizado.

En 1998, los autores en [15] proponen los obstáculos de velocidad, consistentes en la selección de maniobras de evitación para evadir obstáculos estáticos y dinámicos en un espacio de velocidad que se crea considerando las posiciones y velocidades de los agentes involucrados. Los inconvenientes de esta propuesta son las oscilaciones de las trayectorias generadas y el hecho de no considerar las restricciones cinemáticas dependiendo del tipo de robot.

Soluciones a estos dos inconvenientes han orientado sus propuestas desde la generalización de la idea inicial, teniendo en cuenta las restricciones de un robot [29], hasta la proposición de nuevos algoritmos con el fin de reducir oscilaciones y optimizar las trayectorias generadas para cada robot, como en [21] y [28]. Dos de los más conocidos ejemplos son: Evitación óptima de colisiones recíprocas ORCA (por sus siglas en inglés) y obstáculos de velocidad recíproca RVO (por sus siglas en inglés). En [11] y [24] se encuentra un compendio teórico de ambas propuestas.

Estos nuevos algoritmos generan trayectorias más suaves en comparación con el original. No obstante, algunas siguen sin considerar restricciones cinemáticas y por lo tanto, los últimos desarrollos en el área buscan incluir dichas restricciones en los modelos y a su vez satisfacer el criterio de ruta suave y óptima para la navegación de robots móviles. Trabajos relacionados se encuentran en [1], [2], [6], [65], [66].

En este trabajo, haciendo uso de simulaciones, se evalúan métodos representativos para la generación de trayectorias con el fin de calcular algunos índices de desempeño como: el tiempo empleado, la distancia y la eficiencia de las rutas. Dadas las diferentes características de las metodologías programadas, medir el desempeño en tres escenarios típicos de arreglos de obstáculos permite analizar las ventajas de cada algoritmo, orientando la elección de los métodos a la solución de problemas particulares.
Teniendo en cuenta que son múltiples los estudios relacionados en el área, pero que encaran el problema considerando al robot como una partícula sin restricciones cinemáticas, concentrándose en desarrollos fuera de línea (es decir validaciones virtuales con un único robot) y que los estudios relacionados con implementaciones en plataformas robóticas físicas son escasos, al igual que los estudios de comparación entre los métodos existentes, este trabajo presenta una síntesis teórica orientada a la navegación de vehículos terrestres de tracción diferencial exponiendo una técnica de control de movimientos y dos para la generación de trayectorias para vehículos con restricciones cinemáticas de tipo no holonómica. Como punto clave de este trabajo, se aporta una metodología de comparación analítica entre métodos y se da a conocer los resultados de la implementación virtual de algoritmos representativos en tres escenarios, para el caso de un único robot. Además, contribuye con la propuesta de dos algoritmos para la navegación de múltiples robots en un escenario controlado y expone los resultados de la

implementación en una plataforma física real de una de las técnicas analizadas, en el caso de un único robot.

Por último, a partir del compendio teórico-práctico desarrollado, da a conocer las limitaciones de cada una de las técnicas analizadas y se concluye respecto los resultados obtenidos, ofreciendo miras a futuras investigaciones que mejoren o amplíen este estudio en robótica móvil.

Este trabajo está dividido en 7 secciones principales organizadas como sigue: Introducción, Conceptos básicos de planeación de trayectorias en robótica móvil, Procedimientos para implementación de algoritmos y análisis de resultados, Metodología de comparación analítica y análisis de resultados, Resultados, Conclusiones y Trabajo futuro.

En el capítulo 1, se introduce la temática a desarrollar, abordando los objetivos de la presente investigación. Así mismo, expone las contribuciones científicas de esta tesis, justificando la importancia de realizar el estudio en torno a la planeación de trayectorias.

Además se tratan los antecedentes referentes a la planeación de trayectorias, con el fin de establecer un marco de referencia que guía al lector en los desarrollos científicos propuestos y las modificaciones recientes de los mismos.

En el capítulo 2, se expone el marco teórico que brinda nociones sobre robótica móvil terrestre y detalla los aspectos teóricos y prácticos de los métodos de planeación de trayectorias según distintos enfoques. Además, presenta la teoría propuesta para la extensión de dos algoritmos al caso de múltiples robots y la teoría existente sobre una de las iniciativas más conocidas dentro de la evitación de colisión entre robots.

El capítulo 3, describe las metodologías programadas, dando a conocer las modificaciones propuestas a ciertos enfoques con el fin de orientar sus nociones a la planeación de trayectorias. De igual manera, expone la metodología formulada para comparar las técnicas, detallando índices de desempeño cuantitativos y cualitativos.

En el capítulo 4, se presenta la metodología de comparación propuesta. Además se aborda la aproximación a trabajo futuro como contribución añadida a los objetivos de esta investigación.

En el capítulo 5, se da a conocer los resultados obtenidos de la investigación realizada.

En el capítulo 6, se presenta las conclusiones resultantes de esta investigación,.

Finalmente, en el capítulo 7 se sugiere el posible trabajo futuro, puesto que los resultados obtenidos son base para definir nuevas investigaciones sobre el tema.

2. CONCEPTOS BÁSICOS DE PLANEACIÓN DE TRAYECTORIAS EN ROBÓTICA MÓVIL

Este capítulo ilustra sobre definiciones, conceptos y metodologías remarcables que solucionan el problema de planeación de trayectorias en robótica móvil. A su vez, considera la navegación de vehículos completa, es decir, la planeación de trayectorias, el post-procesado de ruta (generación de trayectoria) y el control de movimientos, en conjunto. Sin embargo, esta tesis se enfoca a la planeación de trayectoria, y por lo tanto, los demás componentes orientados a navegación son expuestos como requisito necesario para el desarrollo de la implementación de los algoritmos de planeación, en una plataforma física real. Para esto, se presentan dos técnicas de post-procesado y un método de control de movimientos lineal.

2.1. ROBÓTICA MÓVIL TERRESTRE

2.1.1. Introducción

La robótica móvil terrestre se limita al estudio de robots con capacidad de desplazarse por tierra mediante el uso de ruedas, patas, orugas u otro mecanismo de locomoción. Los problemas se estudian en entornos bidimensionales y tridimensionales. En este trabajo, el enfoque es bidimensional y supone las ruedas como mecanismo de locomoción.

2.1.2. Definiciones y notación

Las siguientes definiciones toman como base los estudios presentados en [10], [14], [44], [68].

a) Espacio de trabajo

Es el conjunto W de todas las posiciones a las que puede acceder un robot A, siendo un sub conjunto del plano cartesiano $\mathbb{R}^2$.

Comúnmente, el espacio de trabajo contiene obstáculos y por esta razón existen los subconjuntos W_f, que incluyen las secciones del plano donde no hay presencia de obstáculos y W_o, que incluye aquellas zonas que si los poseen. Luego $W = W_f + W_o$.

b) Espacio de configuraciones

Es un vector q que permite representar la localización de un robot A en W. Incluye todas las configuraciones posibles que simbolizan las posiciones de las partes móviles del robot respecto a un marco de referencia fijo.

Se representa con un vector columna $C = [q_1, q_2, \dots q_n]^T$ donde q_i representa un valor particular de desplazamiento.

c) Grados de libertad

Son el número de parámetros necesarios para calcular los movimientos de un robot A en el espacio de trabajo.

En robótica móvil terrestre bidimensional, se consideran tres grados de libertad respecto a un punto de referencia fijo y son: la posición del robot en coordenadas (x, y) y el ángulo de orientación (Θ).

d) Cinemática Directa e Inversa

Teniendo en cuenta la cinemática para modelar matemáticamente la relación entre actuadores y movimiento del robot, se tiene una representación de cinemática directa o inversa dependiendo de las variables independientes y dependientes del sistema inmersas en las ecuaciones de movimiento.

En cinemática directa, se calcula la posición cartesiana del robot conociendo las configuraciones iniciales y los movimientos de las ruedas, es decir, sus velocidades angulares. Por consiguiente, se logra la transición del espacio de configuraciones al espacio de trabajo. En cinemática inversa, se calcula las velocidades angulares requeridas para lograr el movimiento de una posición a otra. Entonces, se logra la transición del espacio de trabajo al espacio de configuraciones.

e) Planificación de trayectorias

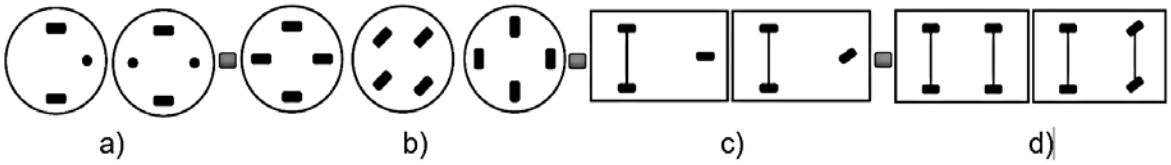
El problema de planificación de trayectorias se define como sigue: Sea $q = (x, y, \Theta)$ el vector de configuración del robot, $A_q = q$ el subconjunto de C ocupado por el robot A cuando se encuentra en q y $B(q) = \{b_{1,} b_2, \dots, b_{n,}\}$, los obstáculos definidos como un conjunto de objetos rígidos en el entorno. Entonces, planificar una trayectoria se define como el hecho de encontrar una sucesión de posturas q que deben pertenecer a C_f, donde $C_f = \{q \in C: A(q) \cap \bigcup_{i=1}^{n} b_i(q) = \emptyset\}$, es decir que aquellas posturas q ocupadas por el robot A no coinciden en ningún momento con las posiciones en que están definidos los obstáculos en el escenario de navegación.

2.1.3. Sistemas de locomoción de robots móviles

Según [44] y [71], existen los siguientes tipos de tracción en vehículos móviles dependiendo del centro instantáneo de rotación, la cantidad de ruedas motrices y directrices, fijas o locas y de castor (Ver Figura 1):

- Diferencial
- Síncrona
- Triciclo
- Ackerman
- Holónomas

Figura 1. Esquemas de locomoción terrestre.



Los rectángulos en negrilla representan las ruedas directrices y los círculos en negrilla las ruedas locas o de castor. a) Esquema de tracción diferencial b) Esquema de tracción síncrona c) Esquema de tracción tipo triciclo d) Esquema de tracción Ackerman. **Fuente:** Esta investigación a partir de [14].

Para el desarrollo de esta investigación se trabaja con la tracción diferencial en un eje común, donde cada rueda tiene control independiente.

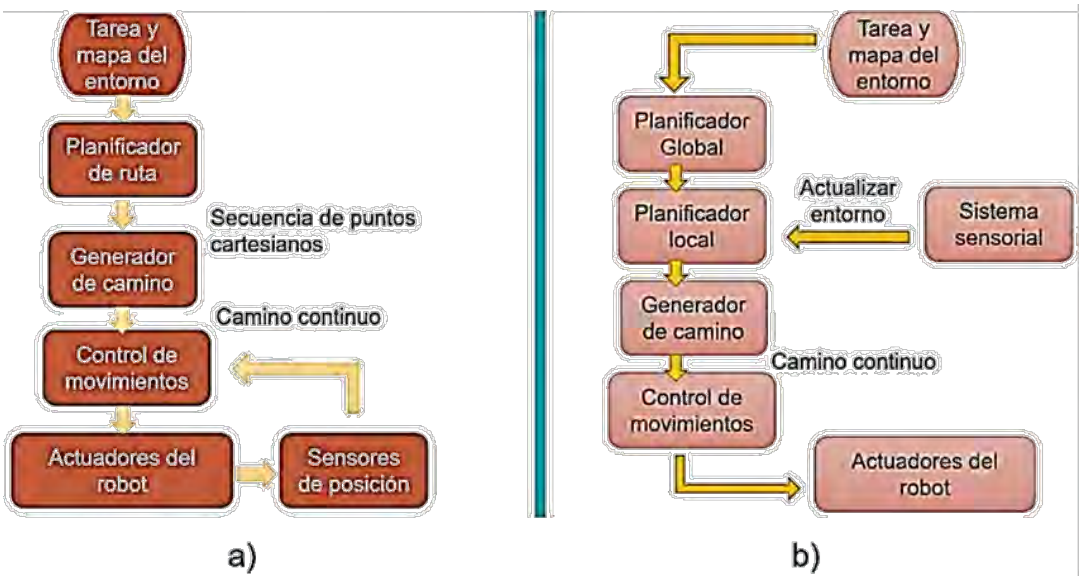
2.1.4. Navegación.

La tarea de navegación implica principalmente tres cosas: la planificación de movimiento, el post-procesado de ruta (generación de camino) y por último, el control de movimientos. Pueden desarrollarse de manera reactiva o no reactiva dependiendo de la actualización del entorno para identificar obstáculos nuevos que pueden añadirse a un escenario de navegación previamente definido (Ver Figura 2). Cabe anotar que estas actividades incluyen percepción del entorno mediante sensores (caso de estudio basado en sensores), localización de la posición del robot y la determinación del accionar del robot para seguir la trayectoria deseada.

Según [36], la planeación de movimiento brinda una serie de puntos cartesianos de entrada para establecer todos los movimientos posibles considerando o no satisfacer las restricciones cinemáticas, ya que estas pueden ser consideradas en el post-procesado de ruta que genera el camino. Por su parte, el control de movimientos se define como la tarea de lograr que el robot sigua el movimiento

planeado, lo que implica medir el actual movimiento del sistema y controlar los actuadores apropiadamente para producir las fuerzas de entrada que hagan seguir el movimiento deseado.

Figura 2. Esquemas de navegación automática de robots.



a) Esquema de navegación con planificación global, no es reactivo. b) Esquema de navegación con planificación global y local, es reactivo. Caso de estudio basado en sensores. **Fuente:** Esta investigación.

2.1.5. Restricciones cinemáticas

Este tipo de restricciones expresan los desplazamientos de muchos grados de libertad en términos de un conjunto más pequeño de desplazamientos. Para el caso de robótica móvil terrestre se tienen tres grados de libertad, dos para posición y uno de orientación. Si el caso es de restricción holonómica, se debería poder posicionar el robot en cualquier punto y orientación en el espacio de trabajo. No obstante, si estos grados de libertad están acoplados, el robot debe realizar movimientos hacia adelante, curvas y hacia atrás ya que no puede ir directamente a la posición y orientación deseada.

En los robots móviles o, en general en cualquier sistema físico, las variables que definen la posición y orientación habitualmente no son variables independientes. En la mayoría de los casos deben cumplir ciertas restricciones de movilidad llamadas restricciones cinemáticas, que pueden ser de dos tipos: ***holónomas u***

holonómicas y no holónomas o no holonómicas. En [44] se definen de la siguiente manera:

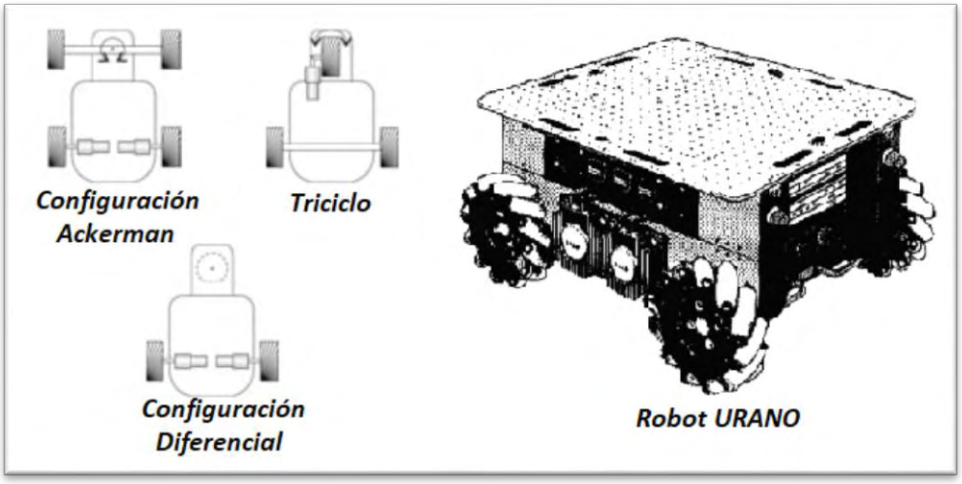
"Sea $p = [p_1, \dots, p_3]^T$ un vector de las variables necesarias para determinar completamente la posición y orientación de todas las partes de un sistema físico (…) Las holónomas *–refiriéndose a las restricciones* son aquellas en las que no interfieren las velocidades; es decir, tienen la forma:

$$G_k(p, t) = 0; \quad k = 1, \dots, s;$$

Las no holónomas dependen de las velocidades. Para que una restricción sea no holónoma se exige además que no sea integrable; es decir, que no se deduzca por derivación total con respecto al tiempo de una holónoma."

Vehículos como las bicicletas, triciclos, en configuración Ackerman o diferencial, son ejemplos de vehículos con restricciones no holonómicas. Por otra parte, el vehículo URANUS es un ejemplo de vehículos con restricciones holonómicas. (Ver Figura 3).

Figura 3. Vehículos con restricciones holonómicas y no holonómicas.



Fuente: Esta investigación a partir de [5], [75].

2.1.6. Cinemática de un robot móvil

El estudio de cinemática de un robot, que se asume sobre una superficie plana con tres grados de libertad (x, y, Φ) (que dependen de su locomoción), es el modelo matemático que representa la posición y orientación en función de la operación de los actuadores. Esto, da lugar a cinemática directa e inversa como se discutió anteriormente. Para efectos de este trabajo se analiza únicamente el caso de interés de un robot de tracción diferencial.

Cinemática de un robot móvil de tracción diferencial:

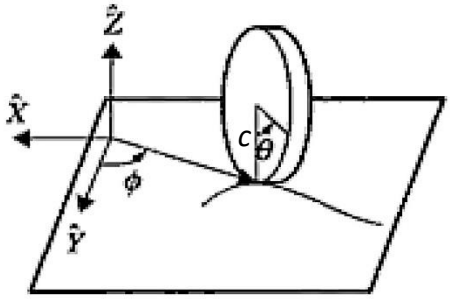
Tal como lo propone el autor en [44], para precisar el modelo se especifican las siguientes condiciones controladas:

1. La superficie de contacto con las ruedas es totalmente plana.
2. Los ejes de guiado son perpendiculares al suelo.
3. Se supone rozadura pura, es decir, se deprecia el deslizamiento en los periodos de control.
4. No se consideran partes flexibles en el robot, por lo tanto, este se comporta como un sólido rígido, de tal forma que ruedas de orientación (ejemplo: ruedas de castor, que estabilizan la plataforma y no tienen control), se posicionarán adecuadamente gracias al sistema de control.
5. Las trayectorias se pueden aproximar como arcos de circunferencia entre dos periodos de muestreo.

El comportamiento cinemático se estudia considerando las relaciones no holonómicas involucradas.

Teniendo en cuenta el movimiento de una rueda en el plano como lo muestra la Figura 4, tal que el diámetro correspondiente al punto de contacto con el suelo esté siempre en posición vertical, se pueden usar cuatro coordenadas para definir la posición y orientación de la rueda: (x, y) para el punto de contacto y (θ, Φ) para el ángulo entre la vertical y un radio de referencia (indica cuánto ha girado la rueda), y el ángulo de orientación de la rueda, respectivamente.

Figura 4. Coordenadas para especificar posición y orientación de una rueda en el plano.



Fuente: Esta investigación a partir de [44].

Cabe anotar que, la condición de rozadura pura, introduce dos restricciones, porque el espacio que el punto de contacto recorre sobre el borde la rueda es igual al que recorre en el plano. Luego que, proyectando la velocidad del punto de contacto en el plano, paralela y perpendicularmente al disco se llega a

$$-x'.sen\Phi + y'.cos\Phi = \theta'.c$$

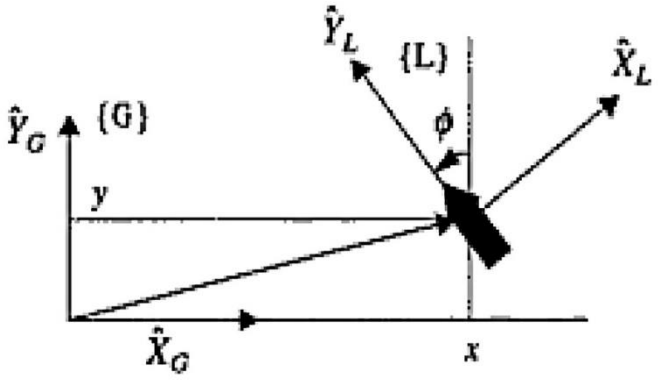
$$x'.cos\Phi + y'.sen\Phi = 0 \qquad (1)$$

Donde x', y' representan el movimiento dinámico sobre el plano, Θ' la velocidad de giro de la rueda y c el radio de la rueda.

Nótese que no es posible obtener funciones que relacionen las variables (x, y, Θ, Φ) a partir de (1), y que las direcciones de movimiento deben satisfacer esta ecuación.

Ahora, considérese un sistema de referencia {G} y un sistema {L} con centro en el punto de guiado del vehículo y el eje $\hat{Y}_L$ en la dirección del eje longitudinal del vehículo, como lo muestra la Figura 5.

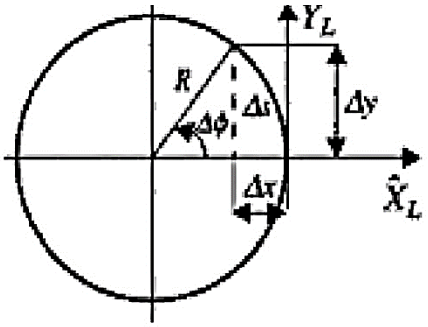
Figura 5. Sistema de referencia en navegación de robots móviles.



Fuente: [44].

Bajo la última de las condiciones controladas, supóngase que el vehículo se desplaza en un intervalo de control según un arco de circunferencia, como lo indica la Figura 6.

Figura 6. Contacto osculador.



Fuente: [44].

En este orden de ideas, la velocidad lineal y angular del vehículo es:

$$v = \frac{\Delta s}{\Delta t} \qquad y \qquad w = \frac{\Delta \Phi}{\Delta t} \tag{2}$$

Siendo Δs y $\Delta \Phi$, el espacio recorrido por el punto de guiado del robot y su cambio de orientación durante un intervalo de control Δt.

Por geometría básica, se sabe que la longitud del arco Δs recorrido en Δt se puede calcular como $R.\Delta\Phi$, donde R es el radio de giro, y que la curvatura se define como la inversa de este radio de giro, así:

$$\gamma = \frac{1}{R} = \frac{\Delta\Phi}{\Delta s} \tag{3}$$

Por consiguiente, las ecuaciones de movimiento del sistema {L} de la Figura 6, en la posición de inicio son:

$$L_{(\Delta x)} = -(R - R.cos(\Delta\Phi))$$

$$L_{(\Delta y)} = R.sen(\Delta\Phi) \tag{4}$$

En caso de que la orientación del robot con respecto al sistema {G} sea de Φ, el movimiento en el sistema {G} se determina rotando Φ:

$$\Delta x = R.[\cos(\Delta\Phi) - 1].\cos\Phi - R.sen(\Delta\Phi).sen\Phi$$

$$\Delta y = R.[\cos(\Delta\Phi) - 1].\mathrm{sen}\Phi - R.sen(\Delta\Phi).cos\Phi \tag{5}$$

Suponiendo intervalos de control suficientemente pequeños, también lo serán los cambios de orientación $\Delta\Phi$ y por lo tanto se aproximan como

$$\cos(\Delta\Phi) = 1$$

$$sen(\Delta\Phi) = \Delta\Phi \tag{6}$$

Que luego de sustituir en (5) se tiene

$$\Delta x = -R.\Delta\Phi.\mathrm{sen}\Phi$$

$$\Delta y = R.\Delta\Phi.cos\Phi \tag{7}$$

Reescribiendo (7) en términos de Δs y dividiendo ambas expresiones por Δt, tal que $\Delta t \rightarrow 0$, se llega a

$$x' = -v.\mathrm{sen}\Phi$$

$$y' = v.cos\Phi \tag{8}$$

A estas ecuaciones puede añadirse la que se obtiene a partir de la velocidad angular y que proporciona la variación de la orientación:

$$\Phi' = w \tag{9}$$

Una vez logradas estas fórmulas, se procede a representarlas a partir de un modelo jacobiano. Para esto se considera p como un vector que representa un punto en el espacio de n coordenadas generalizadas y q como un vector de m variables de actuación, donde $n > m$. En lo que sigue se considera que las variables se expresan en el sistema de referencia global {G}, obteniendo, el modelo directo de la forma

$$p' = J(p)q' \tag{10}$$

Donde $J(p)$ es el jacobiano de la expresión. Según [44], este jacobiano puede escribirse de la forma:

$$p' = f(p) + \sum_{i=1}^{m} g(p)_i . q'_i \tag{11}$$

Donde f y g son funciones vectoriales analíticas. Si $p = [x, y, \Phi]^T$ es el vector de configuración del robot, las ecuaciones (8) y (9) pueden expresarse en la forma de (11) con $f(p) = 0$ y $m = 2$, así

$$p' = \begin{bmatrix} -sen\Phi \\ cos\Phi \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w \tag{12}$$

Donde v y w son la velocidad lineal y angular, respectivamente.

La expresión (12) también puede ser escrita considerando $q' = [v, w]^T$ como el vector de variables de entrada

$$\begin{bmatrix} x' \\ y' \\ \Phi' \end{bmatrix} = \begin{bmatrix} -sen\Phi & 0 \\ cos\Phi & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} v \\ w \end{bmatrix} \tag{13}$$

Combinando las dos primeras relaciones de (13) se llega a la segunda restricción no holonóma del movimiento, identificada en (1), según la cual el robot debe moverse en cada instante acorde la dirección de su eje longitudinal de simetría, ya que la posición (x, y) y la orientación Φ no son independientes.

Por otra parte, para obtener el modelo inverso, y teniendo en cuenta que el jacobiano no es cuadrado, se aplica la seudoinversa, multiplicando ambos miembros de (13) por J^T y despejando q', para finalmente llegar a

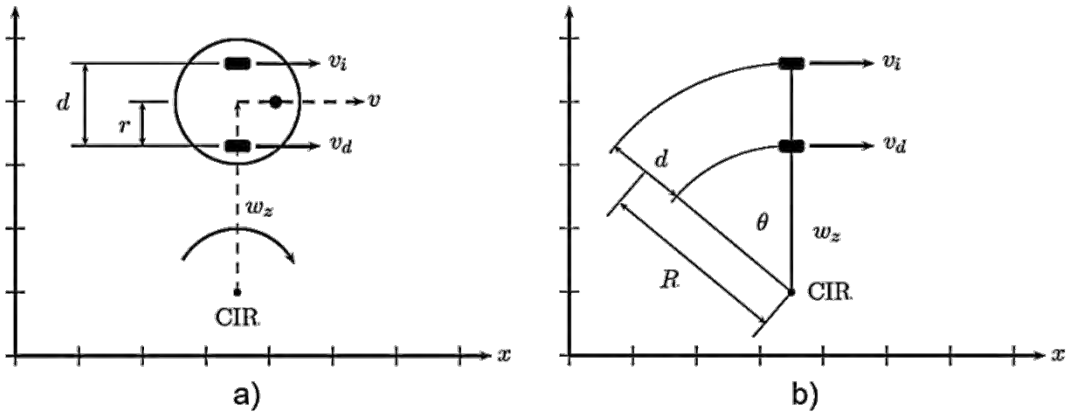
$$q' = \{[J(p)]^T . J(p)\}^{-1} [J(p)]^T . p'$$

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} -sen\Phi & cos\Phi & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x' \\ y' \\ \Phi' \end{bmatrix} \tag{14}$$

De donde la relación para v coincide con la primera restricción no holónoma expuesta en (1).

Consecuentemente, este modelo aplica a la tracción diferencial usando v y w dependiendo de la geometría del vehículo. Por esto, se estudia la plataforma según la Figura 7 que relaciona las variables de interés con el centro instantáneo de rotación.

Figura 7. Velocidades de vehículo de tracción diferencial respecto centro instantáneo de rotación CIR.



a) Velocidades inherentes a un robot de tracción diferencial con plataforma circular respecto al centro instantáneo de rotación CIR. b) Relación velocidad lineal y velocidad angular w_z con las velocidades de la rueda izquierda v_i y derecha v_d y el CIR. **Fuente:** [14]

De la figura 2b es claro que

$$v_i = w_z(R + r)$$
$$v_d = w_z(R - r) \quad (15)$$

Restando estas dos expresiones se obtiene el valor para w_z

$$w_z = \frac{V_d + V_i}{2.r} = \frac{R(w_i + w_d)}{2.r} \quad (16)$$

La velocidad de cada una de las llantas tiene una componente lineal v y otra debida a la velocidad angular, por lo cual

$$v_i = v + rw_z$$
$$v_d = v - rw_z \quad (17)$$

Sumando las relaciones anteriores, se obtiene la velocidad lineal del robot dada por

$$v = \frac{V_i + V_d}{2} = \frac{R(w_i + w_d)}{2} \quad (18)$$

2.1.7. Generación de trayectorias

Los métodos de planificación de trayectorias, otorgan trayectorias para vehículos con restricciones holonómicas, puesto que el único objetivo que cumplen es generar trayectorias *continuas* (C^0) de un punto final a uno deseado, sin tener en cuenta las

dinámicas del móvil. Sin embargo, no es suficiente con generar un camino, puesto que se debe asegurar de que pueda ser atravesado por el vehículo. Aquí radica la importancia de considerar las restricciones.

En la literatura se han planteado bastantes soluciones al problema de las restricciones cinemáticas. En [60], por ejemplo, se modifica el método de campos potenciales para acoplarlo a las características no holonómicas de un robot diferencial. Por otra parte, en [57] se utilizan círculos en los vértices del grafo para suavizar los "bordes" de la trayectoria, ya que el empleo de círculos es una técnica bastante usada en métodos basados en grafos y mapas de ruta. Pero las técnicas que generalmente se usan, son las basadas en interpoladores.

A continuación, con base en [55] se explica resumidamente, la generación de las curvas paramétricas Spline Cúbicas y B-Spline, y sus aplicaciones como técnicas de interpolación al solucionar el problema de las restricciones cinemáticas.

a) Técnica de interpolación basada en curvas Spline Cúbicas:

Las curvas paramétricas Spline son una familia de curvas diseñadas para interpolar puntos en un plano. Estos puntos, que se denominan *puntos de control*, pueden ser vértices de la trayectoria o puntos tomados ordenada u arbitrariamente de la misma.

La más sencilla de las curvas es la Spline cúbica, genera polinomios de tercer grado entre cada punto de control, como se muestra en la Figura 8, de la siguiente manera:

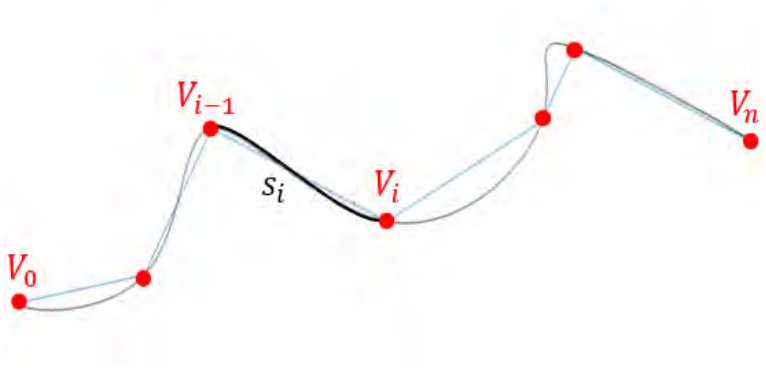
Sea $p(t) = [x(t), y(t)]$ un vector que expresa las coordenadas de un punto sobre una curva en el plano cartesiano, donde $x(t)$ y $y(t)$ son funciones que dependen del parámetro *t*. Entonces la derivada de $p(t)$ respecto a *t* está dada por

$$p' = \frac{d}{dt}p(t) = \left(\frac{d}{dt}x(t), \frac{d}{dt}y(t)\right) = (x', y') \tag{19}$$

Por lo tanto, la mayoría de aplicaciones requieren de la existencia de al menos la primera derivada de la trayectoria. Nótese, por ejemplo, que del modelo inverso del robot diferencial (14) se puede inferir que tomando como insumo x' y y', es posible encontrar las variables de entrada requeridas para que la planta siga la trayectoria deseada. Por ello, si se limita el parámetro *t* al intervalo $[0,1]$, para cada segmento de curva que se crea entre dos puntos de control, se tiene:

Sean $V = \{V_0, \ldots, V_n\}$ con $n \in \mathbb{Z}^+$, un polígono de *n+1* vértices o puntos de control y s_i, definido por $p_i(t)$, el segmento de curva generada entre los vértices V_{i-1} y V_i, donde $i \in \mathbb{Z}\ tales\ que\ 0 < i \leq n$. Entonces, si $t \in [0,1]$, $p_i(0) = V_{i-1}$ y $p_i(1) = V_i$.

Figura 8. Una interpolación de los vértices de un polígono de control V, por el método Spline.



Fuente: Esta investigación.

Para que la curva resultante de unir todos los segmentos de curva tenga primera derivada, se debe cumplir además que $p_i{}'(t)$ exista para todo t y que $p_i{}'(0) = p_{i-1}{}'(1)$ y $p_i{}'(1) = p_{i+1}{}'(0)$. Esto se cumple con la aplicación de un polinomio de grado tres como mínimo, el cual se define como

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (20)$$

$$p'(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (21)$$

$$p''(t) = 2a_2 + 6a_3 t \quad (22)$$

Con a_0, a_1, a_2 y a_3 constantes. Para el *i-esimo* punto de control, se tiene

$$p_i(0) = V_{i-1} = a_0 \quad (23)$$

$$p_i(1) = V_i = a_0 + a_1 + a_2 + a_3 \quad (24)$$

$$p_i{}'(0) = a_1 \quad (25)$$

$$p_i{}'(1) = a_1 + 2a_2 + 3a_3 \quad (26)$$

Donde

$$a_0 = V_{i-1} \quad (27)$$

$$a_1 = p_i{}'(0) \quad (28)$$

$$a_2 = 3(V_i - V_{i-1}) - 2p_i'(0) - p_i{}'(1) \quad (29)$$

$$a_3 = -2(V_i - V_{i-1}) + (p_i'(0) + p_i{}'(1)) \quad (30)$$

Sustituyendo en (20) y agrupando términos se obtiene

$$p_i(t) = (2t^3 - 3t^2 + 1)V_{i-1} + (-2t^3 + 3t^2)V_i + (t^3 - 2t^2 + t)p_i'(0) + (t^3 - t^2)p_i{}'(1) \quad (31)$$

La Ecuación (31), es la ecuación paramétrica de una curva única que pasa por los puntos V_{i-1} y V_i, y cuya derivada con respecto al parámetro en dichos puntos es $p_i'(0)$ y $p_i{}'(1)$, respectivamente. Cumplir la condición $p_i{}'(0) = p_{i-1}{}'(1)$ y $p_i{}'(1) = p_{i+1}{}'(0)$, garantiza que en la unión de dos segmentos exista la primera derivada, implicando la inclusión de todos los segmentos en el cálculo. Para esto se utiliza el siguiente método

Nótese que si en un punto V_i se cumple la condición $p_i{}''(1) = p_{i+1}{}''(0)$, que garantiza la existencia de la *segunda* derivada; se garantiza también la existencia y suavidad de la primera derivada en dicho punto. De tal manera, de la ecuación (22) se tiene

$$p_i{}''(1) = 2a_{i_2} + 6a_{i_3} \quad (32)$$

$$p_{i+1}''(0) = 2a_{i+1_2} \quad (33)$$

Sustituyendo las constantes de acuerdo al índice *i,* con las ecuaciones (27) – (30)

$$p_i{}''(1) = -6(V_i - V_{i-1}) + 2p_i'(0) + 4p_i'(1) \quad (34)$$

$$p_{i+1}''(0) = 6(V_{i+1} - V_i) - 4p_{i+1}'(0) - 2p_{i+1}'(1) \quad (35)$$

Aplicando $p_i{}''(1) = p_{i+1}''(0)$ y $p_i{}'(1) = p_{i+1}'(0)$, y agrupando se tiene

$$6(V_{i-1} - V_{i+1}) = -2(p_i'(0) + 4p_i'(1) + p_{i+1}'(1)) \quad (36)$$

Para un conjunto $V = \{V_0, V_1, V_2\}$ de únicamente tres puntos de control, se puede encontrar el valor de $p_1'(1)$ en la ecuación (36) (es decir, el valor de la derivada en el punto que los dos segmentos de curva compartirían). Este cálculo se realiza asignando valores predefinidos a $p_i'(0)$ y $p_{i+1}'(1)$, como por ejemplo, la velocidad inicial y final del trayecto, para aplicaciones en robótica.

Para *n+1* puntos de control se crea un sistema de ecuaciones como el siguiente:

$$\begin{pmatrix} p_i' \\ 6(V_0 - V_2) \\ 6(V_1 - V_3) \\ \vdots \\ 6(V_{n-3} - V_{n-1}) \\ 6(V_{n-2} - V_n) \\ p_f' \end{pmatrix} = -2 \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & & 0 & 0 & 0 & 0 \\ & & \vdots & & & \ddots & & \vdots & & \\ 0 & 0 & 0 & 0 & 0 & & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} p_1'(0) \\ p_2'(0) \\ p_3'(0) \\ \vdots \\ p_{n-1}'(0) \\ p_n'(0) \\ p_n'(1) \end{pmatrix} \quad (37)$$

Donde los valores p_i'y p_f' son los valores de la derivada al inicio y al final del tramo, que pueden ser escogidos con algún criterio especial de la implementación, igualados a cero o simplemente ser valores arbitrarios. La solución del sistema de ecuaciones, tiene la forma:

$$\boldsymbol{A}_{(n+1)\times 1} = -2\boldsymbol{B}_{(n+1)\times(n+1)} \times \boldsymbol{C}_{(n+1)\times 1}$$

$$\boldsymbol{C}_{(n+1)\times 1} = -\frac{1}{2}\left(\boldsymbol{B}_{(n+1)\times(n+1)}\right)^{-1} \times \boldsymbol{A}_{(n+1)\times 1} \quad (38)$$

Donde $\left(\boldsymbol{B}_{(n+1)\times(n+1)}\right)^{-1}$ denota la matriz inversa de $\boldsymbol{B}$, que es una matriz diagonalmente dominante en sentido estricto, por lo que es invertible [72].

b) Técnica de interpolación basada en curvas B-Spline:

Las B-Spline son curvas paramétricas basadas en las curvas de Beizer y el algoritmo de Boor [51]. Su implementación se debe a que ofrecen un mayor control de la forma en los puntos sobre la curva, utilizando los vértices del polígono de control *V,* no como puntos de paso obligatorio, sino como "puntos de atracción" de la curva. De esta manera se generan curvas más suaves, ideales para la generación de trayectorias.

Los puntos sobre las curvas B-Spline paramétricas se definen por:

$$p(t) = \sum_{i=0}^{n} V_i N_{i,k}(t) \quad (39)$$

Donde $N_{i,k}(t)$ son funciones de ponderación de orden *k*, para la *i-ésima* curva B-Spline base, definida por las fórmulas de recurrencia

$$N_{i,1}(t) = \begin{cases} 1 & si \quad t_i \le t \le t_{i+1} \\ 0 & otro\ caso \end{cases} \quad (40)$$

y

$$N_{i,k}(t) = \frac{(t-t_i)N_{i,k-1}(t)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-t)N_{i+1,k-1}(t)}{t_{i+k}-t_{i+1}} \quad (41)$$

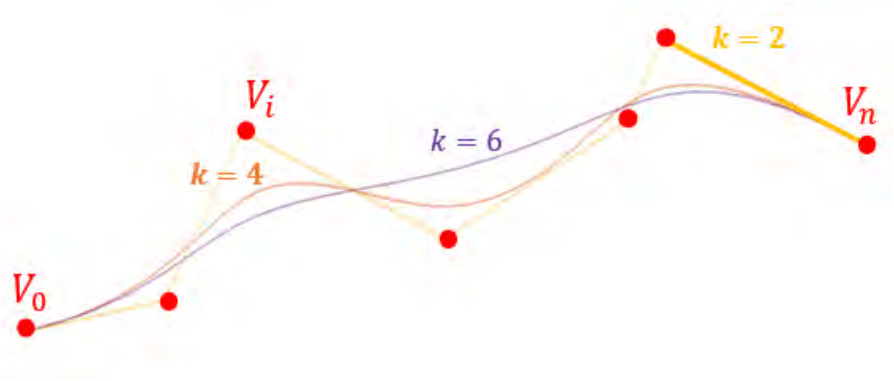
Los t_i se denominan nudos de la curva y dependen intrínsecamente del grado de las funciones de ponderación y la cantidad de vértices del polígono de control. En [44] se afirma: "Un caso especialmente útil es cuando los nudos están uniformemente espaciados con el primero y el último *k* veces", de la siguiente manera

$$t_i = \begin{cases} 0 & si \quad 0 \le i < k \\ i-k+1 & si \quad k \le i \le n+1 \\ n-k+3 & si \quad i > n+1 \end{cases}$$

Las B-Spline, son curvas paramétricas generadas por la variación de *t* en el intervalo [0, t_{max}]. Así, cuando no hay puntos de control duplicados $t_{max} = n - k + 3$.

El grado *k* de las funciones de ponderación, basadas en los polinomios de Bernstein con grado $k - 1$, está relacionada con la cercanía de la curva a los puntos de control Figura 9 y su continuidad es de orden C^{k-2}. Si el grado es $k = 2$, la curva B-Spline es exactamente igual al polinomio de control. Si *k* es igual al número de vértices del polinomio *n+1*, su comportamiento es idéntico al de las curvas de Beizer. Para garantizar la continuidad de la segunda derivada, $k = 4$ como mínimo.

Figura 9. Variaciones de la interpolación de los vértices de un polígono de control V, por el método B-Spline, variando en orden k.



Fuente: Esta investigación.

Aumentar el número de vértices puede ser una ventaja para imponer las condiciones de posición, orientación y curvatura, puesto que los vértices extra sirven como variables de control de la curva. En [44] se observa, que la orientación y curvatura de las B-Spline se deducen de la diferenciación de la ecuación (39) dada por

$$V_i^{(0)} = V_i$$

$$V_i^{(j)} = \frac{V_i^{(j-1)} - V_{i-1}^{(j-1)}}{t_{i+k-j} - t_i} \qquad para\ j > 0$$

$$p^{(j)}(t) = (k-1)\dots(k-j)\sum_{i=j-k+1}^{j} V_i^{(j)} N_{i,k-j}(t) \tag{42}$$

Donde (j) representa el orden de la derivada. La orientación y la curvatura, en función del parámetro *t*, se calcula por

$$\theta(t) = arctg\left(\frac{y^{(1)}(t)}{x^{(1)}(t)}\right)$$

$$\kappa(t) = \frac{\left[x^{(1)}(t) \cdot y^{(2)}(t)\right] - \left[x^{(2)}(t) \cdot y^{(1)}(t)\right]}{\{[x^{(1)}(t)]^2 + [y^{(1)}(t)]^2\}^{3/2}} \tag{43}$$

2.1.8. Control cinemático

En el libro de Ollero [44], capítulo 4, el problema de control de un robot diferencial se formula como:

"*El control de movimientos consiste en la obtención de leyes de control que permitan estabilizar el robot sobre un punto de trabajo, anulando el efecto de las perturbaciones, o que el robot siga de forma autónoma una trayectoria de referencia*"

Así, se describen dos tipos de control para el modelo cinemático de un robot diferencial, uno para seguimiento de trayectorias y otro para seguimiento de caminos, ambos aplicando leyes de control lineales y no lineales. Aquí se describen las generalidades del método para seguimiento de trayectorias utilizando leyes de control lineales, partiendo de lo siguiente:

Sea $\boldsymbol{z} = [x, y, \phi]^T$ un vector de variables de estado y $\boldsymbol{u}$ un vector de velocidades de control.

Si se reescriben las ecuaciones (13) del modelo cinemático para el robot diferencial en función de $\boldsymbol{z}$, tales que

$$\boldsymbol{z}' = B(\boldsymbol{z}) \cdot \boldsymbol{u} \tag{44}$$

se tiene:

$$B = \begin{bmatrix} -sin(\phi) & 0 \\ \cos(\phi) & 0 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} v \\ w \end{bmatrix} \tag{45}$$

Linealizando en $\boldsymbol{z} = 0$ y $\boldsymbol{u} = 0$:

$$\boldsymbol{z}' = B(0) \cdot \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ w \end{bmatrix} \tag{46}$$

Por tanto, nótese que con una linealización de este tipo, no es posible conseguir que el sistema resultante sea controlable, ya que el rango de la matriz de controlabilidad $c = B(0)$ es solo 2 (no es completo). Sin embargo, el robot móvil es controlable, ya que aplicando las velocidades lineal y angular correctas, el robot diferencial es capaz de llegar a cualquier posición.

Si al sistema descrito por las ecuaciones (13) se le aplica el cambio de variables

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -sin(\phi) & \cos(\phi) & 0 \\ \cos(\phi) & sin(\phi) & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \tag{47}$$

Cuya derivada respecto al tiempo es

$$\begin{aligned} x_1' &= w \\ x_2' &= -x' \sin(\phi) - wx \cos(\phi) + y' \cos(\phi) - wy \sin(\phi) \\ x_3' &= x' \cos(\phi) - wx \sin(\phi) + y' \sin(\phi) + wy \cos(\phi) \end{aligned} \tag{48}$$

Y se sustituye las expresiones para *x'* y *y'* de (13) en (47), se tiene

$$\begin{aligned} x_1' &= w \\ x_2' &= v - wx_3 \\ x_3' &= wx_2 \end{aligned} \tag{49}$$

Si ahora se hace

$$\begin{aligned} u_1 &= w \\ u_2 &= v - wx_3 \end{aligned} \tag{50}$$

El modelo puede escribirse en la forma

$$\begin{aligned} x_1' &= u_1 \\ x_2' &= u_2 \\ x_3' &= x_2 u_1 \end{aligned} \tag{51}$$

Entonces, se obtiene un modelo denominado “en cadena”, importante para el estudio de los sistemas dinámicos, puesto que es posible aplicar condiciones a las entradas del sistema para lograr su controlabilidad, que garanticen su linealidad o su invarianza en el tiempo. Además, son el objeto de estudio para técnicas de control no lineal, como el método “*Backsteping*” basado en la teoría de estabilidad de Lyapunov.

Ahora, considérando que la trayectoria que se quiere seguir tiene el siguiente modelo de referencia

$$\begin{aligned} x_{ref}' &= -v_{ref} \sin(\phi_{ref}) \\ y_{ref}' &= v_{ref} \cos(\phi_{ref}) \\ \phi_{ref}' &= w_{ref} \end{aligned} \tag{52}$$

Se suponen v_{ref} y w_{ref} acotadas tales que

$$\lim_{t \to \infty} v_{ref}(t) \neq 0; \qquad \lim_{t \to \infty} w_{ref}(t) \neq 0$$

Esto implica que no se está siguiendo una postura final, sino una trayectoria.

Entonces, el objetivo es encontrar una ley de control, tal que

$$\lim_{t\to\infty} |p - p_{ref}| = 0 \tag{53}$$

Con $p = [x_{ref}, y_{ref}, \phi_{ref}]^T$, lo que se busca es una ecuación dinámica del error. Considérese el siguiente cambio de variables

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} -sin(\phi) & \cos(\phi) & 0 \\ -\cos(\phi) & -sin(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ref} - x \\ y_{ref} - y \\ \phi_{ref} - \phi \end{bmatrix} \tag{54}$$

Derivando con respecto al tiempo y sustituyendo $x'_{ref}, y'_{ref}, x', y'$, de las ecuaciones (52) y (13), se obtiene:

$$\begin{aligned} e'_1 &= w\left[\left(x - x_{ref}\right)\cos(\phi) + \left(y - y_{ref}\right)\sin(\phi)\right] - v(\sin^2(\phi) + \cos^2(\phi)) \\ &\quad + v_{ref}\left(\sin\left(\phi_{ref}\right)\sin(\phi) + \cos\left(\phi_{ref}\right)\cos(\phi)\right) \\ e'_2 &= w\left[\left(x_{ref} - x\right)\sin(\phi) + \left(y_{ref} - y\right)\cos(\phi)\right] \\ &\quad + v_{ref}\left(\sin\left(\phi_{ref}\right)\cos(\phi) - \cos\left(\phi_{ref}\right)\sin(\phi)\right) \end{aligned}$$

$$e'_3 = w_{ref} - w \tag{55}$$

Reemplazando (54) en (55) y haciendo

$$u_1 = -v + v_{ref}\cos(e_3) \qquad u_2 = w_{ref} - w \tag{56}$$

Se llega a

$$e' = \begin{bmatrix} 0 & w & 0 \\ -w & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ \sin(e_3) \\ 0 \end{bmatrix} v_{ref} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{57}$$

Linealizando el sistema (57) alrededor de los puntos de equilibrio ($e = 0, u = 0$), para e_3 suficientemente pequeño, puede emplearse el siguiente sistema lineal variante en el tiempo

$$e' = \begin{bmatrix} 0 & w_{ref}(t) & 0 \\ -w_{ref}(t) & 0 & v_{ref}(t) \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{58}$$

Si w_{ref} y v_{ref} son constantes, se tiene un sistema lineal e invariante en el tiempo. De su matriz de controlabilidad *C* se puede afirmar, que si w_{ref} y v_{ref} son diferentes de 0, el sistema es controlable, caso contrario (sistema en reposo), se pierde la controlabilidad.

Aplicando la ley de control

$$u_1 = -k_1 e_1 \qquad u_2 = -k_2 sgn\left(v_{ref}\right) e_2 - k_3 e_3 \tag{59}$$

Y sustituyendo en (58), se tiene

$$e' = Ae = \begin{bmatrix} -k_1 & w_{ref} & 0 \\ -w_{ref} & 0 & v_{ref} \\ 0 & -k_2 sgn(\, v_{ref}) & -k_3 \end{bmatrix} e \tag{60}$$

Cuyo polinomio característico está dado por

$$\det(sI - A) = s(s + k_1)(s + k_3) + v_{ref} k_2 sgn(\, v_{ref})(s + k_1) + (s + k_3) w_{ref}^2 \tag{61}$$

Si se hace $k_3 = k_1$ la ecuación característica es

$$(s + k_1)\left[s(s + k_1) + v_{ref} k_2 sgn(\, v_{ref}) + w_{ref}^2\right] = 0 \tag{62}$$

Que se puede comparar con la ecuación característica

$$(s + 2\delta b)(s^2 + 2\delta b s + b^2) = 0 \tag{63}$$

Sintonizando los valores de δ y *b* para que los polos se sitúen en lugares deseados, las ganancias de realimentación serían

$$\begin{aligned} k_1 &= 2\delta b \\ k_2 &= \frac{b^2 - w_{ref}^2}{|v_{ref}|} \\ k_3 &= 2\delta b \end{aligned} \tag{64}$$

Nótese, que k_2 aumenta cuando $|v_{ref}|$ se hace pequeña. Para evitar el problema se utiliza una técnica de variación de polos deseados con la velocidad, denominada "escalado de velocidad". De esta manera, si se hace

$$b = \left(w_{ref}^2 + \beta v_{ref}^2\right)^{\frac{1}{2}} \tag{65}$$

Con $\beta > 0$, las ganancias de control son:

$$\begin{aligned} k_1 &= 2\delta\left(w_{ref}^2 + \beta v_{ref}^2\right)^{\frac{1}{2}} \\ k_2 &= \beta |v_{ref}| \\ k_3 &= 2\delta\left(w_{ref}^2 + \beta v_{ref}^2\right)^{\frac{1}{2}} \end{aligned} \tag{66}$$

Cuando $w_{ref} = v_{ref} = 0$ no se realiza acción de control, lo cual es coherente con la pérdida de control mencionada.

Con esta técnica, los polos son variables y dependen de las velocidades de referencia, por ello es complejo determinar valores como el tiempo de establecimiento, el sobrepaso, etc, por el método del lugar geométrico de los polos.

Un método analítico podría ayudar a encontrar rangos de operación de estas variables, o bien dar una mejor perspectiva a la hora de elegir las constantes, aquí tal como en [44], se asignan valores arbitrarios a las mismas tales que $\beta = 13.75$ y $\delta = 0.9$.

2.2. PLANIFICACIÓN DE TRAYECTORIAS

2.2.1. Introducción

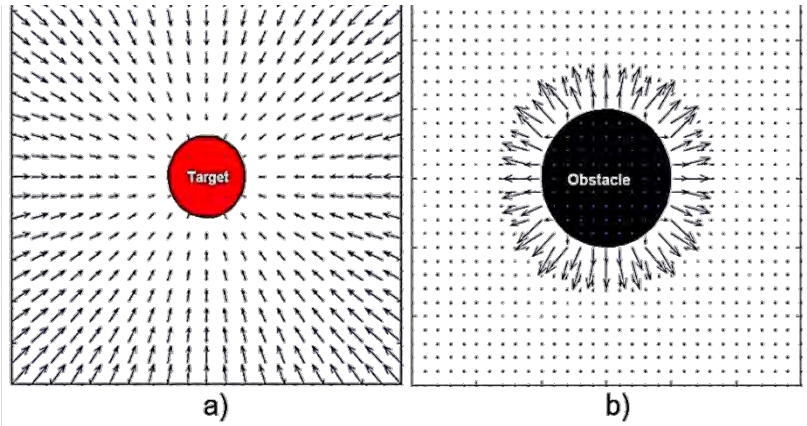
En esta sección del documento se exponen los métodos elegidos que resuelven el problema de planeación de trayectorias para un solo robot con base en desarrollos existentes en el área.

2.2.2. Método de campos potenciales artificiales. (PF)

El enfoque general de los campos de potencial artificial propuestos en [46] supone la creación de una función tridimensional a minimizar, en donde la posición objetivo se modela con una función atrayente y las funciones que representan los obstáculos con funciones repulsivas.

La Figura 10 muestra el gráfico de los vectores gradiente para el potencial atractivo y repulsivo para el objetivo y los obstáculos respectivamente.

Figura 10. Vectores gradiente para el campo de potencial atractivo y repulsivo.



a) Campo atrayente hacia el objetivo. b) Campo repulsivo de los obstáculos. **Fuente:** Esta investigación a partir de [36].

En [36] se indican dos funciones de potencial atrayente con parámetros de sintonización que permiten cambiar la forma y el efecto del campo con el fin de evitar

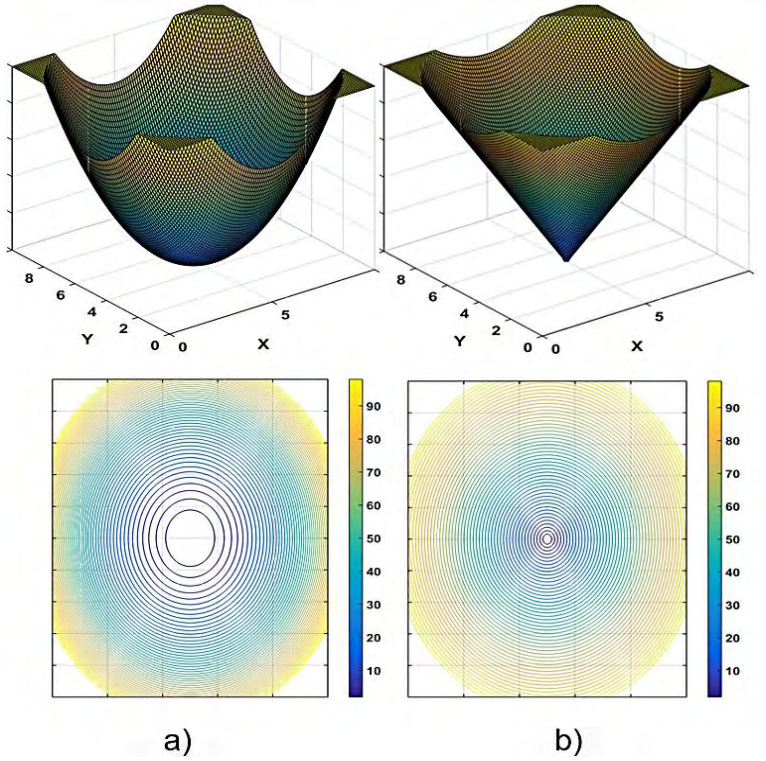
formación de mínimos locales y aumentar el descenso al objetivo para la creación de rutas directas. U_{att_1} es la propuesta original, tiene la desventaja de no siempre ubicar el mínimo global en el objetivo. Para contrarrestar este inconveniente en U_{att_2} se añade un radio para descenso lineal y así garantizar el mínimo global en la posición deseada. Para esto se definen los siguientes parámetros.

Sea q_{tar} la posición objetivo del robot y q_{rob} la posición del móvil, entonces la función de potencial está dada por

$$U_{att_1} = \frac{1}{2}\xi\|q_{tar} - q_{rob}\|^m \text{ ó } U_{att_2} = \begin{cases} \xi\|q_{tar} - q_{rob}\|^2 & , \|q_{tar} - q_{rob}\| < S \\ 2kS\|q_{tar} - q_{rob}\| - KS^2, & \|q_{tar} - q_{rob}\| > S \end{cases} \quad (67)$$

donde $m > 0$ permite un cambio en forma y $\xi > 0$ de suavidad, es decir, el efecto de campo. En U_{att_2}, $\xi > 0$ es un factor de escalamiento que modifica la forma, S y k parámetros de forma y efecto que hacen que la función sea cuadrática en un rango para luego aumentar linealmente desde un radio definido S. La Figura 11 muestra cada una de estas funciones para el objetivo en la coordenada (5,5).

Figura 11. Funciones de atracción para objetivo en coordenada (5,5).



a) U_{att_1} con $m = 2.1$ y $\xi = 5$. b) U_{att_2} con $k = 90$, $S = 0.1$ Y $\xi = 15$. **Fuente:** Esta investigación.

La función multidimensional de potencial total se obtiene sumando a la función de potencial atrayente las funciones de potencial repulsivo que figuran como los obstáculos del escenario, tal como lo indica la relación (68). Por esta razón este enfoque se conoce como campo potencial artificial local, porque las funciones de atracción y repulsión se modelan por separado.

$$U_{total}(q) = U_{att_1}(q) + U_{rep}(q) \tag{68}$$

La trayectoria se genera aplicando un algoritmo simple de descenso tal como el cálculo de gradiente descendiente de la función de potencial total U_{total}. (Ver anexo 1).

a) Funciones de potencial artificial repulsivas

Existen diversos tipos de funciones de repulsión, entre las que destacan la función FIRAS propuesta en [46], la función armónica propuesta en [34] y aquella conocida como "Ge New Potential" propuesta en [18]. Son ampliamente usadas por sus favorables efectos frente a la presencia de mínimos locales como lo expresa el autor en [36], donde destaca la función armónica con la característica de al aumentar el número de obstáculos esta es capaz de modelarlos evitando la creación de mínimos localizados y valles extensos.

A continuación se muestra la relación matemática que permite el cálculo de cada una de estas funciones donde p_o es el límite de influencia del obstáculo, η es un factor de escalamiento, λ es un parámetro que determina la función como fuente ($\lambda < 0$) o sumidero ($\lambda > 0$) y n es un parámetro de sauvizado.

FIRAS: $$U_{rep}(q) = \left\{ \begin{matrix} \frac{1}{2}\eta(\frac{1}{p_{ro}} - \frac{1}{p_o})^2 & , p_{ro} \le p_o \\ 0 & , p_{ro} \ge p_o \end{matrix} \right\}, \; p_{ro} = \|q_{obs} - q_{rob}\| \tag{69}$$

ARMÓNICA: $$\varphi = \frac{\lambda}{2\pi} \log p_{ro} \tag{70}$$

GE NEW POTENTIAL: $$U_{rep}(q) = \left\{ \begin{matrix} \frac{1}{2}\eta(\frac{1}{p_{ro}} - \frac{1}{p_o})^2 {p_{rt}}^n & , p_{ro} \le p_o \\ 0 & , p_{ro} \ge p_o \end{matrix} \right\},$$

$$p_{rt} = \|q_{tar} - q_{rob}\| \tag{71}$$

b) Función de navegación para campos potenciales artificiales *(PF)*

En [54] se expone la técnica global para la creación de la función multidimensional, cuya iniciativa es contrarrestar los inconvenientes mencionados anteriormente. Para esto se formula la función como un todo, es decir, considerando objetivo y obstáculos al mismo tiempo de creación.

Tal como el enfoque local, esta propuesta permite la planeación de una trayectoria desde un punto de partida q_{start} hasta uno de llegada q_{tar}.

La expresión matemática propuesta es la siguiente

$$U(q) = \begin{cases} \frac{\|q-q_{tar}\|^2}{\|q-q_{tar}\|^{2k}+B(q)^{\frac{1}{k}}} & para\ B(q) > 0 \\ 1 & para\ B(q) \leq 0 \end{cases},$$

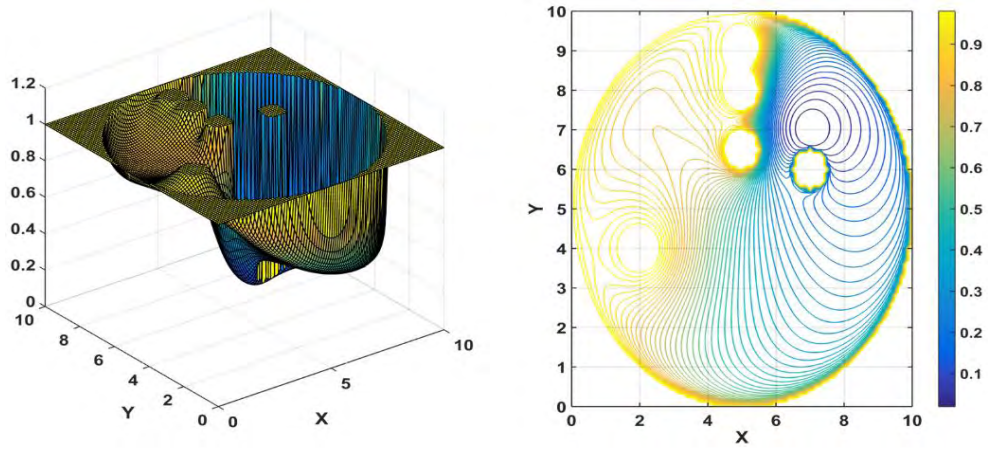
$$B(q) = B_0(q)\prod_{i=0}^{n} B_i(q), \qquad B_0(q) = -\|q - q_{tar}\|^2 + {r_0}^2$$

$$B_i(q) = \|q - q_i\|^2 + {r_i}^2. \tag{72}$$

Donde k es un parámetro de sintonización para suavizado, n es el número de obstáculos presentes en el escenario, r_0 es el radio de la función total ubicada en q_0 y r_i el radio de cada obstáculo ubicado en q_i. Cabe mencionar que, la sintonización de estos parámetros es fundamental para definir una buena relación inicio llegada, tal que, sea posible el cálculo de gradiente descendiente que conduzca a la obtención de la solución.

La Figura 12 muestra la representación de esta función para un arreglo general de obstáculos.

Figura 12. Función de navegación con cinco obstáculos estáticos.



Obstáculos en las coordenadas: (2,4), (5,6.5), (5,8), (5,9.1), (7,6) y la meta ubicada en (7,7). Con un valor de parámetros igual a: k=4.5, r_0=5 y q_0= (5,5). **Fuente:** Esta investigación.

Para el desarrollo de esta comparación se ha escogido este enfoque global para generar la función de campo potencial artificial. Sin embargo, el enfoque local presentado se utiliza para la creación de la función de aptitud como aporte al método de enjambre de partículas, como se verá posteriormente.

2.2.3. Método de Descomposición por celdas trapezoidales (CD)

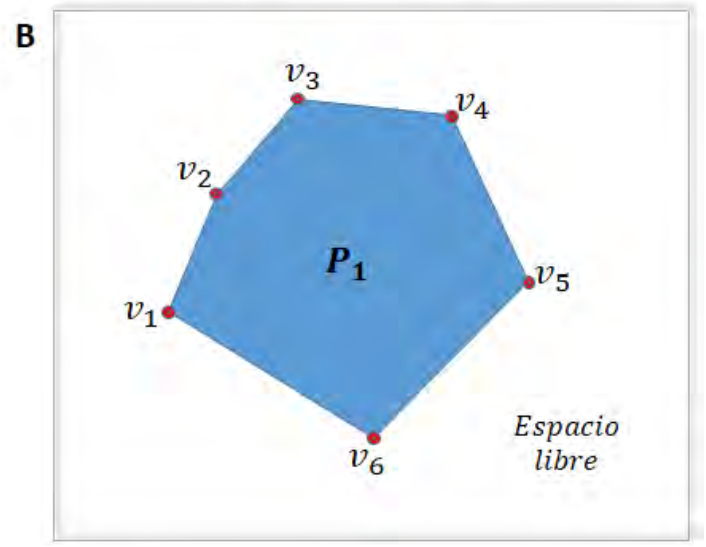
Para realizar la comparación de estrategias se ha escogido el método por descomposición en celdas trapezoidales, porque no solo aplica la teoría de grafos para resolver el problema de planeación sino que también divide el espacio de trabajo en subconjuntos convexos libres de colisión. De esta manera, permite identificar todas las regiones accesibles (libres de colisión) y siempre determinar una ruta segura (solución) si es que existe alguna.

Este método recurre a la descomposición trapezoidal del escenario, como se presenta en [20] y [39], distinguiendo entre obstáculos y posible área de recorrido. Básicamente, esta técnica busca generar trayectorias en el espacio libre encontrado.

Para ello, se parte de la representación del espacio de trabajo, es decir, de la representación de los objetos y el espacio delimitador con polígonos:

Sea P_i un conjunto en R^2 que representa el i-ésimo objeto en el espacio definido por los vértices del polígono delimitador ordenados en dirección de las manecillas del reloj y *B* un conjunto en R^2 definido por los límites del espacio de trabajo tal y como se aprecia en la Figura 13.

Figura 13. Definición de los objetos y el espacio de trabajo en CD.



Polígono $P_1 = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. **Fuente:** Esta investigación.

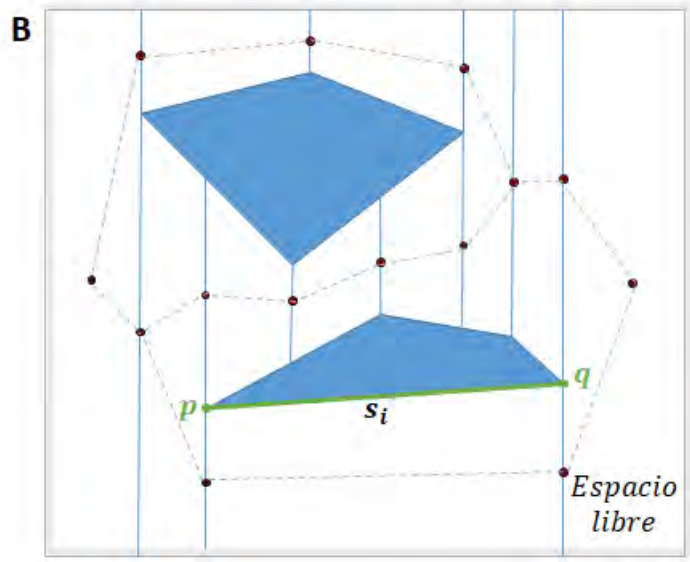
Entonces, el espacio libre para un robot puntual, se define como:

$$C_{free} = B \setminus \cup_{i=1}^{t} P_i \tag{73}$$

(Esta definición se aplica a un robot representado solo por un punto, pero se puede considerar la forma del robot cuando se define un espacio de trabajo con ayuda de las sumas de Mincouski [70]).

Lo que se busca con el algoritmo es generar “Celdas” en el espacio libre, extendiendo rectas verticales sobre los vértices de cada objeto del escenario. Estas extensiones se detienen si interceptan una arista del grafo o los límites del espacio de trabajo, para posteriormente crear un mapa de ruta con ayuda de las relaciones de adyacencia de las celdas, como lo muestra la Figura 14.

Figura 14. Descomposición trapezoidal, mapa de conexiones y definición de un segmento de línea.



Fuente: Esta investigación.

Tal objetivo se logra generalmente con la ayuda de un algoritmo de barrido de línea [56] (Se puede encontrar una interesante aplicación del algoritmo para el método de descomposición por celdas en [43]), que identifican los vértices del grafo barriendo una extensión vertical de un extremo a otro. Construye bien el mapa, pero no ofrece un método de búsqueda de puntos dentro de este. Por otro lado, el algoritmo descrito en los capítulos 6 y 13 de [4], genera además del mapa una estructura de búsqueda, mejorando el tiempo de computación y permitiendo ubicar al robot en el mapa. Por lo tanto, es el que se ha implementado cuando la posición del grafo es general (esto se explicará más adelante).

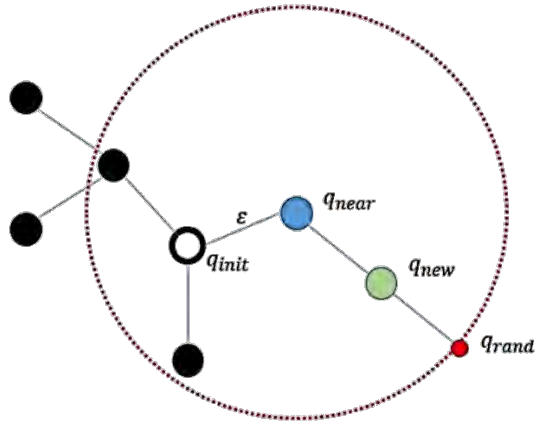
En la Sección 3.1.3 se presentan los pormenores del método incluyendo algunas funciones que se han propuesto como una interpretación y algunas aclaraciones. Una explicación mucho más detallada del método se encuentra en [4].

2.2.4. Método de búsqueda aleatoria. (RRT*)

Dentro de los enfoques de búsqueda aleatoria se ha escogido la variante del algoritmo de exploración rápida de árboles aleatorios propuesta en [30] y [31]. Esta técnica denominada RRT* mantiene la ventaja de completitud probabilística de RRT y hace que el costo de la solución encontrada converja casi seguramente al óptimo. Por lo tanto, es una de las variantes más prometedoras como método de planificación autónomo.

El algoritmo se basa en la construcción de un árbol de configuraciones que crece explorando a partir de una posición de partida. La Figura 15 muestra el esquema de expansión del algoritmo RRT, cuya operación es similar en RRT*.

Figura 15. Esquema de expansión de RRT.



En esta gráfica, q_{int} es la posición de partida del robot, ε es una distancia de avance predefinida (longitud de cada rama), q_{rand} es el nodo escogido aleatoriamente en el espacio de configuraciones, q_{near} es el nodo más próximo a q_{rand} y q_{new} es el nuevo nodo a agregarse a las ramificaciones existentes en dirección a q_{rand} . **Fuente:** [17].

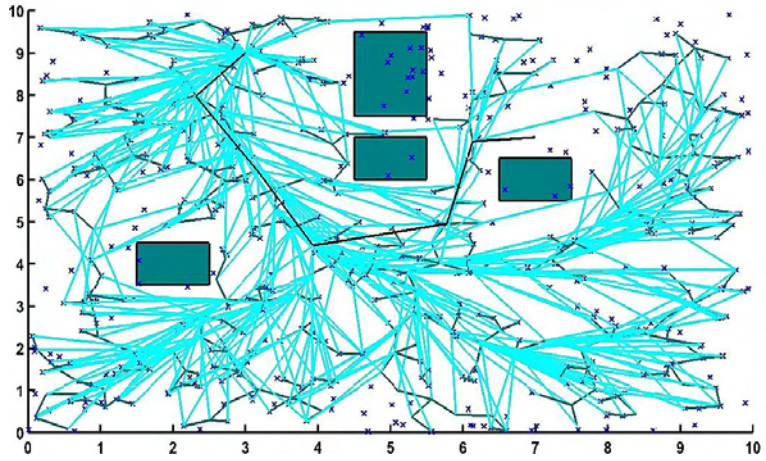
El algoritmo RRT arranca con la inicialización de una estructura de nodos con la posición de partida. Posteriormente se inicia el bucle principal en donde destacan tres instrucciones: la primera es seleccionar un punto al azar en el espacio de configuraciones q_{rand}, la segunda es identificar el punto más próximo a dicha posición aleatoria q_{near}, tal que pertenezca a las ramas existentes, y la tercera es avanzar para ramificarse desde el punto más próximo escogido hasta q_{rand} o hacia un nuevo nodo q_{new} cuando la distancia de avance predefinida no permite llegar hasta q_{rand}, garantizando que en el avance no existe colisión. Este bucle se cumple hasta que se llegue a un máximo permitido de iteraciones.

El algoritmo RRT* ejecuta las mismas tres instrucciones de RRT con la particularidad de buscar un nodo aún más cercano a la posición aleatoria que el punto más próximo que se encuentra en la versión original. Esto se realiza en un

radio de búsqueda r con centro en q_{new}. Así, se generan muchas más ramas en el árbol de exploración que en consecuencia definen un mapa de ruta más completo y robusto. Finalmente, cuando ya se ha creado el mapa de ruta, es decir, cuando se completa un máximo número de nodos Nod_{max} definidos por el programador, se aplica el algoritmo A* para encontrar la trayectoria bajo el criterio de menor distancia recorrida. Este algoritmo es de búsqueda en grafos, es completo y por lo tanto si la solución existe en el mapa de ruta siempre la encontrará con el menor coste entre un nodo de partida y uno de llegada. Consiste en la evaluación de una función que tiene en cuenta el valor heurístico de los nodos y el coste real del recorrido (Ver anexo 2).

En la Figura 16 se muestra la ruta planeada por RRT* para un arreglo general de obstáculos.

Figura 16. Ruta planeada por método RRT*



En línea negra se muestra la ruta que brinda el algoritmo A*, en azul oscuro los puntos aleatorios de cada iteración y en color cian los segmentos de recta que forman el mapa de ruta. El número máximo de nodos es 500 con un avance de 0.5 y un radio de búsqueda de nodos aún más cercanos igual a 2 **Fuente:** Esta investigación.

2.2.5. Método de optimización por enjambre de partículas (PSO)

El método de optimización por enjambre de partículas se selecciona para comparación debido a sus múltiples aplicaciones en problemas de optimización y por ser un método representativo de bio-inspiración basado en comportamiento de enjambres. Aquí se usa como una alternativa a la técnica de gradiente descendente en el método de campos potenciales.

PSO es un algoritmo de búsqueda multi-agente inspirado en el comportamiento de enjambres de insectos, pájaros y peces, descrito en [33]. Existen acercamientos a la planificación de trayectorias, por ejemplo en [52], dónde se busca optimizar la posición de los puntos de control que definen una ruta y como se verá más adelante, también es usada como técnica de optimización de velocidades en el método RVO. La naturaleza del problema de planificación de trayectorias, puede entenderse también como un problema de optimización, pues lo que se busca es encontrar rutas óptimas, que conecten dos puntos en un espacio en el que se presentan obstáculos.

Para orientar este método a la generación de trayectorias, se ha considerado a cada individuo de una población (enjambre) de tamaño n, como una partícula ubicada el vector posición $x_i(t) \in \mathbb{R}^2$, que se mueve iterativamente en el plano cartesiano con una velocidad dada. Tal movimiento está dado por la ecuación (74).

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{74}$$

Así como, por ejemplo, algunos animales suelen dejar rastros químicos para informar a otros sobre la abundancia de alimentos o alertar de posibles peligros, y tal rastro puede ser sentido en cada punto o zona del espacio con distinta concentración por dichos animales; así mismo el método PSO pretende simular el comportamiento animal para optimizar funciones o procesos. A ese tipo de rastro químico, o demás forma de comunicación, se lo representa con una función de aptitud $F(x_i(t))$ que toma, en éste caso, el vector posición $x_i(t)$ y le asigna un valor numérico.

Siguiendo la analogía propuesta, el comportamiento de cada individuo estará relacionado con la función de aptitud, haciéndolo alejar de las “zonas de concentración”, si la señal percibida es identificada como un peligro, o acercarse a ella, si se trata de un beneficio. Este tipo de movimiento, visto como una velocidad en la ecuación (74), está definido por la ecuación (75) para el *i-ésimo* individuo.

$$v_i(t+1) = v_i(t) + \varphi_1(P_{ibest} - x_i) + \varphi_2(P_{gbest} - x_i) + \varphi_3(P_{nbest} - x_i) \tag{75}$$

Como se ve, la velocidad con la que viajaba el individuo se redirige hacia las posiciones P_{ibest}, P_{gbest} y P_{nbest}, que son: la posición en la que el individuo tuvo el mejor resultado en la función de aptitud, la posición en la que un individuo de todo el enjambre, tuvo el mejor resultado en la función de aptitud y la posición en la que un individuo de un vecindario, tuvo el mejor resultado en la función de aptitud, respectivamente. Lo que simboliza es que un individuo de un enjambre se mueve hacia el lugar en el que mejores resultados tiene (factor cognitivo), pero no solo influye su perspectiva en la decisión de hacia dónde dirigirse, P_{gbest} y P_{nbest} representan la influencia social al tomar tal decisión. La velocidad en la iteración anterior del individuo se considera para el cálculo de la nueva velocidad, como un término que representa la costumbre o la inercia.

φ_1, φ_2 y φ_3 son valores de una distribución aleatoria uniforme entre [0,1] que representan los factores cognitivos y sociales que influyen en cada individuo,

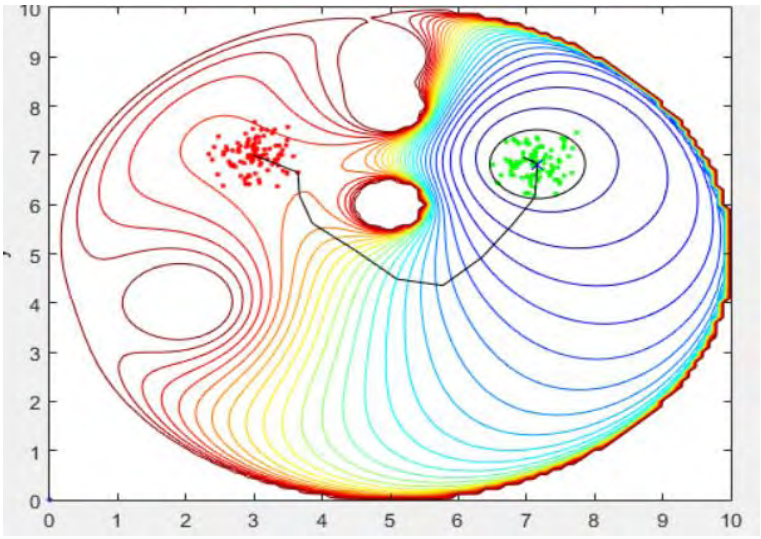
tratando de elegir de manera aleatoria, a que factor se le da más prioridad en cada iteración. Los vecinos de cada individuo, son los *m* individuos que están más cercanos a él.

Por último, las "zonas de concentración" de las que el individuo debe escapar son los obstáculos, y la meta, a las cuales es atraído. Esto implica una función de aptitud, con puntos máximos en las zonas ocupadas por obstáculos y mínimo global en la meta, así las partículas se dirigirán a éste mínimo. Los pasos que se proponen para ejecutar el algoritmo desde la perspectiva de la planificación de trayectorias, asumiendo, son los siguientes:

1. Inicializar las *n* partículas en posiciones aleatorias $x_i(t)$ dentro de un círculo de radio *r* (radio de dispersión) con centro en la posición del móvil, o del punto de control inicial de la trayectoria, a cada partícula se le asigna una velocidad $v_i(t)$ también aleatoria dentro de un rango.
2. Se evalúa $F(x_i(t))$ para cada individuo.
3. Para cada partícula **Si** $F\big(x_i(t)\big) < F(P_{ibest})$ **entonces** $P_{ibest} = x_i(t)$.
4. Se evalúa la partícula con mejor función de aptitud de todo el enjambre $x_{best}(t)$, **Si** $F\big(x_{best}(t)\big) < F\big(P_{gbest}\big)$ **entonces** $P_{ibest} = x_i(t)$.
5. Se evalúa la partícula con mejor función de aptitud de todo el enjambre $x_{best}(t)$, **Si** $F\big(x_{best}(t)\big) < F\big(P_{gbest}\big)$ **entonces** $P_{ibest} = x_i(t)$.
6. Se asignan valores a los φ y se calculan las nuevas posiciones con ayuda de las ecuaciones (74) y (75)
7. La nueva posición del móvil, (o del punto de control siguiente), se calcula como el de cualquier otra partícula.
8. A medida que el móvil (o punto de control se desplaza), el radio de dispersión se desplaza con él, por ello, las partículas que queden fuera del círculo, deben ser reasignadas dentro del mismo en la siguiente iteración.
9. En cada iteración las partículas con los *c* peores resultados en cuanto a la función de aptitud, son reasignadas a nuevas posiciones aleatorias.
10. Se repite desde el paso 2.

La función de aptitud a la que nos referimos, ya existe y es la misma que se genera en el método de campos potenciales, por ello se dice que esta técnica es una alternativa al uso de gradiente descendente. La Figura 17 muestra una trayectoria generada con éste método.

Figura 17. Ruta planeada por método PSO.



En color negro se muestra la trayectoria de referencia planeada en un entorno de arreglo general de obstáculos modelado con campos de potencial artificial local del cual se indican las curvas de nivel. En color rojo la población inicial agrupada y en color verde las últimas posiciones de los individuos del enjambre. El valor de los parámetros es: n=100 y una velocidad máxima de 0.01 **Fuente:** Esta investigación.

2.3. PLANEACIÓN DE TRAYECTORIAS PARA MÚLTIPLES ROBOTS

2.3.1. Introducción

Cuando se refiere al problema de planeación de trayectorias en ambientes dinámicos se consideran dos casos de estudio: el primero consiste en planear y controlar el movimiento de un robot desde una posición inicial hacia un objetivo en movimiento, evitando al mismo tiempo los obstáculos dinámicos [17]. El segundo, considera la presencia de múltiples robots navegando en un mismo entorno, abordando el estudio de evitación de colisión entre robots y obstáculos que pueden o no estar en movimiento [25]. Para los objetivos de esta investigación, se trabaja en el segundo caso de estudio considerando robots homogéneos de tracción diferencial que navegan en un entorno con obstáculos estáticos.

Definición del problema:

Para el caso de múltiples robots, el problema no se resuelve simplemente construyendo un camino geométrico puesto que se necesita que el tiempo se tome en cuenta como una dimensión más dentro del espacio de configuración. Además se debe considerar la capacidad racional y de reacción de cada uno de los robots,

que dentro de este caso de estudio se llaman agentes. En consecuencia, al problema inicial de planear una trayectoria debe añadirse el análisis de cooperación y/o tiempo entre agentes que cumplen un objetivo común. Para este caso, el objetivo es navegar dentro de un mismo entorno.

Brindar una solución a este problema puede tener dos enfoques: uno centralizado en una estación base y otro descentralizado en los múltiples agentes que están navegando.

a) Enfoque centralizado

Las iniciativas centralizadas son provechosas cuando el número de robots no es grande. En este caso, se calcula la trayectoria de forma global y posteriormente es transmitida a cada uno de los robots, quienes asumen distintos roles u órdenes de prioridad al momento de seguir las rutas planeadas. Este enfoque es útil para aplicaciones fuera de línea, es decir, no adaptables para cálculos en tiempo real. Un ejemplo de este tipo de planeación en múltiples robots se encuentra en [16] que muestra la metodología de seguimiento del líder (órdenes de prioridad) para formación de equipos de robots.

El principal inconveniente del enfoque centralizado es cuando se presenta un problema en el procesamiento central, ya que todo el sistema se detiene.

b) Enfoque descentralizado

Las iniciativas descentralizadas son mucho más robustas y útiles cuando el número de robots es elevado. En este enfoque cada robot se considera como un agente inteligente capaz de planear su propia trayectoria siendo consciente de su entorno y los demás agentes involucrados en él [75].

La tarea de navegación es dividida y por lo tanto si uno de los sistemas falla, no necesariamente implica que todo el conjunto de sistemas se detenga.

Para la solución del problema distribuido, se realiza una “extensión” de los métodos clásicos de planificación de trayectorias estudiados hasta ahora. Entonces, se implementan ajustes necesarios para la aplicación del algoritmo en ambientes con más de un agente, quienes realizan distintas trayectorias para la ejecución de tareas similares, complementarias o completamente diferentes. A tal tarea se le llama “evitación”, entendiendo, que la interacción entre otros agentes solo tendrá el objeto de garantizar que no se choquen los unos a los otros. La aclaración se debe a que existen algoritmos dedicados, por ejemplo, a la creación de formaciones de robots o a que estos asimilen el comportamiento de enjambres de insectos.

Por esta razón, además de la extensión que refiere a los campos de potencial y la descomposición por celdas, se presenta el método de obstáculos de velocidad recíproca como representante de las técnicas más famosas en evitación de colisión.

2.3.2. Extensión método campos potenciales artificiales

Para extender el método de campos potenciales para múltiples robots, se emplea una técnica intuitiva. Aunque la mayoría de las investigaciones se enfocan en la formación de robots (dado lo práctico que resulta utilizar las fuerzas de atracción y repulsión) se puede observar que en cada una de esas técnicas es inherente el problema de evasión. Por ejemplo, en [58], [67], se añade una fuerza de repulsión entre los robots para conseguir la formación deseada, o bien en [38], donde tales fuerzas son necesarias para el logro de metas comunes.

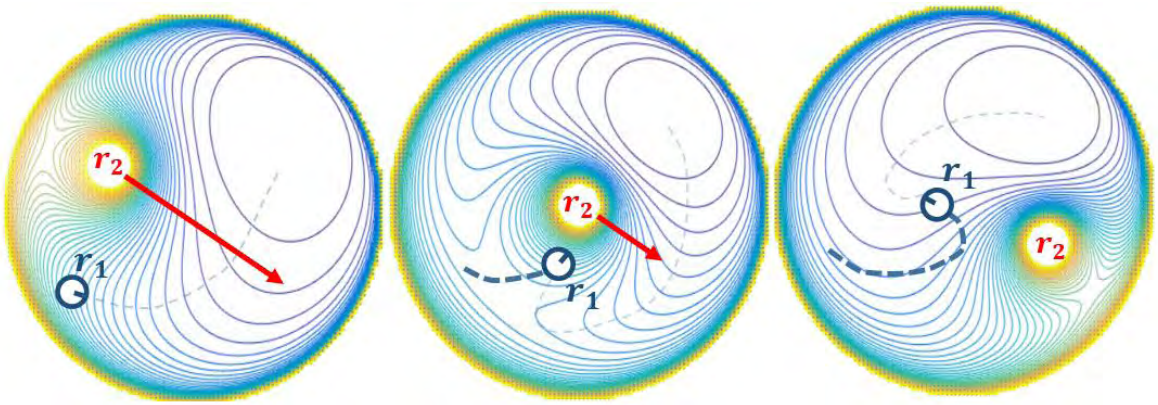
De esta manera, el principal enfoque para la evasión son las fuerzas repulsivas entre los mismos robots, en donde en cada posición del vehículo se aplica un nuevo potencial al ya existente para el escenario. Aquí se ha aplicado tal criterio con una interpretación particular, así:

Sea *W* un espacio de trabajo delimitado, que contiene *n* objetos O_i y *m* robos r_j, donde $i \in \mathbb{Z}\, tales\ que\ 0 < i \leq n$ y $j \in \mathbb{Z}\, tales\ que\ 0 < i \leq m$.

Supóngase que cada robot r_j, tiene como meta una posición $p_{goal\ j}$ en *W,* entonces, es posible generar un campo de potencial artificial para cada r_j, basado en los objetos y la posición meta, con el método sustentando en la Sección 2.2.2.

Si solo el robot r_1 se desplaza en *W*, los otros robots r_j con $j \neq 1$, pueden ser tomados como objetos estáticos y por ello, la fuerza de repulsión que ejercen sobre r_1 es la misma que la asignada a los obstáculos O_i durante la creación del campo artificial. Ahora, si los robots r_j se encuentran en movimiento, basta con considerar un campo de potencial variable, dependiente de la posición de los "objetos móviles" que representan a cada r_j, un ejemplo se observa en la Figura 18.

Figura 18. Planificación de trayectoria para robot r_1 en un campo de potencial variable debido al robot r_2.



Fuente: Esta investigación.

En la Figura 18 es posible observar como este método garantiza la evasión, tal y como garantiza la evasión de obstáculos estáticos. Sin embargo, como se vio en la Sección 2.2.2., agregar objetos al cálculo de la función de potencial total requiere reajustar los parámetros de definición del campo para lograr una relación óptima de gradiente descendente.

2.3.3. Extensión método de descomposición por celdas trapezoidales

La extensión del método de descomposición por celdas trapezoidales al caso de múltiples robots, tiene dos enfoques generalmente.

Puede verse como un problema dimensional al agregar el tiempo como tercera dimensión (o como cuarta si la navegación es en 3D, [74]). Esto se logra con la creación de celdas multidimensionales en el espacio libre, pero con el problema que el grafo debe cumplir con "requisitos de linealidad" y necesita usar el método clásico, ya no de líneas, sino de planos de barrido. Todo esto aumenta considerablemente la complejidad de comprensión y ejecución del algoritmo. Además, implica que se conozcan las rutas a seguir de cada robot para poder "curvar" las trayectorias en el tiempo, entendiendo así a los robots, no como agentes sino como objetos dinámicos del entorno. Por otro lado, la complejidad a la que se hace referencia no garantiza la mayor exactitud, puesto que las nuevas curvas creadas en el tiempo tienen los mismos problemas que las creadas en dos dimensiones, como se observará posteriormente en el capítulo 5.

Por otro lado, en [26] se propone un enfoque para la inclusión de más agentes sin necesidad de cambiar las dimensiones al descomponer el grafo, y sin perder las ventajas de garantizar la evasión entre robots. A continuación se explica este enfoque y las modificaciones que aquí se realizaron para implementarlo.

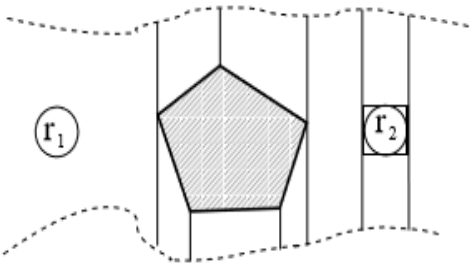
Sea *W* un espacio de trabajo delimitado que contiene *n* objetos O_i y *m* robots r_j, donde $i \in \mathbb{Z}\ tales\ que\ 0 < i \leq n$ y $j \in \mathbb{Z}\ tales\ que\ 0 < i \leq m$.

Supóngase que se quiere trazar una ruta para el robot r_1. Entonces, existen solo dos estados para los otros robots r_j con $j \neq 1$:

i) Estáticos
ii) En movimiento

Cuando los otros robots están en el estado estático, por ejemplo r_2, no son más que objetos para r_1, y simplemente son tratados como áreas prohibidas en el espacio de trabajo al descomponer el grafo. Esto se muestra en la Figura 19.

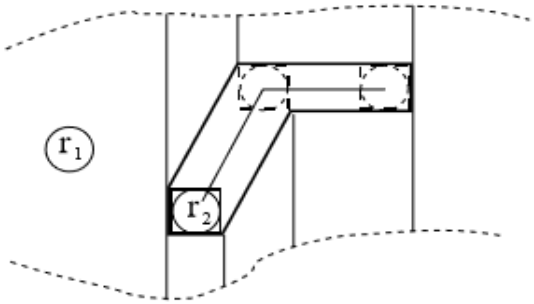
Figura 19. Ejemplo en el que el robot r_2, es tomado como un objeto más en la descomposición trapezoidal.



Fuente: [26].

Cuando los robots se encuentran en movimiento, la idea es utilizar el espacio ocupado por los robots r_j al avanzar por sus trayectorias como un objeto para r_1, al que se denomina "*Path Obstacle*" u Objeto Camino (Figura 20). Con lo anterior, es posible generar un camino seguro para r_1, pero hasta aquí, el método es bastante ineficiente dado que largas trayectorias de r_2 generan obstáculos demasiado grandes, y por ello, tramos de ruta innecesarios. Para mejorar la eficiencia, se da a los Objetos Camino, la propiedad de poder generar trayectorias a través de ellos. Esto implica la necesidad de una técnica para evitar la colisión en las intersecciones.

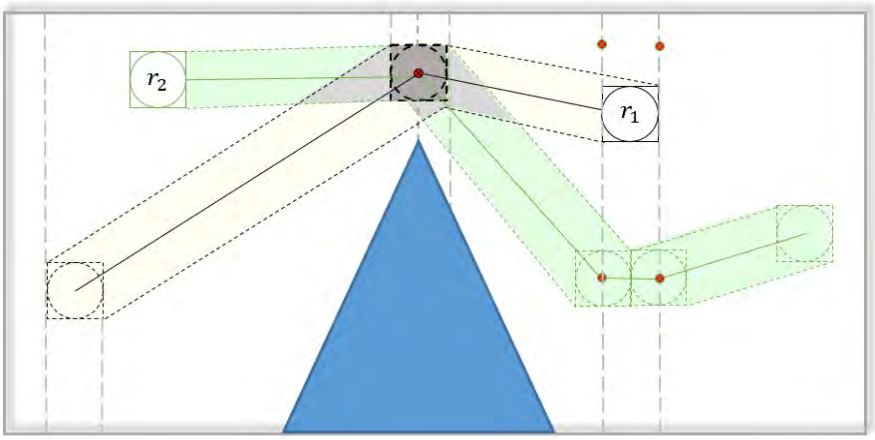
Figura 20. Descomposición del grafo para el Objeto Camino, referente al espacio tomado por r_2, al desplazarse por una trayectoria.



Fuente: [26].

Ahora bien, cuando se planea rutas de varios robots con este enfoque, se realiza con una aplicación sucesiva. Es decir, primero se toma a r_1 teniendo en cuenta solo obstáculos, luego r_2 teniendo en cuenta r_1 y los obstáculos, luego r_3 teniendo en cuenta r_1, r_2 y los obstáculos, etc. Es muy probable que los Objetos Camino se traslapen o intersectan, debido a que se ha permitido generar trayectorias sobre ellos y la trayectoria generada crea un nuevo Objeto Camino, como se muestra en el ejemplo de la Figura 21.

Figura 21. Objetos Camino que se traslapan.



Nótese que no se generan nodos para r_2 en las extensiones verticales de la zona sombreada más oscura puesto que considerando la forma, no existe un camino libre de obstáculos. **Fuente:** Esta investigación.

Para que los robots no sufran colisiones en las zonas de intersección, se proponen tres reglas que regulan el tránsito de cada robot en dichas zonas:

1. Un robot *r*, solo puede transitar en una zona de intersección, siempre y cuando éste la "reserve", es decir la acción de "reservar" otorga al robot el permiso de atravesar la zona.
2. Si un robot *r,* que viaja en una trayectoria planeada, entra a una zona de intersección, la "reserva".
3. Una zona de intersección solo puedo ser "reservada" por un robot a la vez.
4. Un robot no puede asignar una zona de intersección, sin antes haber liberado la que previamente había asignado.

Al aplicar solo estas reglas se puede probar que no existirán colisiones y se garantiza la evasión entre robots [26].

En la Sección 3.1.3 se explica este enfoque y las modificaciones que aquí se realizaron para implementarlo.

2.3.4. Obstáculos de velocidad recíproca (RVO)

El método RVO presentado en [28] es una modificación al algoritmo de obstáculos de velocidad propuesto en [15]. Consiste en la selección de maniobras para evitar obstáculos estáticos y dinámicos en un espacio de velocidad que se crea teniendo en cuenta las posiciones y velocidades de los objetos y los agentes presentes en un mismo entorno. La noción de obstáculo de velocidad (VO, por sus siglas en inglés), que no es más que una simple representación geométrica de potenciales

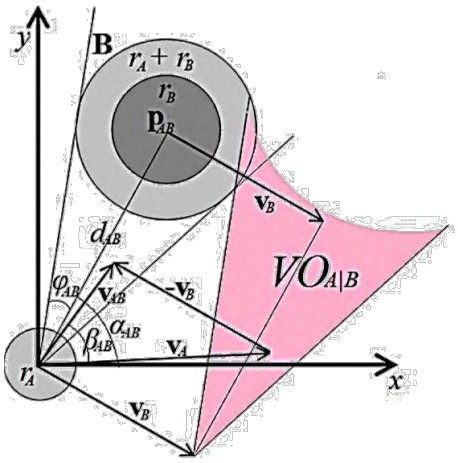
maniobras de evitación comprende el conjunto consistente de todas las velocidades V_A de un robot A que resultará en alguna colisión en algún momento en el tiempo con un obstáculo B moviéndose a una velocidad V_B.

Las maniobras de evitación se generan seleccionando velocidades por fuera de los obstáculos de velocidad. Para garantizar que estas maniobras sean admitidas por una plataforma con restricciones cinemáticas, las velocidades seleccionadas se intersectan con el conjunto de velocidades admisibles que son determinadas por las restricciones de movimiento que determina el tipo de plataforma robótica.

La Figura 22 muestra el obstáculo de velocidad (definido geométricamente) del robot circular A, localizado en p_A, de radio r_A y con velocidad V_A, introducido por el robot circular B, localizado en p_B, de radio r_B y con velocidad V_B.

Como lo expresa el autor en [2], considerando al robot A como un punto en el origen y B como un disco centrado en p_{AB} con radio igual a $r_A + r_B$, si B es estático se puede definir un cono C de velocidades que conduzcan al robot A hacia el robot B como el conjunto de rayos disparados desde p_A que cruzan el límite de B. Entonces, para derivar el $VO_{A|B}$ (obstáculo de velocidad de A introducido por B) se traslada el cono C con la velocidad v_B de B como se ve en la Figura 22 en color rosado.

Figura 22. Representación geométrica del obstáculo de velocidad del robot A impuesto por el robot B.



Fuente: [2].

Escrito formalmente

$$VO_{A|B} = \{v | \exists t > 0 :: p_A + t(v - v_B) \in B\} \quad (76)$$

Visto de esta manera, las rutas generadas presentan oscilaciones significativas debido a la escogencia de velocidades aleatorias por fuera de cada obstáculo de velocidad. RVO responde ante este inconveniente, teniendo en cuenta el comportamiento mutuo de los dos robots implicados en la creación del VO. En lugar

de escoger una velocidad aleatoria para cada robot, se escoge una nueva velocidad igual al promedio de las velocidades que lleva cada agente y las velocidades aleatorias por fuera del VO. De este modo la representación general tiene la forma:

$$RVO_{A|B} = \{v|\exists t > 0 \ :: \ p_A + t(v - (1-\alpha) * v_A - \alpha * v_B) \in B\} \tag{77}$$

donde α determina el esfuerzo compartido entre los robots. En [28] el valor de este parámetro está entre 0 y 1.

Para el desarrollo de esta investigación, se trabaja con la versión de RVO propuesta en [1] y [2], que permite el uso de algoritmos de optimización como el de abejas (ABC, por sus siglas en inglés) y PSO de optimización estocástica.

Según esta propuesta, el algoritmo inicia calculando la distancia euclidiana y el argumento entre los robots respecto a las coordenadas cartesianas

$$dist_{AB} = \|p_A - p_B\| \qquad \alpha_{AB} = atan2(y, x) \tag{78}$$

El valor de $dist_{AB}$ y su argumento α_{AB} son el eje cónico de simetría del VO. De esta manera el semi-ángulo de la base del cono denotado con φ_{AB} puede calcularse aplicando

$$\varphi_{AB} = \sin^{-1}\frac{r_A + r_B}{dist_{AB}} \tag{79}$$

Con los parámetros V_B, $dist_{AB}$, α_{AB} y φ_{AB} se revisa las posibilidades que tiene el robot A de entrar en colisión con el robot B calculando la diferencia de velocidades entre los dos robots V_{AB} y su argumento respecto las coordenadas cartesianas β_{AB}, aplicando

$$\begin{gathered} V_{AB} = \left(V'_A - \frac{V_A + V_B}{2}\right) = [x, y] \\ V_{AB} = \|V_{AB}\| = \sqrt{x^2 + y^2} \\ \beta_{AB} = atan2(y, x) \end{gathered} \tag{80}$$

*Donde $atan2$ es una función propia de lenguajes de programación que permite encontrar el ángulo evaluando los valores de seno y coseno de y/x por separado. Dicho de otra manera, es la versión de 4 cuadrantes de $atan$.

Lo anterior, asumiendo una nueva velocidad para el robot A que es escogida de forma arbitraria y denotada con V'_A. Esta velocidad está dentro de $RVO_{A|B}$ cuando la diferencia absoluta entre α_{AB} y β_{AB} es menor que φ_{AB}. Luego que asumiendo

$$\begin{gathered} \text{Sea: } \Psi_{AB} = \left|\beta_{AB} - \alpha_{AB}\right| \\ \begin{Bmatrix} V'_A \in RVO_{A|B} \ si \ \Psi_{AB} \leq \varphi_{AB} \\ V'_A \in RVO_{A|B} \ si \ \Psi_{AB} > \varphi_{AB} \end{Bmatrix} \end{gathered} \tag{81}$$

Si V'_A esta dentro del obstáculo de velocidad recíproca, es obligatorio cambiar su magnitud y/o dirección para escapar de la colisión. En este orden de ideas, se procede al cálculo del tiempo mínimo de posible colisión t_c dado por

$$t_c = \frac{dist_{AB}\cos\Psi_{AB} - \sqrt{(r_A + r_B)^2 - dist^2{}_{AB}\sin\Psi_{AB}{}^2}}{V_{AB}} \quad (82)$$

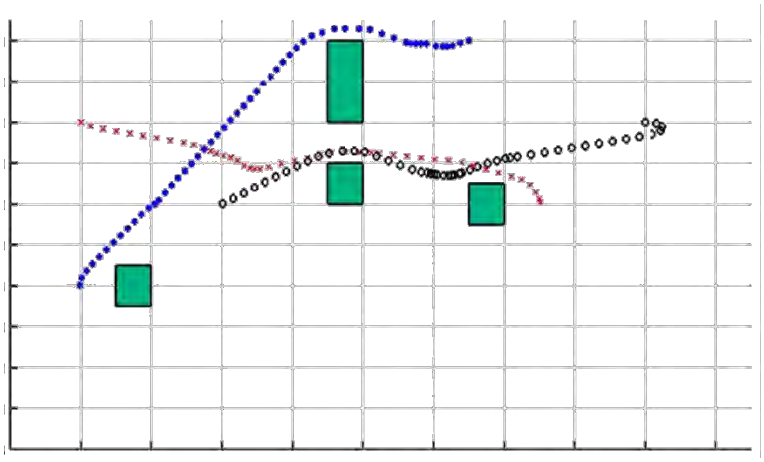
Este tiempo debe compararse con el tiempo de muestreo de la simulación denotado con τ, ya que la colisión es inminente si τ > t_c. En caso contrario, la colisión no ocurrirá en la siguiente iteración, sin embargo, aún existe probabilidad de colisión y por lo tanto debe aplicarse una fórmula de penalidad que permita elegir la mejor velocidad de entre un grupo de velocidades candidatas. El autor en [1] plantea la siguiente fórmula de penalidad

$$p(V'_A) = \frac{k}{t_c} + \|V^G{}_A - V'_A\| \quad (83)$$

Aquí $V^G{}_A$ es la velocidad dirigida hacia la meta del robot A y k es una constante. La velocidad óptima llamada $V_A{}^*$ será la velocidad con la menor penalidad, es decir, cercana a la velocidad dirigida a la meta y por fuera del $RVO_{A|B}$.

$V_A{}^*$ puede optimizarse aplicando algoritmos tipo enjambre como PSO o colonia de abejas (ABC) como se observa en [1] y [2], respectivamente (Ver anexo 4); siempre sujetos a las restricciones no holonómicas del robot A. En la Figura 23 se aprecia las trayectorias generadas utilizando este método para tres robots en un entorno de arreglo general de obstáculos.

Figura 23. Ruta planeada por método RVO para tres robots homogéneos de tracción diferencial.



Como técnica de optimización se aplica PSO. En color azul, rojo y negro las rutas trazadas para los robot 1, 2 y 3 respectivamente. En color verde, la representación de obstáculos estáticos. **Fuente:** Esta investigación.

3. PROCEDIMIENTOS PARA IMPLEMENTACIÓN DE ALGORITMOS

3.1. METODOLOGÍAS PROGRAMADAS PARA PLANEACIÓN DE TRAYECTORIAS

En esta sección se indica los escenarios programados en Matlab, para la validación de cada método y las modificaciones requeridas para orientar sus nociones a la planeación de trayectorias, para uno y/o múltiples robots.

Los entornos de navegación simulados son: Trampa, Pasaje Estrecho y Arreglo General de obstáculos. En el primero los obstáculos rodean el punto inicial impidiendo una línea directa hacia el objetivo. Puede generar mínimos locales y requiere de rutas que rodeen la totalidad del objeto, incluso alejándose de la meta. En el segundo, a pesar de tener una ruta directa es posible que no se generen trayectorias seguras o que tengan un alto grado de colisión con los obstáculos y el tercero es el caso general y permite establecer la adaptabilidad de las metodologías. La justificación de su utilización se expone en metodología de comparación analítica Sección 3.3.

Dichos escenarios ofrecen gran dificultad al momento de planear una trayectoria, bien sea por la dimensión y ubicación de los obstáculos o por limitar el rango de solución a una pequeña porción del espacio.

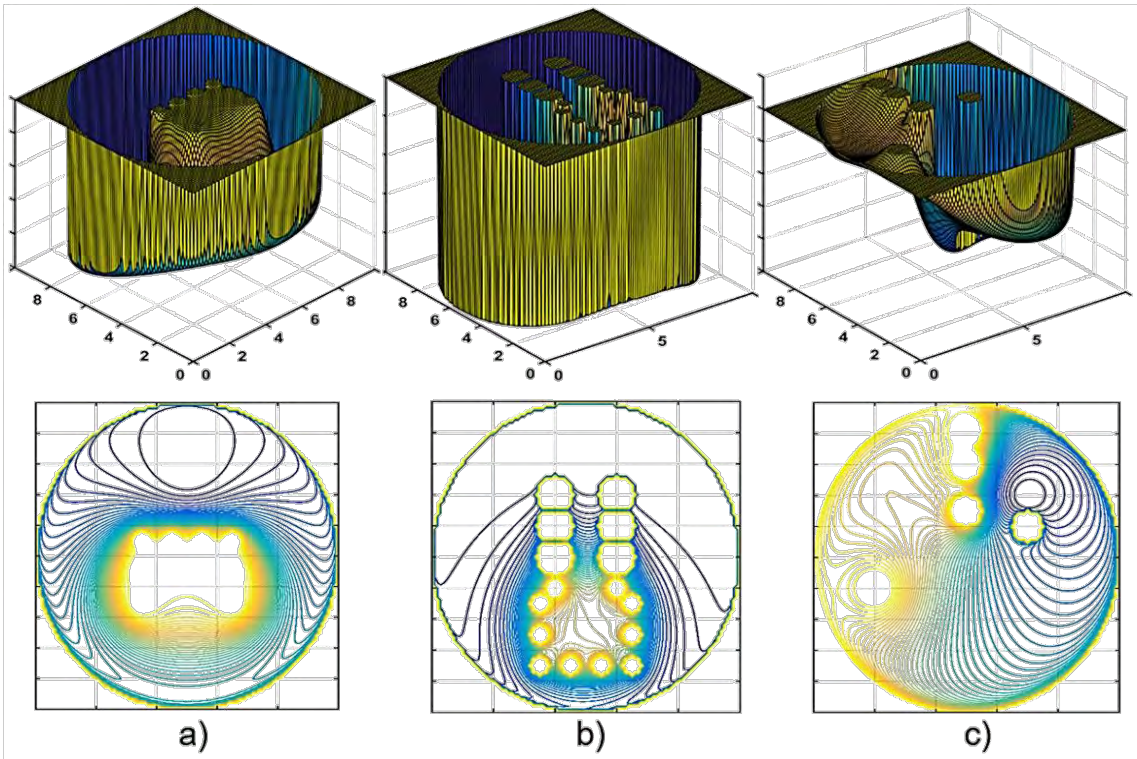
Cada uno de estos escenarios para las validaciones virtuales, considera una región cuadrada de diez metros por diez metros en donde cada unidad de división equivale a un metro. Para validaciones virtuales de múltiples robots, se opta por un arreglo de obstáculos modificado con el fin de facilitar la implementación de la técnica de campos de potencial artificial.

3.1.1. Función de navegación para campos potenciales artificiales caso único robot

Dado que el fundamento de esta técnica se orienta a la planeación de trayectorias, para la implementación virtual, y en el caso de un único robot, no se realizaron modificaciones a su propuesta teórica fundamental.

La Figura 24 muestra la representación tridimensional obtenida y las curvas de nivel de cada uno de los escenarios programados.

Figura 24. Representación de escenarios de prueba para el método de campos potenciales artificiales.



Escenario tipo trampa b) Escenario tipo pasaje estrecho c) Escenario tipo arreglo de obstáculos. **Fuente:** Esta investigación.

3.1.2. Optimización por enjambre de partículas

La optimización por enjambre de partículas tiene múltiples aplicaciones dentro del marco de la optimización. Esta técnica orientada a la planeación de trayectorias, considera a cada individuo de la población enjambre como una solución que se representa mediante un vector de posición $x_i(t)$ movilizado iterativamente según (74), con una velocidad determinada por (75). Dicho enjambre inicia desde el centro del rango de distribución de las partículas denotado con x_0.

Aunque la versión original de PSO solo toma en cuenta la mejor posición del enjambre (P_{gbest}) para influir en la velocidad de cada individuo, en este caso se agrega P_{nbest} que representa la mejor posición del vecindario alrededor de un radio determinado.

Cabe anotar que, para calcular las mejores posiciones de los individuos se usa una función de aptitud dada por un campo de potencial artificial local dado por la

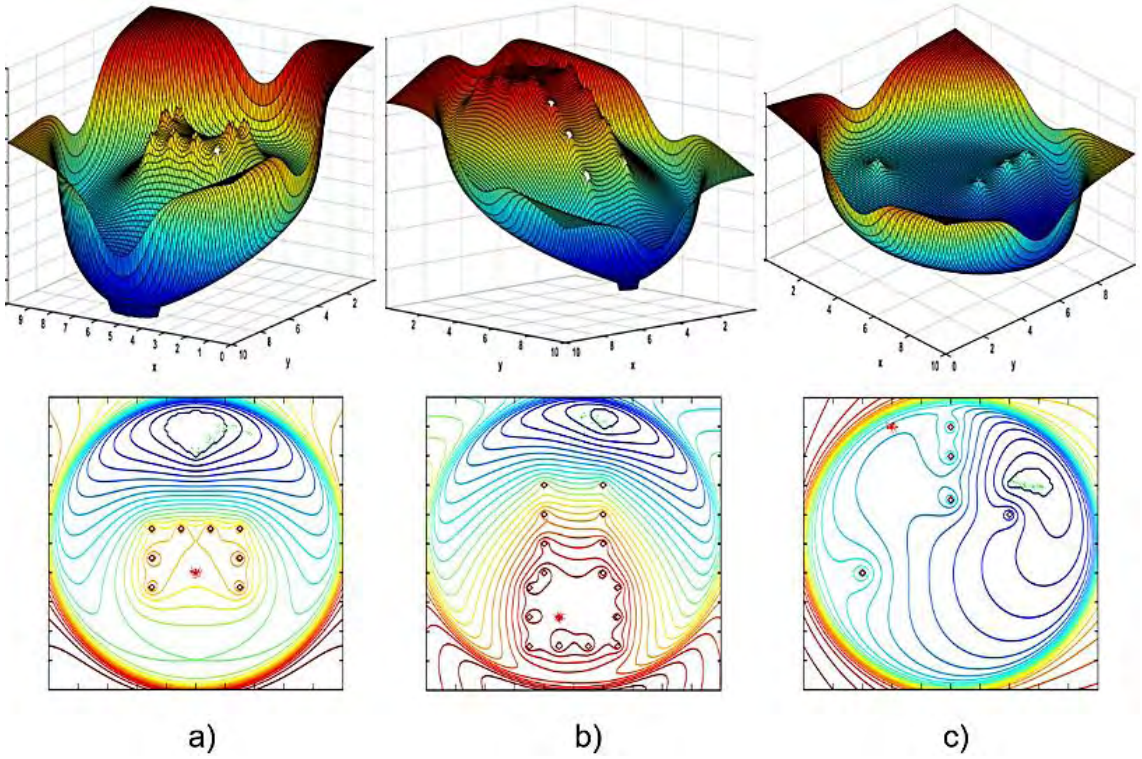
ecuación (68), igual a la suma de las funciones atrayente U_{att_2} de la ecuación (67) y funciones repulsivas armónicas de la ecuación (70).

En adición, el algoritmo se implementa con algunas propuestas propias de la planeación de trayectorias:

- Las partículas se inicializan en posiciones aleatorias dentro de una esfera con radio definido $\boldsymbol{r}$ y centro en la posición inicial.
- Se elimina el número fijo de partículas $\boldsymbol{nwf}$ con la peor aptitud en cada iteración para mejorar la búsqueda.
- El algoritmo se detiene cuando P_{gbest} se encuentra en una esfera de radio pequeño con centro en la posición de la meta.

La Figura 25 muestra la representación de los escenarios y sus respectivas curvas de nivel.

Figura 25. Representación de escenarios de prueba para método PSO



a) Escenario tipo trampa b) Escenario tipo pasaje estrecho c) Escenario tipo arreglo de obstáculos. **Fuente:** Esta investigación.

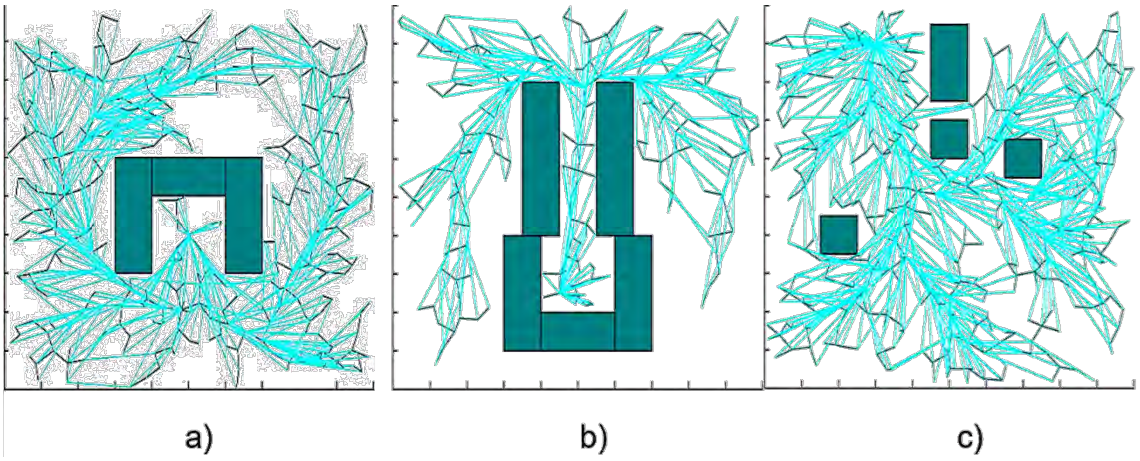
3.1.3. Exploración rápida de árbol aleatorios*

Como método representativo de la búsqueda aleatoria, generalmente sus aplicaciones tienen el enfoque de exploración de espacios. No obstante, independientemente de cómo se cree el árbol de búsqueda, siempre quedará establecido un mapa de ruta, que tiene la consideración de evitar colisiones con obstáculos estáticos y definir la mayor cantidad de rutas posible. Al final de las iteraciones del algoritmo es posible escoger la mejor combinación de estas rutas minimizando el criterio de desempeño de la menor distancia recorrida.

Con respecto a la representación de obstáculos, se opta por rectángulos expresados mediante el vector $Obstáculo_i = [a, b, c, d]$, donde a, b son el vértice inferior izquierdo y c, d el ancho y alto del rectángulo respectivamente. Para la creación del mapa de ruta se define una longitud para cada rama denotada con EPS y tal como en la propuesta teórica, un máximo número de nodos permitidos Nod_{max}.

La Figura 26 indica la representación de los escenarios y la creación del mapa de ruta en cada uno de ellos.

Figura 26. Representación de escenarios de prueba para el método RRT*.



a) Escenario tipo trampa b) Escenario tipo pasaje estrecho c) Escenario tipo arreglo de obstáculos. En verde oscuro se representa los obstáculos y en negro y cian las ramas creadas para definir el mapa de ruta. **Fuente:** Esta investigación.

3.1.4. Descomposición por celdas trapezoidales caso único robot

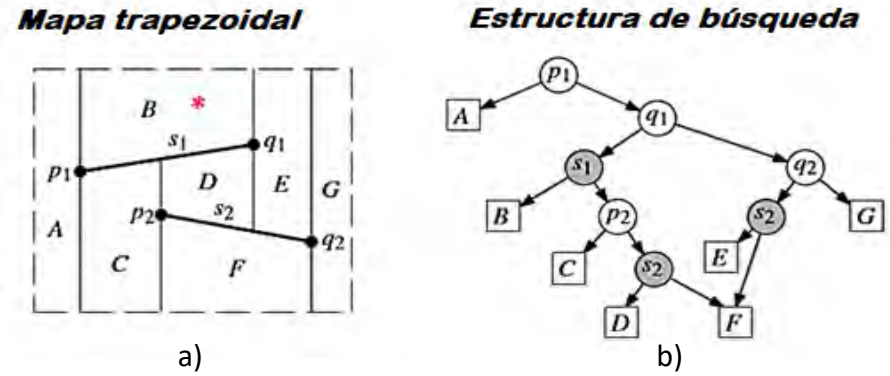
Para definir la programación de este algoritmo en planeación de trayectorias se empieza por definir *S* como un conjunto de *n* segmentos de línea $(s_1, s_2, \ldots, s_n)$, definidos a su vez por los vértices $p\ y\ q\ \in \mathbb{R}^2$ como en el ejemplo de la Sección 2.2.3 Figura 13. Donde *p* corresponde al vértice izquierdo y *q* al derecho.

A razón de las definiciones expuestas en la Sección 2.2.3, *S* y B (que es la caja delimitadora) componen el grafo que se quiere descomponer. Para lograrlo se plantea la ejecución de un algoritmo que ingrese aleatoriamente cada segmento de línea de *S,* generando progresivamente su mapa trapezoidal y una estructura de búsqueda de puntos en el mapa que además de facilitar la búsqueda ayude a crear el mapa trapezoidal. En otras palabras se quiere ingresar punto a punto los vértices *p* y *q* que componen cada s_i y extender las líneas verticales progresivamente, generando así los trapezoides.

Para simplificar el problema se tratará con grafos en posición general limitando así la orientación de los segmentos de línea, los cuales no pueden estar en posición vertical, es decir, que sus vértices *p* y *q* no estén alineados "verticalmente" como se observa en la Figura 29a. De la misma forma, para evitar que al "extender las líneas verticales" se creen virtualmente los segmentos de línea con la misma orientación, los vértices *p* y *q* que definen un segmento de línea no deben estar alineados con los que definen a otro como en el ejemplo de la Figura 29b.

Dicho esto, se empieza definiendo la estructura de búsqueda *D* como un grafo acíclico dirigido con una sola raíz y una única hoja por cada trapezoide del mapa trapezoidal de *S*, tal como se muestra en el ejemplo de la Figura 27 donde los nodos internos se definirán como *x-nodos* que corresponden a los puntos *p* y *q* que definen cada segmento en *S* y *y-nodos* que corresponden a cada segmento $s \in S$.

Figura 27. Estructura de búsqueda para el mapa trapezoidal, generado por los segmentos de línea s_1 y s_2



Fuente: Esta investigación a partir de [4].

Supóngase que en el ejemplo de la Figura 27 se quiere ubicar al trapezoide que contiene el punto marcado por * en el mapa trapezoidal. Para ello se comienza en la "raíz" de la estructura de búsqueda creada para este mapa que en cuyo caso es el *x-nodo* "p_1". Las reglas para descender hacia la izquierda o la derecha de dicho nodo, con dirección a las "hojas" que identifican cada trapezoide en la estructura, son las siguientes:

Si se trata de un *x-nodo* se debe responder a la pregunta ¿Se encuentra a la derecha o a la izquierda del vértice? refiriéndose a la ubicación de * respecto al *p* o *q* en cuestión. Para el caso de "p_1", se desciende a la hoja A si la respuesta es *a la izquierda*, o al *x-nodo* "q_1" si la respuesta es *a la derecha*. Si se trata de un *y-nodo* la pregunta será ¿Se encuentra arriba o debajo del segmento de línea? refiriéndose a la ubicación de * respecto al *s* en cuestión. Para el caso del *y-nodo* "s_1", se desciende a la hoja B, si la respuesta es *arriba* o al *x-nodo* "p_2" si la respuesta es *debajo*. Así, la ruta de nodos seguida para encontrar al punto * en la estructura de búsqueda de la Figura 27 será: p_1, q_1, s_1 y finalmente *B.*

La función de búsqueda propuesta para la estructura *D* en Matlab tiene la siguiente forma:

Función *Dsearch(D,v)*

Entrada: La estructura de datos *D*, el punto *v* de consulta
Salida: Índice del trapezoide que contiene a *v*

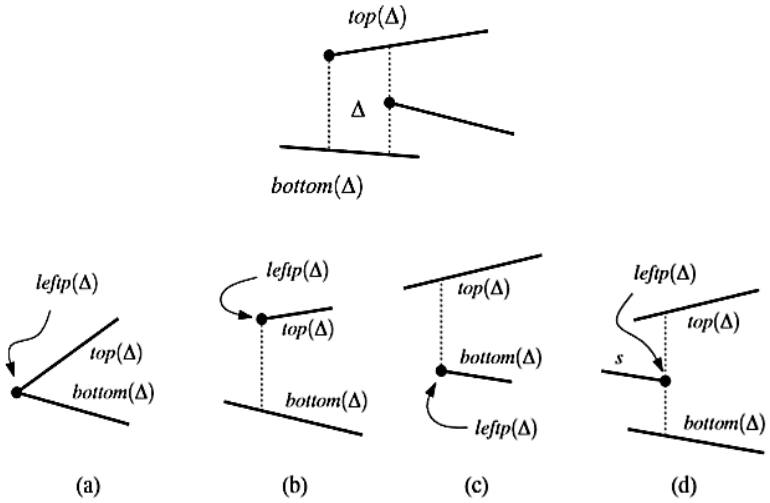
1. a ← 1, El índice de la raíz, para iniciar la búsqueda siempre será la primera posición
2. ***For*** *i* ← 1 *hasta n*
3. ***Switch*** *clase del nodo D(a)*
4. ***Case*** *x-nodo:*
5. ***If*** v_x*(la coordenada x del punto)* se encuentra a la derecha de *Valor del nodo D(a) (coordenada x)*
6. *a* ← índice del nodo descendente derecho
7. ***Else***
8. *a* ← índice del nodo descendente izquierdo
9. ***Case*** *y-nodo:*
10. ***If*** v_y*(la coordenada y del punto)* Está sobre *Valor del nodo D(a) (*la función de la recta que define el segmento de línea evaluada en v_x*)*
11. *a* ← índice del nodo descendente derecho
12. ***Else***
13. *a* ← índice del nodo descendente izquierdo
14. ***Case*** *hoja:*
15. ***Return*** Valor del nodo D(a) (índice del trapezoide en cuestión)
16. ***Break*** *to for*

Para continuar se procede a definir *T(S)* como el mapa trapezoidal del grafo definido por *S*, representado por una estructura que identifica cada trapezoide, los puntos que lo definen, sus relaciones de adyacencia, su índice y el puntero hacia la estructura de búsqueda *D.* Así como la estructura *D* contiene los índices de los trapezoides de *T(S).*

Un trapezoide Δ ∈ *T(S)* está definido por un punto en su extremo izquierdo $leftp(\Delta)$, un punto en su extremo derecho $rightp(\Delta)$ (cada uno correspondiente al punto *p* o *q* de algún segmento de S). Además, tiene un segmento s_i de recta sobre el

trapezoide $top(\Delta)$ y un segmento s_j de recta bajo el trapezoide $bottom(\Delta)$ (Figura 28) y se identifica por un "índice" (posición en el mapa trapezoidal) o un identificador que puede ser cualquier carácter.

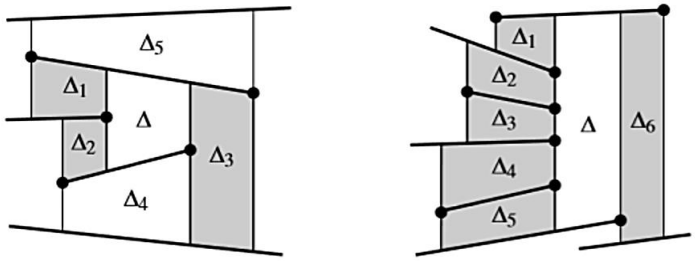
Figura 28. Definición del trapezoide y cuatro de los cinco casos posibles para el punto left(Δ) en el caso general.



El quinto caso es cuando el punto coincide con los límites en B. **Fuente:** [4].

En tanto a las relaciones de adyacencia, en el caso general cada trapezoide solo puede tener un máximo de 4 vecinos, 2 a la izquierda y 2 a la derecha. Cada uno de ellos puede compartir el segmento de línea superior $top(\Delta)$ o el segmento de línea inferior $bottom(\Delta)$. Cuando el vecino comparte el segmento de línea superior, se le nombra con el prefijo *'Upper'* seguido del lado al que se encuentra. Si comparte el segmento de línea inferior se añade el prefijo *'Lower'*. Por ejemplo, en la Figura 29a $\Delta_1$es el *UpperLeftNeighbor* de Δ. Nótese que Δ no tiene *LowerRightNeighbor,* sin embargo su *LowerLeftNeighbor* es Δ_2. Cada una de estas definiciones y características de adyacencia deben hacer parte de toda subestructura $\Delta \in T(S)$.

Figura 29. Grafos de posición general y no general.



a) b)

a) Relaciones de adyacencia entre trapezoides de un grafo en posición general. b) Caso en que no se trata de un grafo en posición general. **Fuente:** [4].

Con estas definiciones, se pretende generar *T(S)* añadiendo progresiva y aleatoriamente segmentos de línea de *S*, como en el ejemplo de la Figura 30 donde se pretende crear el mapa trapezoidal al agregar el nuevo segmento de línea s_i al mapa trapezoidal creado previamente. Nótese que al generar el nuevo mapa se genera también una nueva estructura de búsqueda *D* que obedece a las nuevas condiciones. No se parte de cero ya que, como se verá más adelante la nueva estructura de búsqueda tiene como base a la anterior. Por esto mismo se entiende que la estructura de búsqueda también se genera progresivamente. La aleatoriedad propuesta se justifica puesto que la ubicación aleatoria de los nodos mejora los tiempos de búsqueda de un punto en el grafo [4].

Al agregar un nuevo segmento de línea, no es necesario modificar todo el mapa o toda la estructura, basta con modificar los trapezoides afectados en el mapa y por ende las hojas afectadas en la estructura de búsqueda.

Para encontrar los trapezoides afectados al ingresar un nuevo segmento s_i, es necesario fijarse solo en las relaciones de adyacencia y la ubicación de los puntos que definen los trapezoides respecto al segmento de recta agregado. Si Δ_j es el primer trapezoide afectado por contener al punto *p*, el siguiente trapezoide Δ_{j+1} (si lo hay) será uno de los vecinos derechos de Δ_j. Para saber cuál, solo se pregunta por si el $rightp(\Delta_j)$ está por encima de s_i. Si es así, entonces Δ_{j+1} es el *LowerRightNeighbor* de Δ_j, de lo contrario es el *UpperRightNeighbor* como se muestra en la Figura 30. Tal función sigue los siguientes pasos:

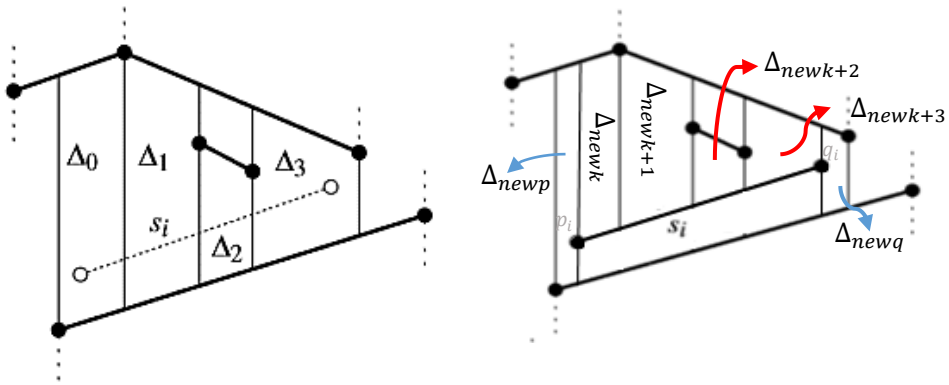
Función *FollowSegment(T,D,* s_i*)*

Entrada: El mapa trapezoidal T, la estructura de búsqueda D y el segmento de recta a agregar s_i
Salida: Índices de trapezoides intersectados por s_i

1. Se toman *p* y q de s_i
2. Se utiliza la *función* ***Dsearch(****D,p****)*** *para encontrar el índice de* Δ_1

3. $j \leftarrow 1$
4. ***While*** *q* se encuentre a la derecha de $rightp(\Delta_j)$
5. ***If*** $rightp(\Delta_j)$ se encuentra encima del segmento s_i
6. $\Delta_{j+1} \leftarrow$ *LowerRightNeighbor* de Δ_j
7. ***Else***
8. $\Delta_{j+1} \leftarrow$ *UpperRightNeighbor* de Δ_j
9. $j \leftarrow j$++
10. ***Return*** *índices de los trapezoides* $\Delta_1, \Delta_2, \Delta_3, \ldots, \Delta_j$

Figura 30. Ejemplo para la identificación de trapezoides afectados por un segmento de línea y la creación posterior del nuevo mapa trapezoidal.



Fuente: Esta investigación a partir de [4].

Después de encontrados los trapezoides afectados por s_i es necesario reemplazarlos por nuevos trapezoides. Para ello se propone lo siguiente.

Como se dijo anteriormente, para la implementación se toma a *T(s)* como una estructura que contiene a las subestructuras Δ, las que a su vez contienen información sobre los puntos de definición ($leftp$, *Top, etc.*) y relaciones de adyacencia (*LowerRightNeighbor, UpperLeftNeighbor, etc.*) ya definidas de cada trapezoide del mapa.

Ahora bien, es posible comprobar que al agregar un segmento de línea se agregan como mínimo dos trapezoides nuevos, arriba y abajo de s_i, puesto que siempre se afectará al menos un trapezoide. Por lo tanto como primer paso se crea un trapezoide representado por Δ_{newk} en *T(s),* con los puntos de definición y características de adyacencia del primer trapezoide afectado Δ_j, cambiando solo el $bottom$ de Δ_{newk} por s_i, $leftp(\Delta_{newk})$ por p_i *(punto izquierdo que define* s_i*)* y las relaciones de adyacencia, debido a que este trapezoide no tendrá *LowerLeftNeighbor*. Esto se puede observar en el ejemplo de la Figura 30.

Para definir los siguientes trapezoides que se crean sobre el segmento de línea (si se crean), se pregunta si $leftp(\Delta_{j+1})$ se encuentra por encima de s_i (recuerde que

Δ_{j+1} es el segundo trapezoide afectado). En caso afirmativo, se crea un nuevo trapezoide Δ_{newk+1} que asume las definiciones del trapezoide Δ_{j+1} cambiando $bottom(\Delta_{newk+1})$ por s_i y las relaciones de adyacencia con el trapezoide creado con anterioridad Δ_{newk} el cual pasa a ser su *LowerLeftNeighbor* (nótese que de manera recíproca Δ_{newk+1} pasa a ser el *LowerRightNeighbor* de Δ_{newk}). En caso negativo, es decir si el punto $leftp(\Delta_{j+1})$ se encuentra por debajo de s_i, no se crea otro trapezoide y en cambio Δ_{newk} toma como definición $rightp(\Delta_{j+1})$ y las relaciones de adyacencia *UpperRightNeighbor* del trapezoide Δ_{j+1} (nótese que el último trapezoide creado de esta manera no tiene *LowerRightNeighbor*). Este mismo proceso se lleva a cabo para los trapezoides que se crean debajo de s_i.

Por último, se pregunta si los puntos *p* y *q* que definen s_i ya han sido evaluados en iteraciones anteriores. Si *p* no ha sido evaluado, se crea un nuevo trapezoide Δ_{newp} con las definiciones y características de adyacencia de Δ_j (sus vecinos y puntos de definición) cambiando $rightp(\Delta_{newp})$ por *p* y las relaciones de adyacencia de los "*RightNeighbors"* que pasarán a ser los Δ_{newk}, tanto el que se crea para la parte superior de s_i como el que se crea para la inferior. Así mismo si *q* no ha sido evaluado, se crea un último trapezoide Δ_{newq} con las definiciones y características de adyacencia del último trapezoide afectado Δ_{jend} cambiando $leftp(\Delta_{newq})$ por *q*, las relaciones de adyacencia de los "*LeftNeighbors"* y los $rightp$ de los últimos trapezoides creados en el proceso anterior.

La función propuesta se ejecuta siguiendo los siguientes pasos:

Función *NewTrapezoids(T,J, s_i)*

Entrada: El mapa trapezoidal T, J arreglo con los índices de los trapezoides afectados de primero a último y el segmento de recta a agregar s_i
Salida: El nuevo mapa trapezoidal tras las anexiones

1. Se definen Δ_{new1}
2. $\Delta_{new1} \leftarrow \Delta_{J_1}$ Siendo Δ_{J_1} el primer trapezoide afectado
3. Se actualiza $bottom(\Delta_{new1}) \leftarrow s_i$ y $leftp(\Delta_{new0}) \leftarrow p$
4. ***For*** *i=2* hasta el tamaño de *J*
5. ***If*** $leftp(\Delta_{J_i})$ está sobre s_i
6. Se crea un nuevo $\Delta_{newk++} \leftarrow \Delta_{J_i}$
7. $bottom(\Delta_{newk++}) \leftarrow s_i$
8. <u>Se actualizan las relaciones de adyacencia</u>
9. ***Else***
10. $rightp(\Delta_{newk}) \leftarrow rightp(\Delta_{J_i})$
11. <u>Se actualizan las relaciones de adyacencia</u>
12. ***If*** *p* no se ha evaluado
13. $\Delta_{newk++} \leftarrow \Delta_{J_1}$
14. $rightp(\Delta_{newk}) \leftarrow p$

15. ***If*** *q* no se ha evaluado
16. $\Delta_{newk++} \leftarrow \Delta_{J_{end}}$
17. $leftp(\Delta_{newk}) \leftarrow q$
18. Se actualizan los rightp de los últimos trapezoides creados en el ***for***
19. Se reemplazan los trapezoides creados en el mapa trapezoidal

En el ciclo **for** de la función anterior, solo se ha descrito el proceso para los trapezoides creados en la parte superior de s_i pero puede añadirse en el mismo lugar el proceso para los creados en la parte inferior.

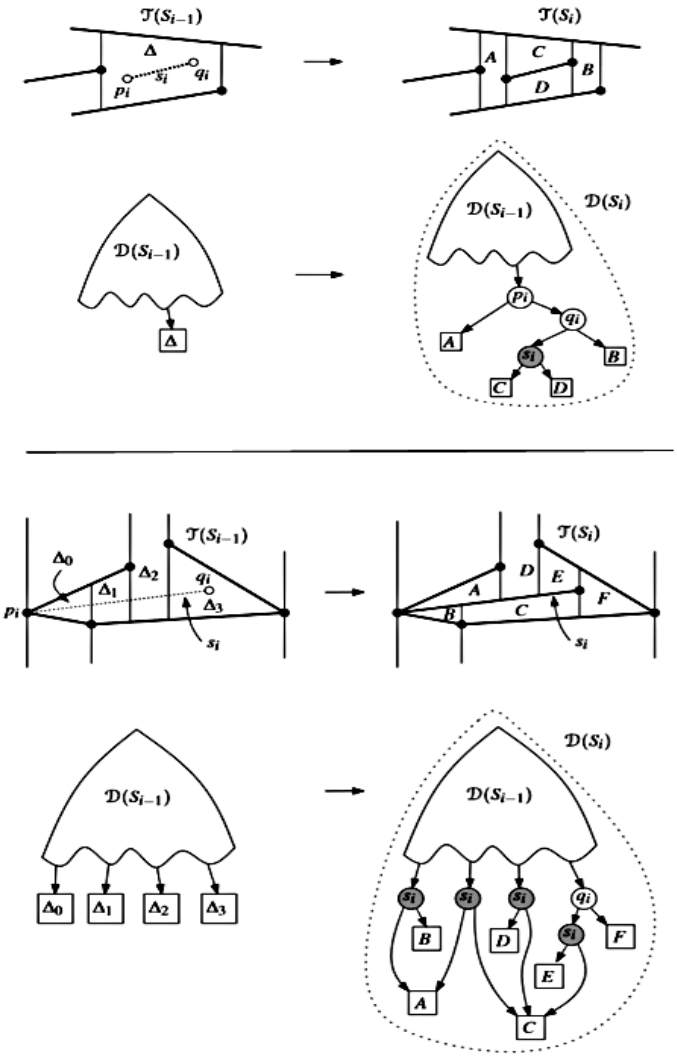
Para identificar los trapezoides se ha utilizado su puesto en la estructura *T*. Por lo tanto se debe tener cuidado a la hora de ubicar los nuevos trapezoides ya que eliminar un trapezoide implica que otros "escalen" a su lugar en la estructura y su identificador cambie. Aquí se opta por reemplazar los lugares de los trapezoides afectados y crear nuevos lugares al final de la estructura para los trapezoides creados que falte por asignar. Otro enfoque puede ser utilizar identificadores para cada trapezoide, como por ejemplo con cadenas de caracteres.

En tanto a las relaciones de adyacencia, se ha asignado un campo en la estructura *T* que almacena para cada trapezoide el identificador de su vecino. Por ejemplo, si el trapezoide número 3 tiene como vecino superior izquierdo al trapezoide número 7 se almacena en *T(3) UpperLeftNeighbor* el valor '7'. Nótese que al crear nuevos trapezoides se cambian también las relaciones de adyacencia. Así, por ejemplo cuando se crea Δ_{newk} éste toma los vecinos del anterior Δ_j y por tanto tales vecinos deben actualizar su relación de adyacencia con el nuevo Δ_{newk}.

En la función anterior se puede identificar que trapezoides Δ_{newk} reemplazan al trapezoide afectado Δ_j y es posible reemplazar también la hoja que representa a cada trapezoide en la estructura de búsqueda *D,* por una nueva subestructura que contenga los nodos y las hojas de los nuevos trapezoides creados teniendo en cuenta que la primera forma de la estructura es una árbol con solo una hoja.

Para ello, si solo un trapezoide es afectado, la hoja es reemplazada con una subestructura parecida a la que se muestra en la Figura 31 superior. Nótese que si alguno de los puntos *p* y/o *q* ya fue evaluado, la subestructura del ejemplo cambia. Por ejemplo, si *p* ya fue evaluado, la raíz de la subestructura sería el *x-nodo* '*q'* y sus nodos descendentes se mantendrían iguales. Por otro lado, si *q* ya fue evaluado, la raíz se mantiene en el *x-nodo* '*p'* pero su nodo descendente derecho sería el *y-nodo* $'s_i'$. Cuando se afectan varios trapezoides, la estructura para las ramas afectadas cambia de una manera similar al ejemplo de la Figura 31 inferior. Nótese que las hojas de los trapezoides "intermedios" afectados son simplemente reemplazadas por una subestructura que tiene como raíz el *y-nodo* $'s_i'$, cuyos nodos descendentes son las hojas de los nuevos trapezoides. Para los trapezoides creados a los extremos, la actualización de su subestructura dependerá de si los puntos *p* o *q* ya hayan sido evaluados.

Figura 31. Ejemplos de la modificación de la estructura de búsqueda.



Superior: Solo un trapezoide es modificado; Inferior: Varios trapezoides son modificados.
Fuente: Esta investigación a partir de [4].

Con lo anterior, el algoritmo que realiza todo el proceso sigue los siguientes pasos:

Algoritmo *TrapezoidalMap(S)*

Entrada: El conjunto S de segmentas de línea que no se cruzan.
Salida: El mapa trapezoidal *T(S)* y la estructura de búsqueda *D,* en una caja delimitadora B

1. Se inicializan, la caja delimitadora *B* y las estructuras *T* y *D*. El estado inicial *T(0)* será un solo trapezoide definido por *B* y la estructura de búsqueda solo tendrá un nodo en dirección a dicho trapezoide.
2. Se crea una permutación aleatoria para ingresar los elementos de S.

3. ***For*** *i*=1 hasta *n*
4. ***FollowSegment(****T,D,* s_i***),*** Encuentra los índices *J* de los trapezoides atravesados por s_i
5. ***NewTrapezoids(****T,J,* s_i***)***, Crea nuevos trapezoides y los reemplaza por los trapezoides afectados en *T.*
6. Se reemplaza las hojas de los trapezoides afectados en D, por nuevas subestructuras.

Hasta aquí se ha realizado el mapa trapezoidal de un grafo definido por *S* y *B.* Lo que sigue a esto es relativamente más sencillo. Teniendo en cuenta que el objetivo es lograr un mapa de ruta con ayuda de las relaciones de adyacencia de los trapezoides ubicados en el espacio libre, lo primero que se debe hacer es identificar que trapezoides están dentro de los objetos. Esto se puede lograr con un "algoritmo de punto en un polígono" descrito en varios estudios de geografía computacional. Por ejemplo, en el capítulo 4 de [61] se busca identificar si un punto está dentro o fuera de un polígono extendiendo un segmento de recta desde el punto en cuestión, hasta los límites de una caja delimitadora y en cualquier dirección. Si dicho segmento intersecta las aristas del polígono un número par de veces o ninguna, el punto está afuera, y en otro caso está adentro. Se puede también crear un algoritmo basado en las relaciones de adyacencia de los trapezoides, como se sugiere en [4]. Aquí se ha usado la función *inpolygon* de Matlab, usando un punto dentro del trapezoide como punto de consulta.

Para obtener el mapa de ruta, se comienza preguntando por los vecinos de cada trapezoide en el espacio libre (los que resultan de quitar del mapa) identificados en el proceso anterior. Si tales vecinos existen, se crea un nodo en el centro de la extensión vertical que limita al trapezoide de su vecino, ya sea izquierdo o derecho, superior o inferior, y se procede a "conectar" los nodos creados al lado izquierdo con los creados al lado derecho. Nótese que si el "nodo" ya ha sido añadido no se crea uno nuevo pero se debe remitir al creado para definir las conexiones. Si se da el caso en el que el trapezoide no tenga ningún vecino en uno de sus lados, se crea un nodo en el centro dentro del trapezoide que se conectará con los vecinos del lado restante, como en el ejemplo de la Figura 13 de la Sección 2.2.3.

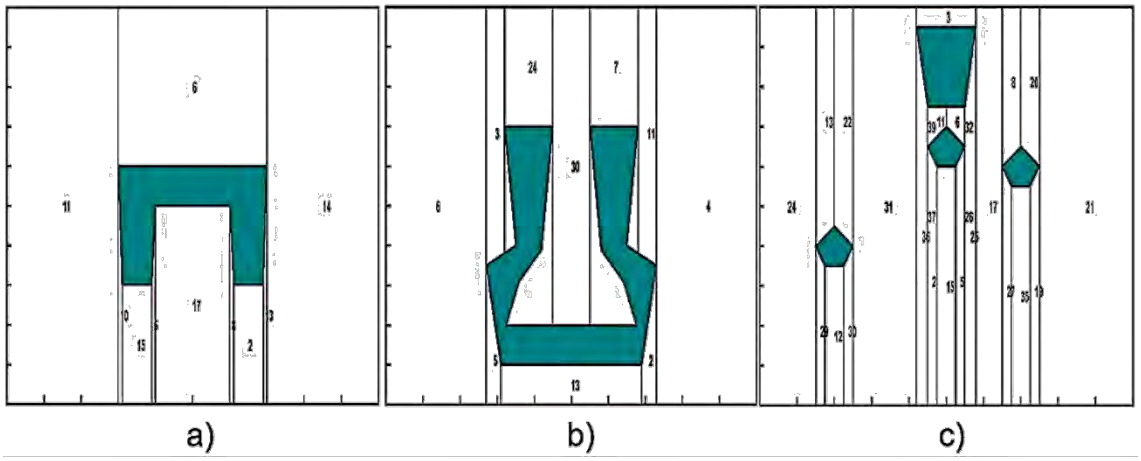
Por último, con el mapa de ruta creado a manera de una matriz de conexiones, lo único que queda es encontrar el camino más corto entre los nodos, teniendo en cuenta que debe añadirse un par de nodos nuevos tanto para el punto de inicio del robot, como para el de llegada. Además, estos nodos deben estar conectados, ya sea al centro del trapezoide que los contiene, o bien al nodo más cercano. Para encontrar tal camino se puede usar cualquier algoritmo de búsqueda de caminos en grafos, recomendándose el algoritmo A* o de camino mínimo por su simplicidad. Aquí se ha usado el algoritmo de *Dijkstra* descrito en [27], por familiaridad. (Ver anexo 3).

En este orden de ideas, se desarrolla la programación, modelando los obstáculos como polígonos con forma irregular. Dicha representación se hace con una

estructura O que viene organizada como $O = [a, b]$; a es la cantidad de polígonos que representan los obstáculos en el entono de navegación y b el vector de posiciónes x,y de cada uno de los vértices de los polígonos.

La Figura 33 muestra la representación de los escenarios y su respectiva descomposición por celdas trapezoidales.

Figura 32. Representación de escenarios de prueba para el método CD



a) Escenario tipo trampa b) Escenario tipo pasaje estrecho c) Escenario tipo arreglo de obstáculos. En verde oscuro se representa los obstáculos y en líneas verticales negras los bordes de cada celda que divide el espacio de configuraciones libre. **Fuente:** Esta investigación.

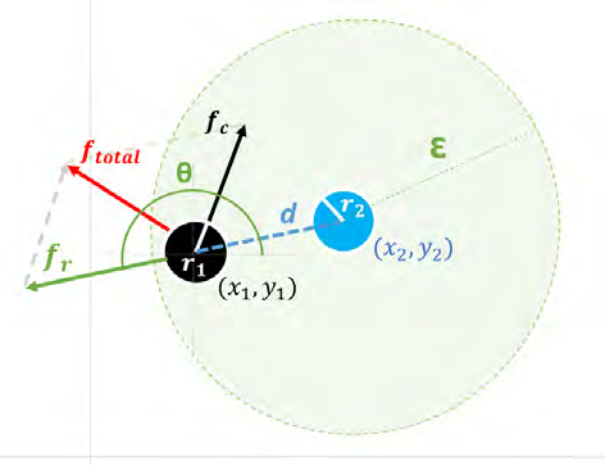
3.1.5. Función de navegación para campos potenciales artificiales. Extensión caso múltiples robots

A partir de lo expuesto en la Sección 2.3.2, nótese que no es necesario incluir siempre el campo generado por los robots al campo de potencial total, puesto que dicho campo cambia en el tiempo y no sería posible generar una trayectoria óptima y libre de mínimos locales. Esta trayectoria se obtendría variando los parámetros de definición de los campos de manera instantánea y diferenciada para cada robot en *W*. Por esto, es conveniente utilizar un radio en donde los campos generados por los robots tengan influencia, de la siguiente manera:

Sea r_1 un robot que viaja dentro de un campo de potencial que ejerce sobre él una fuerza representada por el vector $\boldsymbol{f_c}$, r_2 un robot que viaja a través del mismo campo, *d* la distancia entre r_1 y r_2 , $\varepsilon \in \mathbb{R}^+$una distancia de separación mínima entre los robots y $\boldsymbol{f_r}$ un vector representante de la fuerza que el campo generado por r_2 ejerce sobre r_1 siempre que $d < \varepsilon$.

Por ejemplo, en la Figura 33 se puede observar que cuando r_1 entra al campo generado por r_2, solo entonces experimenta su fuerza de repulsión y por ello, su trayectoria estará definida por la suma vectorial de las fuerzas $\boldsymbol{f_c}$ y $\boldsymbol{f_r}$ donde ε>r_2.

Figura 33. Fuerza de repulsión que r_1 experimenta al entrar al campo generado por r_2.



Fuente: Esta investigación.

Para funciones de potencial complejas como la función de navegación, no es necesario que el campo generado por los robots sea evaluado de igual manera al que se evalúan los objetos estáticos para evitar mínimos locales, por el mismo hecho de haber limitado su influencia. Por esta razón, aquí se ha utilizado una función simple para representar el campo generado por los robots así

$$\boldsymbol{f_r} = \frac{1}{d^2}\boldsymbol{d} \tag{84}$$

Donde $\boldsymbol{d}$ es un vector unitario dirigido del robot que genera el campo hacia el robot que lo percibe. Si (x_1, y_1) y (x_2, y_2) son las posiciones de los robots r_1 y r_2, respectivamente y θ el ángulo de $\boldsymbol{f}$ respecto a un plano de coordenadas (x, y), entonces

$$\boldsymbol{f_r} = \frac{1}{((x_1-x_2)^2+(y_1-y_2)^2)} \times (\cos(\theta)\,\hat{\imath} + \sin(\theta)\hat{\jmath}) \tag{85}$$

Dónde

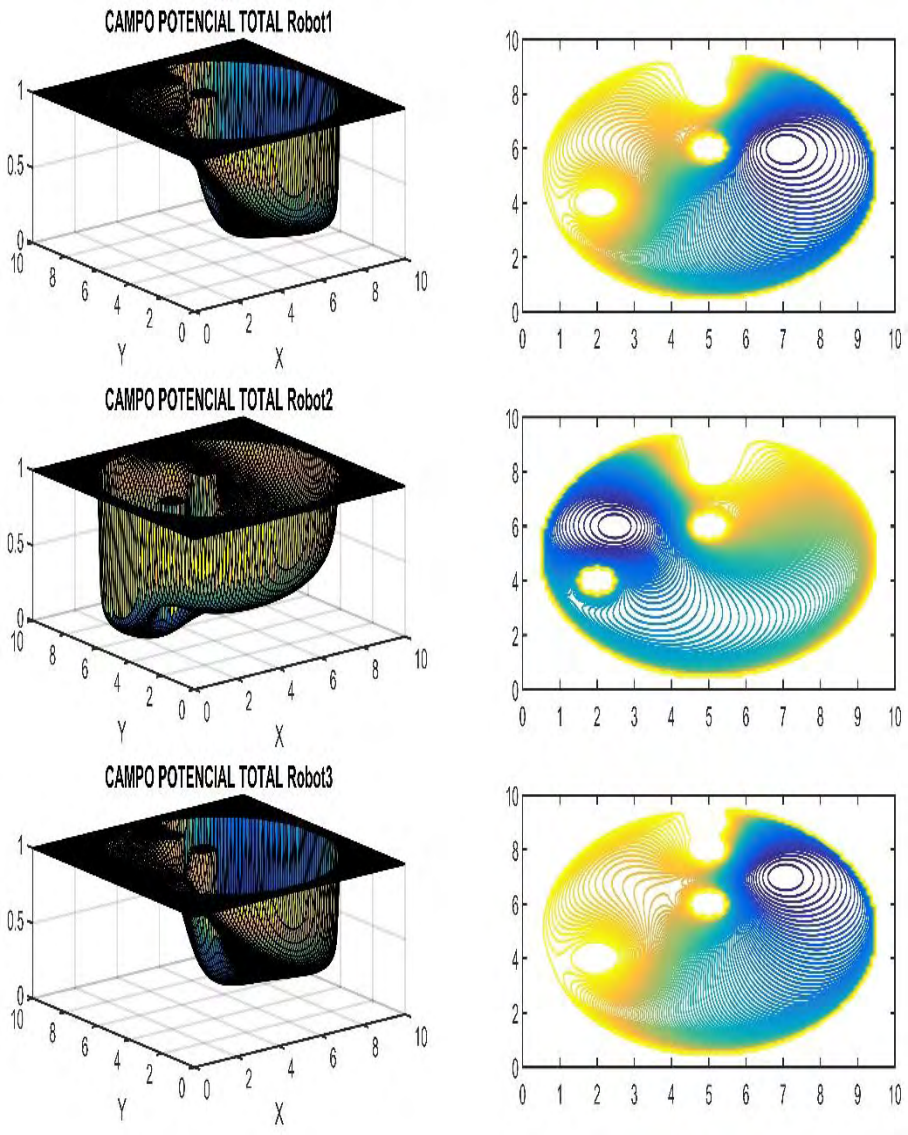
$$\tan(\theta) = \frac{y_1-y_2}{x_1-x_2} \tag{86}$$

De donde es posible despejar θ a partir de métodos numéricos que tienen en cuenta la evaluación de los cuatro cuadrantes como $atan2$, y es calculado de forma

computacional en vista de que las funciones inversas tales como $atan$ presentan dificultades al momento de ser evaluadas.Por último, nótese que si las trayectorias de todos los robots son generadas por este método, pueden formarse mínimos locales. Por ello, tal como se propone para el algoritmo de descomposición por celdas (Sección 3.1.4), es conveniente darle prioridad diferenciada a cada uno de los robots de manera jerarquizada.

La Figura 34 y la Tabla 1 exponen los parámetros sintonizados que determinan los resultados en el caso de múltiples robots y que fueron seleccionados luego de validaciones en simulación.

Figura 34. Representación del escenario de navegación para robot 1,2 y 3



Fuente: Esta investigación.

Tabla 1 Valor numérico de los parámetros. Extensión de PF para múltiples robots.

ESCENARIO DE NAVEGACIÓN		
Posición de los obstáculos: $q_1 = [2,4]$ $q_2 = [5,6]$ $q_3 = [5,8]$ $q_4 = [5,9]$		
Robot 1	**Robot 2**	**Robot 3**
Posición de inicio y llegada: $q_{start} = [2.5\ , 6]$ $q_{tar} = [7,6]\ ;\ K = 3.6$	**Posición de inicio y llegada:** $q_{start} = [4.5,2.5]$ $q_{tar} = [6,9]\ ;\ K = 3.6$	**Posición de inicio y llegada:** $q_{start} = [3,9]$ $q_{tar} = [7,7]\ ;\ K = 3.6$

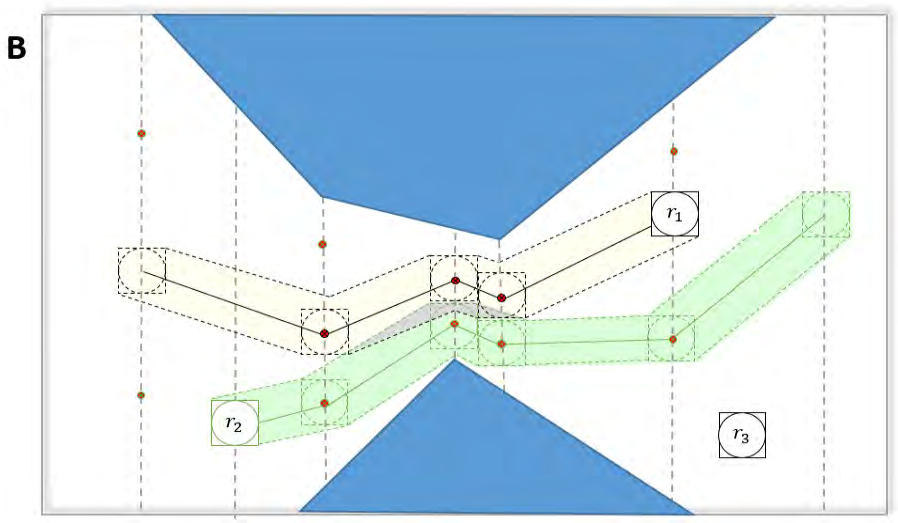
3.1.6. Descomposición por celdas trapezoidales. Extensión caso múltiples robots

Para el caso de múltiples robots, y como aporte a la planeación de trayectorias desde el punto de vista de la descomposición por celdas trapezoidales, se ha desarrollado la siguiente metodología para navegación de múltiples vehículos homogéneos en un entorno estático.

1. La renovación del mapa trapezoidal al agregar un Objeto Camino es conveniente, pero no completamente necesaria, puesto que en muchos casos, solo la técnica de zonas de intersección es capaz de evitar las colisiones después de aplicar el algoritmo de descomposición por celdas para un único robot.
2. Es conveniente actualizar el mapa trapezoidal con el Objeto Camino, lo que permite que las zonas de intersección no sean muy grandes y se limiten a zonas relativamente pequeñas debidas al cruce de trayectorias. Para simplificar el problema, si se trata de robots con formas regulares puede ignorarse la forma en este proceso y actualizar el mapa trapezoidal únicamente con los vértices de la trayectoria, lo que equivale a actualizarlo con el Objeto Camino generado por un robot puntual. Véase el ejemplo de la Figura 35.

Si se actualiza el mapa trapezoidal con los vértices de la trayectoria, las extensiones verticales serán las mismas que se crearon por los objetos estáticos, a excepción de las creadas en los extremos de la trayectoria. Entonces, no es necesario actualizar el mapa trapezoidal, sino crear nuevos nodos para el mapa de ruta sobre la extensión, arriba y abajo del nodo existente y teniendo en cuenta la forma de los robots *solo* en las extensiones verticales. De esta manera, tales nodos se crean, siempre y cuando la distancia entre el nodo existente y el extremo de la extensión vertical (ya sea el superior o inferior), sea mayor que la mayor distancia entre el centroide y la periferia de la suma de Minkowsky de la forma de los robots en cuestión. Esto último se hace para evitar la colisión con los obstáculos estáticos.

Figura 35. Nodos creados por el Objeto Camino generado por r_1 cuando no se toma en cuenta su forma, una trayectoria generada para el robot r_2 en el nuevo mapa de ruta y la formación de un nuevo Objeto Camino debido a esta.



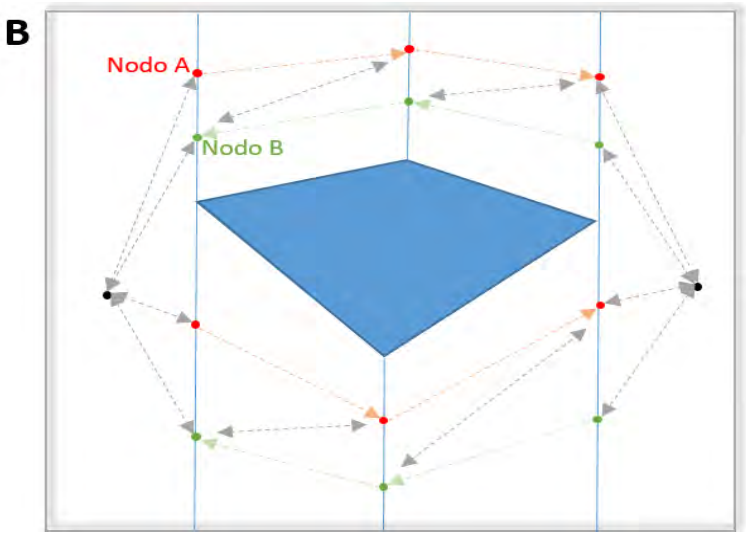
Fuente: Esta investigación.

Entre más robots se agreguen, es posible que se añadan más nodos a las extensiones verticales de manera excesiva e innecesaria, puesto que cada trayectoria definida genera un máximo de dos nodos en cada extensión. Por esto, se ha simplificado aún más el problema tomando solo 2 nodos en cada una de las extensiones verticales debidas a los objetos estáticos y las debidas a los extremos de las trayectorias. El creado en la parte superior será el Nodo A y el creado en la parte inferior será el Nodo B, como se puede observar por ejemplo en la Figura 36. Estos nodos cumplen las siguientes condiciones de conectividad:

1. Un Nodo A tiene conexión con otro Nodo A, solo si este último se encuentra a su derecha. (Color rojo)
2. Un Nodo A tiene conexión con un Nodo B, solo si este último se encuentra a su izquierda. (Color gris)
3. Un Nodo B tiene conexión con otro Nodo B, solo si este último se encuentra a su izquierda. (Color verde)
4. Un Nodo B tiene conexión con un Nodo A, solo si este último se encuentra a su derecha. (Color gris)
5. Un nodo que se crea en el centro de un trapezoide o en una extensión que no tenga espacio para dos nodos, tiene conexión con cualquier nodo cercano y viceversa. (Color negro)

Nótese que las condiciones anteriores implican tanto conexiones bilaterales, como unilaterales. A esto se refiere el sentido de las flechas de la Figura 36.

Figura 36. Mapa de ruta generado para la alternativa de añadir únicamente 2 nodos en las extensiones verticales.

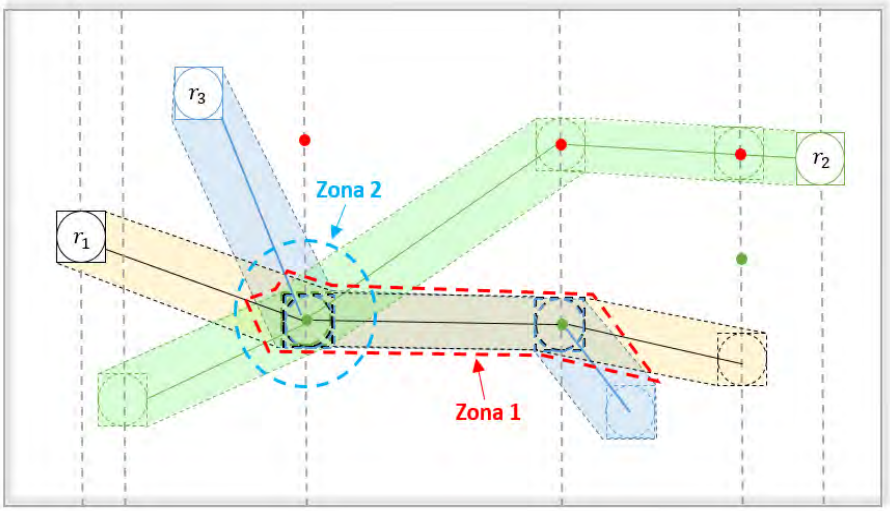


El sentido de las flechas indica la dirección de la conexión y si estás son unilaterales o bilaterales. **Fuente:** Esta investigación.

Con esta simplificación se plantea que no es necesario crear muchos nodos en las extensiones verticales, ya que es suficiente con añadir dos cuyo sentido en las conexiones con los nodos creados y en las otras extensiones o centros, eviten que los robots se encuentren frente a frente. Esto es similar al funcionamiento en una avenida de los carriles en doble sentido y los puntos de retorno. Esto *no* implica olvidarse de los Objetos Caminos, pues las zonas de intersección son indispensables para la técnica de evasión.

En el ejemplo de la Figura 37 se puede observar las intersecciones de los Objeto Camino. La zona remarcada como Zona 1, es una zona donde intersectan mayormente los Objetos Camino de los robots r_1 y r_3, quienes llevan la misma dirección. Con las reglas de transito descritas para las zonas de intersección, el robot r_3 debería esperar a que r_1 salga totalmente de la Zona 1, para poder continuar con su trayecto, lo que resulta innecesario. Para solucionar el inconveniente basta con añadir una exceptuación a la regla 2 para los robots que viajan en la misma dirección y una condición de distancia, que impida que el robot que viaja tras de otro en la misma dirección, se aproxime mucho al primero. En cambio, en la zona 2 se muestra la eficacia de las reglas mencionadas, puesto que se evita una colisión frontal.

Figura 37. Ejemplo de formación de Objetos Camino y zonas de intersección con el método simplificado.



Fuente: Esta investigación.

Siguiendo el espíritu de la técnica de las zonas de intersección y los campos de potencial, en este trabajo se ha hecho un aporte para implementaciones en las que las zonas de intersección donde pueden existir colisiones frontales no son muy extensas y con la ventaja de no requerir un conocimiento global de las trayectorias generadas por todos los robots, aparte de la simplicidad de su aplicación.

La metodología podría resumirse teniendo en cuenta los siguientes conceptos:

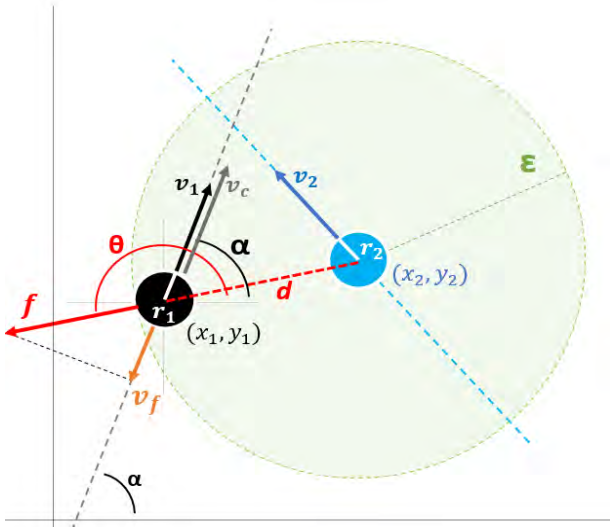
Sea r_1 un robot que viaja por una trayectoria definida a una velocidad lineal v_1, r_2 un robot que viaja por una trayectoria definida a una velocidad lineal v_2, *d* la distancia que los separa, $\varepsilon \in \mathbb{R}^+$una distancia de separación mínima entre los robots y $\boldsymbol{f}$ un vector dirigido de r_1 hacia r_2, cuya magnitud depende de *d* de acuerdo con

$$|\boldsymbol{f}| = \frac{a}{d^2} \tag{87}$$

Donde $a \in \mathbb{R}^+$ es una constante.

Lo que se busca es que la velocidad lineal (sobre la trayectoria) de los robots, dependa de la distancia que los separa. De esta forma, dependiendo del ángulo en el que se aproxime un robot a otro, éste último "frenará o retrocederá en su trayectoria", lo que implica una aceleración contraria a su actual movimiento, o "acelerará" en dirección de su movimiento.

Figura 38. Robot r_1 entrando al campo generado por r_2 y su interacción con el mismo.



Fuente: Esta investigación.

En el ejemplo de la Figura 38 se puede observar cómo un robot r_1 que viaja sobre su trayectoria, entra a un campo de repulsión de radio ε>r_2 generado por r_2 donde $\theta > \alpha$. Al entrar al campo, r_1 experimenta una fuerza de repulsión $\boldsymbol{f}$, cuya *proyección* (refiriéndose a un valor escalar) sobre la trayectoria de r_1 se simboliza como v_f. Se define v_c a la velocidad líneal con la que un robot se desplazaría si no se encuentra con otro robot, que se asume constante y siempre mayor que cero. Entonces la velocidad v_1 será:

$$v_1 = v_c + v_f \tag{88}$$

Si (x_1, y_1) y (x_2, y_2) son las posiciones de los robots r_1 y r_2, respectivamente y θ el ángulo de $\boldsymbol{f}$, respecto a un plano de coordenadas (x, y), entonces:

$$v_1 = v_c + \frac{a}{d^2}\cos(\theta - \alpha); \qquad \theta > \alpha \tag{89}$$

Dónde:

$$\tan(\theta) = \frac{y_1 - y_2}{x_1 - x_2} \tag{90}$$

Bajo la consideración de (86).

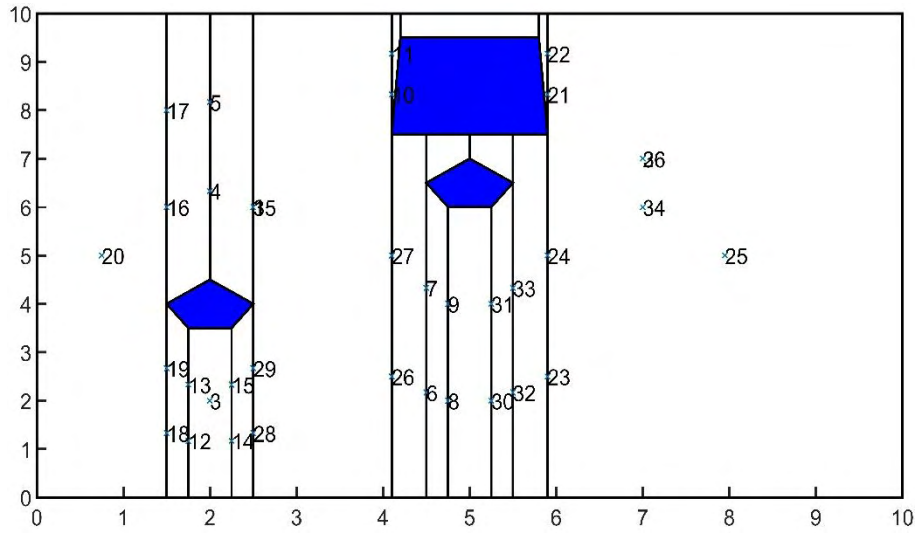
Como se observa, el signo de v_f depende del signo del $\cos(\theta - \alpha)$, de donde determina si el robot, acelera o desacelera en la trayectoria.

Esta técnica garantiza que los objetos no se choquen y es eficiente en varias aplicaciones. Sin embargo, el desempeño no es adecuado en pasajes estrechos puesto que, si dos robots se encuentran en un pasaje, uno de los dos va a tener que retroceder en su trayectoria para que el otro pase. Ahora bien, si la velocidad

del robot r_2 también es calculada con éste método, ambos robots se quedarían inmóviles en la intersección. Por esto, es necesario darle prioridad a uno de los dos, lo que puede referirse a un grado de sensibilidad del campo, como por ejemplo, aumentando o disminuyendo ε, o alterando el valor de la constante a, diferenciadamente para cada robot. Para aplicaciones con varios robots, es conveniente elaborar una jerarquía de asignación, como que el primer robot tenga una sensibilidad muy baja respecto al campo de los otros robots, o simplemente nula; que el segundo robot solo perciba con claridad el campo del primero; que el tercero solo perciba con claridad el campo del primer y segundo robot, y así sucesivamente. Esto es suficiente para que todos los robots lleguen a su meta.

Así, la Figura 39 y la Tabla 2 exponen los parámetros sintonizados que determinan los resultados en el caso de múltiples robots en CD (descomposición en celdas trapezoidales) y que fueron seleccionados luego de validaciones en simulación.

Figura 39. Representación de nodos para el escenario arreglo de obstáculos modificado para extensión de CD.



Fuente: Esta investigación.

Tabla 2. Valor numérico de parámetros. Extensión de CD para múltiples robots.

ESCENARIO DE NAVEGACIÓN		
$\boldsymbol{O} = [3,\ [2\ 2.5\ 2.25\ 1.75\ 1.5\ 2; 4.5\ 4\ 3.5\ 3.5\ 4\ 4.5]$ $[5\ 5.5\ 5.25\ 4.75\ 4.5\ 5; 7\ 6.5\ 6\ 6\ 6.5\ 7]\ ]$ $[4.1\ 4.2\ 5.8\ 5.9\ 4.1; 7.5\ 9.5\ 9.5\ 7.5\ 7.5]$		
Robot 1	**Robot 2**	**Robot 3**
Posición de inicio y llegada: $\boldsymbol{q_{start}} = [\boldsymbol{2.5}, \boldsymbol{6}]$ $\boldsymbol{q_{tar}} = [\boldsymbol{7}, \boldsymbol{6}]$	**Posición de inicio y llegada:** $q_{start} = [4.5,2.5]$ $q_{tar} = [6,9]$	**Posición de inicio y llegada:** $q_{start} = [3,9]$ $q_{tar} = [7,7]$

3.1.7. Obstáculos de velocidad recíproca (RVO)

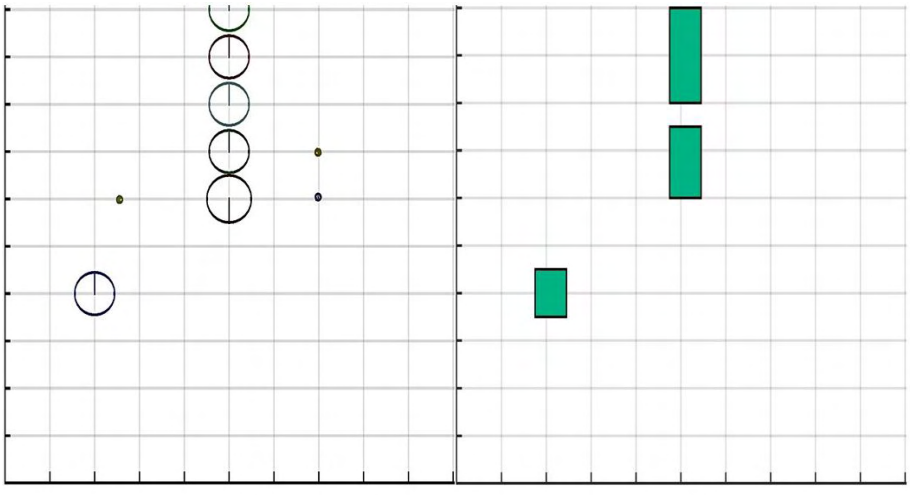
Como se expresó anteriormente, la programación de RVO se hace como lo presenta el autor en [1] y [2], trabajando únicamente en el escenario de arreglo de obstáculos modificado para múltiples robots. Lo anterior se propone porque la importancia en este caso está en la evitación de colisión durante la navegación de varios agentes, independientemente del entorno.

La modificación particular a esta técnica es la representación de los obstáculos estáticos. Aquí se realiza colocando en las posiciones de interés q_{obs_i}, robots con un radio rad_{obs} mayor al radio de los robots rad_{robot} que navegan por el espacio y con velocidad rotacional máxima de sus ruedas w_{rotvel} igual a cero.

La Figura 40 indica la representación del escenario de arreglo de obstáculos modificado y su equivalencia al estilo de representación usado en RRT*, es decir, utilizando rectángulos. En la Tabla 3 se encuentran los parámetros sintonizados que determinan los resultados para éste método y que fueron seleccionados luego de validaciones en simulación.

Cabe mencionar además, que como técnica de optimización para $V_A{}^*$ se aplica PSO, como lo propone el autor en [1].

Figura 40. Representación del escenario de navegación arreglo de obstáculos



a) Los obstáculos se modelan como robots de mayor tamaño y velocidad cero. b) Escenario análogo al representado con el método RRT*. **Fuente:** Esta investigación.

Tabla 3. Valor numérico de parámetros. RVO

ESCENARIO ARREGLO DE OBSTÁCULOS MODIFICADO		
Robot 1	**Robot 2**	**Robot 3**
$\boldsymbol{q_{start}} = [\mathbf{2.5, 6}]$ $\boldsymbol{q_{tar}} = [\mathbf{7, 6}]$	$q_{start} = [7,7]$ $q_{tar} = [2.5,6]$	$q_{start} = [2,2]$ $q_{tar} = [7,7]$
Posición de los obstáculos: $\boldsymbol{q_{obs_1}}$=$[\mathbf{2, 4}]$; $\boldsymbol{q_{obs_2}}$=$[\mathbf{5, 6}]$; $\boldsymbol{q_{obs3}}$=$[\mathbf{5, 8}]$; $\boldsymbol{q_{obs_4}}$=$[\mathbf{5, 9}]$ **Parámetros de los robots:** $\boldsymbol{rad_{obs}}$=0.45 $\boldsymbol{rad_{robot}}$=0.064 $\boldsymbol{w_{rotvel}} = \mathbf{20}$		

3.2. MARCO EXPERIMENTAL

3.2.1. DISEÑO DE EXPERIMENTOS EN PLATAFORMA VIRTUAL

Cada uno de los métodos fue implementado en Matlab®, en un ordenador con procesador Intel Core i7 de 3.0 GHz y 12 GB de memoria RAM.

3.2.2. DISEÑO DE EXPERIMENTOS EN PLATAFORMA FÍSICA REAL

Esta investigación lleva a la práctica el método de función de navegación para campos de potencial artificial en el caso de un único robot utilizando la plataforma robótica “Arduino-Robot” y módulos de comunicación Xbee Pro S3B de referencia XBP9B-DMST-022 revD. Además, se aplica una retroalimentación de la ubicación de las plataformas en el entorno de navegación mediante visión por computador usando una cámara de celular enlazada por dirección IP a una resolución de 720x720 pixeles.

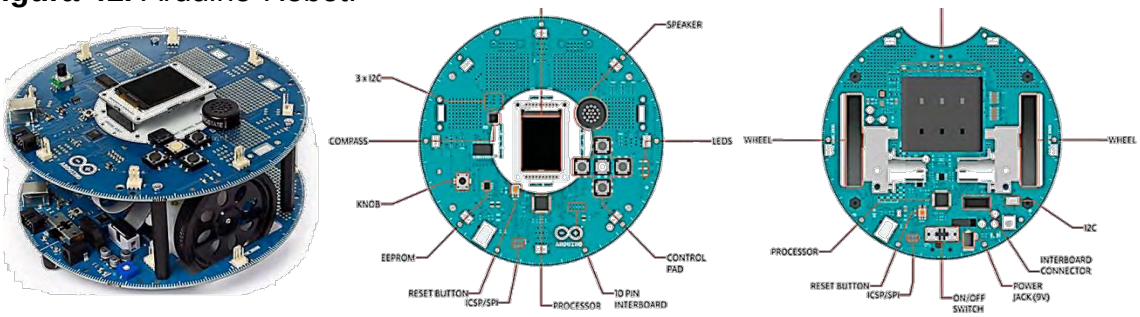
La programación se realiza con el software Arduino IDE versión 1.8.5 en leguaje C y el escenario de navegación se limita a un área cuadrada de 3 m^2 al estilo del propuesto para simulaciones de múltiples robots (arreglo de obstáculos modificado), con la finalidad de facilitar la practicidad de la implementación.

3.2.3. Plataforma “Arduino-Robot”

Es una plataforma de locomoción diferencial, que entre sus características más relevantes destacan una brújula digital integrada, dos microprocesadores ATmega32u4 y un lector de tarjetas SD. Sumado a esto, permite la conexión de

dispositivos externos como sensores y otras tarjetas de procesamiento por puertos digitales y analógicos, y puertos de comunicación I2C e ICSP. Las características detalladas de este dispositivo se encuentran en [3]. La Figura 41 muestra el dispositivo de perfil y sus dos tarjetas que permiten dividir las tareas a cada uno de los microprocesadores, encargándose uno del control global y ejecución del algoritmo principal y el otro del control local de actuadores.

Figura 41. Arduino-Robot.



a) Vista de perfil del robot de tracción diferencial. b) Placa superior del robot ("*Control Board"*). c) Placa inferior del robot ("*Motor Board"*). **Fuente:** Esta investigación a partir de [3].

El modelo matemático descrito en la Sección 2.1.8 se aplica a esta plataforma con radio de las ruedas igual a 0.03 m y distancia entre las ruedas igual a 0.128 m.

Por otra parte, cabe mencionar que como desventaja relativa a este dispositivo, se asocia la falta de información sobre los principales actuadores (Motores). Para la parte de control de movimientos, es indispensable contar con un modelo matemático que permita recibir como señal de entrada la velocidad angular de referencia y entregue a su salida la correspondiente señal de pwm, tal que se cumpla el criterio de control.

Para dar solución a este inconveniente, se tomaron medidas de las velocidades angulares respecto a algunos valores de pwm a plena carga, obteniendo los resultados de la Tabla 4. Con base en estos resultados, se calculó un modelo aproximado de los motores mediante la función *polyfit* de Matlab, resultante en un polinomio de cuarto orden cuyo comportamiento se aprecia en la Figura 42.

x: Velocidad angular en rad/s

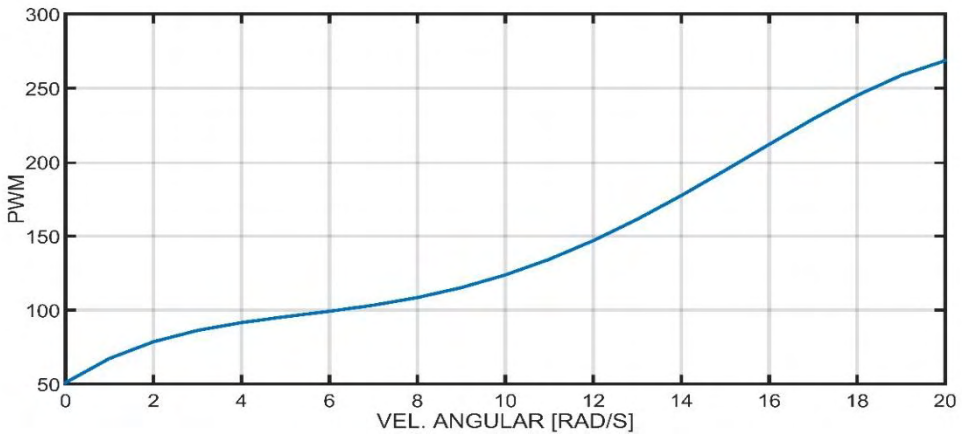
$f(x)$: PWM

$$Modelo\ Motor\ DC(x) = -0{,}0065x^4 + 0{,}2742x^3 - 3{,}2981x^2 + 19{,}3577x + 50{,}9860 \quad (91)$$

Tabla 4. Velocidad angular en rad/s para un valor de pwm.

VEL. ANGULAR [Rad/s]	PWM	VEL. ANGULAR [Rad/s]	PWM
0	30	12,246	150
0	35	12,881	155
0	40	13,152	160
0	45	13,251	165
0	50	13,933	170
0	55	14,100	175
0	60	14,713	180
0	65	14,881	185
0	70	15,346	190
0,500	75	15,269	195
2,815	80	15,309	200
3,575	85	15,490	205
4,521	90	15,796	210
5,350	95	16,140	215
6,353	100	16,446	220
7,040	105	16,558	225
7,818	110	16,413	230
8,524	115	16,486	235
8,836	120	16,916	240
9,618	125	17,738	245
10,140	130	18,640	250
10,760	135	19,066	255
11,568	140	19,066	255
11,974	145		

Figura 42. Modelo matemático motor DC que posee el Arduino-Robot.

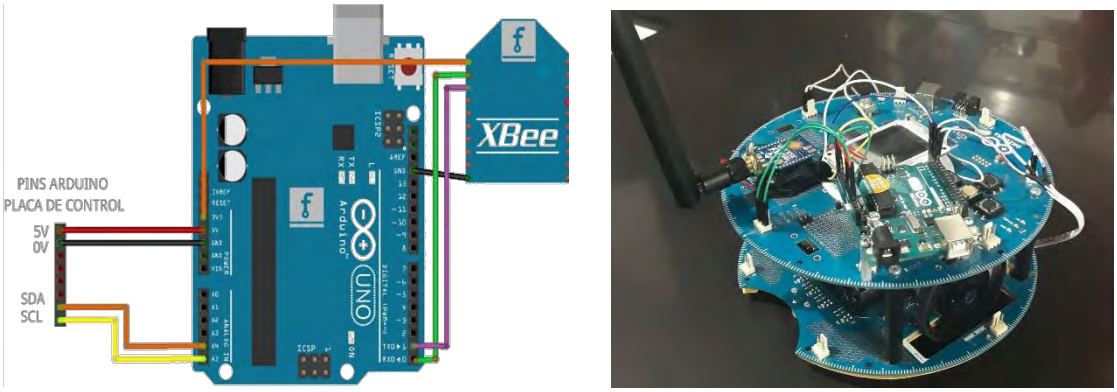


Fuente: Esta investigación.

Cabe aclarar que, para complementar la operación de esta plataforma robótica, se agregó un arduino-uno, conectado por I2C, para administrar la información del

puerto serial usado en las comunicaciones inalámbricas. La conexión y organización final de la plataforma robótica se muestra en la Figura 43.

Figura 43. Conexiones y disposición final de la plataforma robótica.



Fuente: Esta investigación.

3.2.4. Módulos Xbee Pro S3

Este tipo de módulo de comunicación es apropiado por sus características de alcance y penetración, consecuencia de su operación a una frecuencia de 900 MHz.

En el caso de un único robot se configuró una red punto a punto. El algoritmo de comunicación, básicamente cumple con enviar las posiciones calculadas mediante visión por computador usando el puerto serial. (Ver anexo 6)

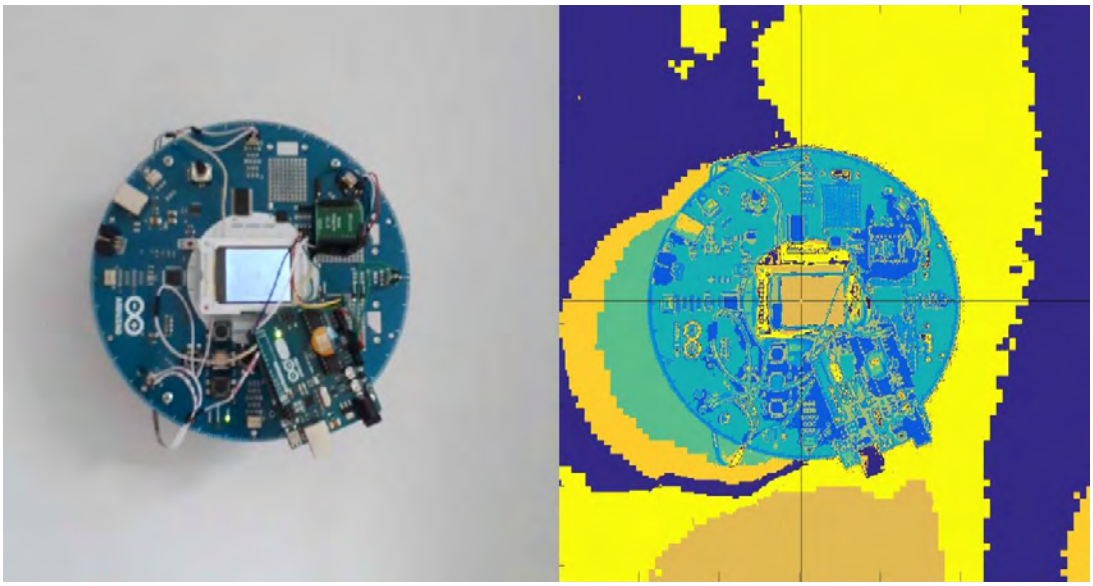
3.2.5. Posicionamiento a partir de visión por computador

El objetivo primordial de este componente es la identificación y rastreo del robot cuándo está navegando por el entorno. Así, se tiene la localización que es retroalimentada al control de posición, que permite seguir la trayectoria planeada.

Para esto se utiliza una cámara de celular Samsung J7, enlazada por comunicación IP al computador mediante la aplicación “IP Webcam” de libre difusión en “*Play Store*” de google. Se configura una resolución de 720 x 720, que es suficiente para identificar con claridad los objetos presentes en un marco de vídeo con una relación de pixel y su equivalente área en el entorno de navegación igual a 0.0577 cm^2 por pixel.

El proceso para la detección del espacio de trabajo con la cámara es el siguiente: En primer lugar, y antes de iniciar la tarea de navegación, se toma una foto del entorno en presencia del robot a interactuar. Luego, se aplica "*kmeans"* (Ver anexo 5), con 10 centroides para el primer k-medias, para segmentar la imagen en una cantidad de grupos representativos de los cuales, el usuario tiene la oportunidad de escoger cuál o cuáles representan el objeto de interés para rastrear y que determinan el número de inicio para el k-medias que se ejecuta. Esto se presenta en la Figura 44.

Figura 44. Imagen segmentada aplicando K-medias e identificador de entrada para la selección de grupo representativo que caracteriza al robot.



Fuente: Esta investigación.

Una vez seleccionado el grupo perteneciente a la ubicación del robot, se obtiene un valor para centroide, igual al promedio de los valores RGB de todo el grupo y su desviación estándar para establecer un rango de valores admisibles que pueden considerarse para la localización del robot. Con estos valores, se establece un filtro de binarización que se aplica a todos los marcos del vídeo durante la navegación, caracterizando al objeto de interés con 1 (blanco) y 0 (negro) a aquello que no está dentro del grupo representativo.

Finalmente, con los parámetros del filtro establecidos, se aplica binarización a todos los marcos del vídeo y con la función *regionprops* de Matlab se obtiene el centroide del objeto de interés, que es marcado con una "x" e identifica en tiempo casi-real la ubicación en coordenadas x, y del robot en el espacio de trabajo.

4. METODOLOGÍA DE COMPARACIÓN ANALÍTICA Y ANÁLISIS DE RESULTADOS

4.1. INTRODUCCIÓN

La técnica base de esta investigación corresponde a la observación científica de tipo directa. Para ello, se han propuesto criterios de evaluación que permiten identificar ventajas y desventajas de cada uno de los métodos frente a distintos escenarios de prueba. Cada validación, permite evidenciar la aplicabilidad de los métodos ante otros problemas de estudio determinando a su vez sus limitaciones.

Se plantea realizar un número *N* variable de pruebas a cada técnica de planeación, modificando los parámetros que las definen, para así poder establecer la relación entre estos y su comportamiento, tal número de pruebas se define independientemente para cada una de ellas, dependiendo de qué prueba en específico sea necesario realizar, la cantidad de datos para que dicha prueba sea diciente o la complejidad en la toma de muestras. Establecer dicha relación es muy relevante, puesto que permite hacer una comparación más justa entre las técnicas e identificar de una mejor manera las características de las curvas que generan.

La evaluación del comportamiento difiere para cada técnica, por ser éstas de naturaleza distinta, por ejemplo, para las técnicas basadas en campos potenciales es necesario considerar la modificación de los parámetros del campo (k, ξ,*etc.*), sin embargo, este carácter pierde relevancia para técnicas basadas en grafos como RRT* dónde, en cambio, será necesario hacer variaciones del número de nodos para la ejecución.

Aunque las técnicas se basen en metodologías distintas, la comparación no pierde validez si está dirigida, no a la metodología, si no al resultado es decir a las trayectorias generadas, puesto que es el mismo para todas: curvas en $\mathbb{R}^2$ que cumplen la misma función, servir como guía a un móvil de un punto a otro en el espacio de trabajo.

Las características de las curvas que se buscan están relacionadas con los índices de comparación, que son los criterios con que se evalúa una trayectoria enfocados netamente a la aplicación práctica: la distancia de la ruta, su suavidad, la cercanía con los objetos o bien la posibilidad de la aplicación en tiempo real, son algunas de ellos. Con esto el objetivo que se pretende alcanzar es una base experimental para que el lector interesado en la implementación pueda juzgar cuál de las técnicas se ajustan más a la necesidad específica que pretende resolver, así habrá aplicaciones en las que importe más la distancia total de la ruta, que la cercanía con los objetos, o bien, en las que se sacrifique suavidad o incluso la probabilidad de encontrar una solución, por la posibilidad de la aplicación en tiempo real.

No es suficiente con la variación de los parámetros en cada técnica, es necesario probarlas en distintos escenarios para comprobar la validez de los resultados obtenidos, pero dado que el objetivo es la aplicación práctica, estos escenarios no deben ser elegidos arbitrariamente, deben ser los que se presentan con más regularidad. En [14] y [36] se puede observar que dichas configuraciones tienen las siguientes características: Obstáculos que rodean al punto inicial impidiendo la conexión directa entre éste y el objetivo, obstáculos muy cercanos los unos de los otros, que forman en ocasiones paredes estrechas, por las que se debe generar la trayectoria u obstáculos dispersos arbitrariamente en el espacio. A estas configuraciones las llamamos Trampa, Pasaje estrecho y Arreglo de obstáculos respectivamente y es en cada una de ellas donde se realizan las pruebas para todas las técnicas.

Por otra parte dentro de la tarea de navegación, se evidencia los campos de profundización para trabajo futuro en relación con el desarrollo para planificación de trayectorias. Para ello, como aporte y propuesta se añade contenido de aproximación en las áreas de control de movimientos, generación de trayectoria, implementación en tiempo real y desarrollo de múltiples robots cooperativos.

En la primera aproximación se expone un control de movimientos lineal bajo la técnica de escalado de velocidad aplicado a un modelo cinemático de un robot de tracción diferencial. En la segunda se presenta una alternativa frente al problema de restricciones no-holonómicas a partir de interpoladores Spline y B-Spline que garantizan la existencia de al menos la primera derivada y por lo tanto la suavidad de la ruta. La tercera aproximación muestra el desarrollo de la implementación del algoritmo de campos potenciales artificiales en una plataforma robótica Arduino-Robot, indicando pasos a seguir para el diseño y adecuación en hardware y software, el modelado matemático de los motores y la etapa de posicionamiento con visión por computador. Finalmente la cuarta y última aproximación muestra resultados de la simulación de la extensión de los algoritmos de campos de potencial artificial y descomposición por celdas trapezoidales, para la navegación de múltiples robots sin colisión. Como complemento a la metodología de comparación establecida, expone índices para el análisis del caso de múltiples robots en relación con el método de obstáculos de velocidad recíproca.

4.2. METODOLOGÍA DE COMPARACIÓN

4.2.1. Índices caso de un único robot

a) Eficacia de la solución encontrada

Se define como la solución u objetivo principal al problema de planeación de trayectoria, sea encontrar o no una ruta libre de colisión desde un punto de partida

hasta un punto de llegada. Se evalúa, tanto la posibilidad de encontrar una ruta, como las condiciones necesarias para ello: variación de los parámetros que definen el espacio de trabajo o de los que son propios para cada técnica.

b) Longitud total de la ruta

Es la longitud de la ruta medida desde el punto inicial hasta el final. Puede considerarse como la suma de las distancias euclidianas entre cada uno de los puntos de control que definen una ruta, Sección 2.1.7.

Es la longitud de la ruta medida desde el punto inicial hasta el final. Puede considerarse como la suma de las distancias euclidianas entre cada uno de los puntos de control que definen una trayectoria. El foco para el estudio de las rutas en este índice son los aciertos, se evalúa la longitud de las rutas generadas de las muestras tomadas tras la variación de los parámetros, con ayuda de los métodos estadísticos de regresión polinomial. Con ello se pretende evaluar si existe relación entre las distancias obtenidas y los parámetros que definen los métodos, o su espacio de trabajo, generando una curva polínómica que se ajusta a los datos.

Las variables de la regresión que resultan relevantes para nuestro estudio son:

SSE: La suma de los errores residuales al cuadrado. Lo que se busca es que este valor sea cercano a 0, en tal caso el polinomio que se genera para definir el comportamiento de los datos, se ajusta de la manera más equidistante posible, entre todos ellos, y es llamado *polinomio de regresión*.

R^2: El coeficiente de determinación, un valor de 0 a 1 que da indicios de cuan diciente es la variable independiente respecto a la dependiente según el polinomio de regresión. Es cercano a 1 si la variable independiente describe completamente el comportamiento de la dependiente (la longitud de la ruta) y cercano a 0 en el caso contrario.

RMSE: La desviación estándar, respecto al polinomio de regresión. Es una medida de cuan dispersos se encuentran los datos respecto al polinomio de regresión.

Estas variables de la regresión, llamadas variables de ajuste, pueden ser consultadas a mayor detalle en cualquier libro de fundamentos de estadística.

Aquí se utilizará la distancia mínima del polinomio de regresión, dependiendo del rango del parámetro en el que se encuentran los aciertos, para ser comparado con los otros métodos. Así mismo también es importante la dispersión de los datos (Variable *RMSE*), para saber qué tan cortas pueden llegar a ser las curvas.

Las variables *SSE y* R^2 Son utilizadas para escoger el grado de la regresión polinomial.

c) Seguridad de la ruta

La posibilidad del contacto con obstáculos. Se mide la distancia euclidiana desde puntos tomados a lo largo de la ruta hasta los límites del objeto, tomando la distancia mínima encontrada en relación a las dimensiones del Arduino-Robot (es decir a su diámetro igual a 18.5 cm) y las porciones de ruta que se acercan a este valor. Nótese que tal distancia dependerá de la forma de los objetos, es decir de la definición del espacio de trabajo, así que es necesario hacer un balance entre las distintas técnicas tomando como base algún criterio de forma.

Por otra parte se referirá a una aproximación al análisis de suavidad de las mismas, refiriéndose a la dispersión de las muestras, medida tomada de los métodos de regresión. En este caso se hará énfasis al *RMSE* que muestra cuan dispersos pueden ser los datos, se entiende que curvas suaves también pueden generar datos muy dispersos, pero como se verá en los resultados, el cuan dispersos son los datos de distancias, nos da una aproximación a la suavidad de las curvas.

d) Complejidad de operación

Relacionada con una descripción de la complejidad de los algoritmos de acuerdo al tiempo usado para entregar resultados a partir de su propuesta teórica de operación. Se hace considerando la eliminación de rutinas de visualización y sentencias innecesarias dentro de la programación en líneas de código.

Su estudio permite diferenciar las técnicas como las que tienen posibilidad de aplicación en tiempo real y aquellas que se limitan a desarrollos fuera de línea.

4.2.2. Procedimientos para la adquisición de muestras

Campos potenciales artificiales

Principalmente se tiene en cuenta la forma de la función que representa el campo potencial artificial.

Para la comparación se han seleccionado la función de navegación como enfoque global y la función U_{att_1} con funciones de repulsión tipo armónica como enfoque local. Esto debido a sus favorables características identificadas por el autor en [36]. De esta manera se realiza un número *N* de pruebas del algoritmo utilizando la técnica de gradiente descendente con variaciones del parámetro *k* (para la función de navegación), y variaciones de los parámetros ξ y λ para la función de campo potencial local, tomando para U_{att_1}, $m = 2$. El rango de variación de cada factor depende de cuan dicientes sean los resultados en el escenario, el entorno de navegación, la posición de los obstáculos y los puntos de inicio y llegada del móvil. La Tabla 5 muestra los datos que se definen en la programación de las funciones.

Tabla 5. Valor numérico de parámetros. PF

ESCENARIO TRAMPA	ESCENARIO PASAJE ESTRECHO	ESCENARIO ARREGLO DE OBSTÁCULOS
Posición de inicio y llegada: $q_{start} = [5,3.5]$ $q_{tar} = [5,8]$ **Posición de los obstáculos:** $Obstáculo_1 = [3.5,5.5]$ $Obstáculo_2 = [4.5,5.5]$ $Obstáculo_3 = [5.5,5.5]$ $Obstáculo_4 = [6.5,5.5]$ $Obstáculo_5 = [3.5,3.5]$ $Obstáculo_6 = [6.5,3.5]$ $Obstáculo_7 = [3.5,4.5]$ $Obstáculo_8 = [6.5,4.5]$ **Rango de los parámetros de la función de navegación:** $q_0 = [5,5]$ r_0=5 $1 < k < 20$ **Rango de los parámetros de la función de campo local, armónica:** $0 < \xi < 30$ $-100 < \lambda < 0$ **Rango de parámetros del algoritmo:** Iteraciones de gradiente descendiente=500 **Número de pruebas realizadas** *N*=1000	**Posición de inicio y llegada:** $x_0 = [5,3]$ $q_{tar} = [8,6]$ **Posición de los obstáculos:** $Obstáculo_1 = [3.5,1.5]$ $Obstáculo_2 = [4.5,1.5]$ $Obstáculo_3 = [5.5,1.5]$ $Obstáculo_4 = [6.5,1.5]$ $Obstáculo_5 = [3.5,2.5]$ $Obstáculo_6 = [6.5,2.5]$ $Obstáculo_7 = [3.5,3.5]$ $Obstáculo_8 = [6.5,3.5]$ $Obstáculo_9 = [4,4.5]$ $Obstáculo_{10} = [6,4.5]$ $Obstáculo_{11} = [4,5.5]$ $Obstáculo_{12} = [6,5.5]$ $Obstáculo_{13} = [4,6.5]$ $Obstáculo_{14} = [6,6.5]$ $Obstáculo_{15} = [4,7.5]$ $Obstáculo_{16} = [6,7.5]$ $Obstáculo_{17} = [3.75,3.75]$ $Obstáculo_{18} = [6.25,3.75]$ **Rango de los parámetros de la función de navegación:** $q_0 = [5,5]$ r_0=5 $0 < k < 30$ **Rango de los parámetros de la función de campo local, armónica:** $0 < \xi < 30$ $-100 < \lambda < 0$ **Rango de parámetros del algoritmo:** Iteraciones de gradiente descendiente=1000 **Número de pruebas realizadas** *N*=1000	**Posición de inicio y llegada:** $q_{start} = [3,7]$ $q_{tar} = [7,7]$ **Posición de los obstáculos:** $Obstáculo_1 = [2,4]$ $Obstáculo_2 = [5,6]$ $Obstáculo_3 = [5,8]$ $Obstáculo_4 = [5,9]$ **Rango de los parámetros de la función de navegación:** $q_0 = [5,5]$ r_0=5 $0 < k < 10$ **Rango de los parámetros de la función de campo local, armónica:** $0 < \xi < 100$ $-100 < \lambda < 0$ **Rango de parámetros del algoritmo:** Iteraciones de gradiente descendiente=1000 **Número de pruebas realizadas** *N*=1000

Nótese que en la Tabla 5, los valores asignados a q_0 y r_0 se mantienen constantes, puesto que como se verá en la Sección 4, no son relevantes en la generación de la trayectoria.

Optimización por enjambre de partículas

Este método es una técnica alternativa a la de gradiente descendente en campos potenciales artificiales. Los rangos de variación de los parámetros para las funciones de navegación y campo local, se mantienen iguales. En este caso se varían un número *N* de ejecuciones los factores que definen el algoritmo PSO, estos son el número *n* de partículas y el radio *r* de distribución.

La Tabla 6 muestra los valores con que se programa el algoritmo.

Tabla 6 Valor numérico de parámetros. PSO

ESCENARIO TRAMPA	ESCENARIO PASAJE ESTRECHO	ESCENARIO ARREGLO DE OBSTÁCULOS
Posición de inicio y llegada: $q_{start} = [5,3.5]$ $q_{tar} = [5,8]$ **Rango de los parámetros de la función de navegación para $\boldsymbol{n} =$** 150 y $r = 0.5$: $q_0 = [5,5]$ r_0=5 $0 < k < 20$ **Iteraciones criterio de parada =300** **Rango de parámetros del algoritmo para $\boldsymbol{k} = \mathbf{3.5}$:** $0 < n < 140$ $1 < r < 3.5$ **Número de pruebas realizadas** *N*=1200	**Posición de inicio y llegada:** $q_{start} = [5,3]$ $q_{tar} = [8,6]$ **Rango de los parámetros de la función de navegación para $\boldsymbol{n} =$ $\mathbf{150}$ y $\boldsymbol{r} = \mathbf{0.5}$:** $q_0 = [5,5]$ r_0=5 $0 < k < 30$ **Iteraciones criterio de parada =300** **Número de pruebas realizadas** *N*=1000	**Posición de inicio y llegada:** $q_{start} = [3,7]$ $q_{tar} = [7,7]$ **Rango de los parámetros de la función de navegación para $\boldsymbol{n} =$ $\mathbf{150}$ y $\boldsymbol{r} = \mathbf{0.5}$:** $q_0 = [5,5]$ r_0=5 $0 < k < 20$ **Iteraciones criterio de parada =500** **Número de pruebas realizadas** *N*=1500

Exploración rápida de árboles aleatorios*

El comportamiento de este algoritmo lo determina el número máximo de nodos permitidos Nod_{max}, la distancia de avance EPS (asignada para cada rama) y el radio de búsqueda de nodos aún más cercanos r.

Los valores de rango de variación y demás constantes que definen el algoritmo se aprecian en la Tabla 7.

Tabla 7 Valor numérico de constantes. RRT*

ESCENARIO TRAMPA	ESCENARIO PASAJE ESTRECHO	ESCENARIO ARREGLO DE OBSTÁCULOS
Posición de inicio y llegada: $q_{start} = [5,3.5]$ $q_{tar} = [5,8]$ **Posición de los obstáculos:** $Obstáculo_1 = [3,3,1,3]$ $Obstáculo_2 = [4,5,2,1]$ $Obstáculo_3 = [6,3,1,3]$ **Rango de parámetros del algoritmo:** $0 < Nod_{max} < 180$	**Posición de inicio y llegada:** $q_{start} = [5,3]$ $q_{tar} = [8,6]$ **Posición de los obstáculos:** $Obstáculo_1 = [3,1,1,3]$ $Obstáculo_2 = [6,1,1,3]$ $Obstáculo_3 = [4,1,2,1]$ $Obstáculo_4 = [3.5,4,1,4]$ $Obstáculo_5 = [5.5,4,1,4]$ **Rango de parámetros del algoritmo:** $0 < EPS < 2$ $100 < Nod_{max} < 500$ $0 < r < 6$	**Posición de inicio y llegada:** $q_{start} = [3,7]$ $q_{tar} = [7,7]$ **Posición de los obstáculos:** $Obstáculo_1 = [1.5,3.5,1,1]$ $Obstáculo_2 = [4.5,6,1,1]$ $Obstáculo_3 = [4.5,7.5,1,2]$ $Obstáculo_4 = [6.5,5.5,1,1]$ **Parámetros del algoritmo:** $0 < Nod_{max} < 150$

Descomposición por celdas trapezoidales

Este algoritmo se fundamenta en la definición de un espacio trapezoidal a partir del cual se forma el grafo de conectividad. Dentro de este proceso no existen variables a modificar puesto que se utilizan los vértices de los polígonos para establecer segmentos de recta que definen los trapezoides a crear. Por este motivo, en este método se realiza un número *N* de pruebas variando la ubicación de los vértices de los polígonos (quienes a su vez definen los obstáculos en el entorno).

La Tabla 8 muestra las constantes requeridas por el algoritmo. Las modificaciones de forma de cada polígono se hacen de forma aleatoria.

Tabla 8 Valor numérico de constantes. CD

ESCENARIO TRAMPA
Posición de inicio y llegada: $q_{start} = [5,3.5]$ $q_{tar} = [5,8]$
ESCENARIO PASAJE ESTRECHO
Posición de inicio y llegada: $q_{start} = [5,3]$ $q_{tar} = [8,6]$
ESCENARIO ARREGLO DE OBSTÁCULOS
Posición de inicio y llegada: $q_{start} = [3,7]$ $q_{tar} = [7,7]$

4.2.3. Procedimiento para el análisis de muestras

1) Identificación de los parámetros

Con base en la caracterización de cada uno de los algoritmos aplicados a la planificación de trayectoria, se seleccionan los parámetros de ajuste que alteran el comportamiento de cada una de las técnicas. Una vez seleccionados, se modifica su valor dentro de un rango de acción para establecer la dependencia de operación respecto cada uno de ellos. De esta manera se varía el número de aciertos y fracasos, complejidad computacional (relacionada con la cantidad de iteraciones necesarias para encontrar la solución) y la planificación de la ruta en tanto a la distancia total recorrida y su seguridad conforme se expresó en la Sección 4.2.1.

El establecimiento de rangos de variación depende del tipo de resultado que ofrezca la aplicación de distintos valores y su relación de afectación sobre la planeación. Esto se verá más adelante en resultados, donde por ejemplo en RRT* (caso pasaje estrecho) para un número de nodos inferior a 100 se observa que ya no existen aciertos y para un número mayor de 500 sí. Como tal el rango 100-500 es concluyente.

2) Comparación

Para efectos de la comparación, en primer lugar se analiza la disposición de las rutas en cada uno de los escenarios. Posteriormente se estudia cada uno de los índices dependiendo de los parámetros analizados, concluyendo respecto a todas las técnicas de planeación investigadas. Cada índice se evalúa teniendo en cuenta las observaciones respecto distancia total recorrida, número de iteraciones necesarias para encontrar una solución y el porcentaje de rutas cercanas a los obstáculos (colisión inminente con el armazón) según todos y cada uno de los algoritmos. Todo ello dependiendo de las variaciones de los parámetros de ajuste.

4.3. APROXIMACIÓN AL TRABAJO FUTURO

4.3.1. Control de movimientos

El control de movimientos dentro de la tarea de navegación puede aplicarse a modelos matemáticos de tipo cinemático o dinámico que dependen del tipo de plataforma robótica a simular o implementar. En esta investigación se estudia la planificación para un robot de tracción diferencial, para ello se hace un estudio cinemático al que debe aplicarse un control de movimientos que permita la ubicación en cada uno de los puntos de control sobre la trayectoria Secciones 2.1.6 y 2.1.8. Los resultados de esta sección dan a conocer los simulaciones de la técnica de control lineal expuesta en [44] aplicando constantes arbitrarias y escalado de velocidad.

4.3.2. Generación de trayectorias

Los resultados de esta sección permiten observar las ventajas y desventajas de los interpoladores Spline y BSpline (analizados teóricamente en la Sección 2.1.7) sobre las trayectorias generadas por cada una de las técnicas en los tres escenarios de navegación seleccionados. Lo hace mediante gráficas representativas y tablas de distancia total recorrida *Dist* para la trayectoria original creada *SDist* para la trayectoria interpolada por Spline y *BSDist* para la trayectoria interpolada por BSpline.

4.3.3. Implementación en plataforma física

Los estudios en navegación dentro de la robótica, no pueden limitarse a desarrollos virtuales, debe tenerse en cuenta la posibilidad de aplicación en tiempo real. Con base en lo expuesto en la Secciones 3.2.2-3.2.5 se presenta los resultados de la aplicación del algoritmo de campos de potencial artificial operando con una función de navegación de tipo global. Se expone metodológicamente los procedimientos sugeridos para la implementación del control de movimientos de acuerdo a las limitaciones que impone la plataforma Arduino-Robot y se demuestra que es posible llevar a la práctica las técnicas de planeación analizadas teniendo en cuenta la información contenida en gráficas y tablas que incluyen la información del tiempo sobre la trayectoria y la distancia total recorrida en línea.

4.3.4. Índices caso de múltiples robots

A continuación se exponen los índices para el caso de múltiples robots, como aproximación de la metodología de comparación propuesta a este campo de estudio.

a) Eficacia de la solución encontrada

Se define como la solución al problema de navegación de múltiples robot. Es decir, encontrar una ruta libre de colisión con obstáculos y agentes presentes en un mismo entorno, tal que lleve a los agentes involucrados a sus destinos programados desde una posición de partida.

b) Evasión y Ruta establecida

Se consideran dos aspectos: *i)* La seguridad de las rutas planeadas, es decir, la probabilidad de contacto con obstáculos y otros robots y *ii)* el valor de distancias totales recorridas, definida como la suma de las distancias euclidianas entre puntos de control que definen las trayectorias de referencia.

c) Tiempo de ejecución

Relacionada con una descripción de la complejidad de los algoritmos de acuerdo al tiempo usado para entregar resultados a partir de su propuesta teórica de operación.

5. RESULTADOS

5.1. METODOLOGÍA DE COMPARACIÓN ANALÍTICA

5.1.1. RELACIÓN ENTRE PARÁMETROS E ÍNDICES

CAMPOS POTENCIALES ARTIFICIALES-GRADIENTE DESCENDIENTE

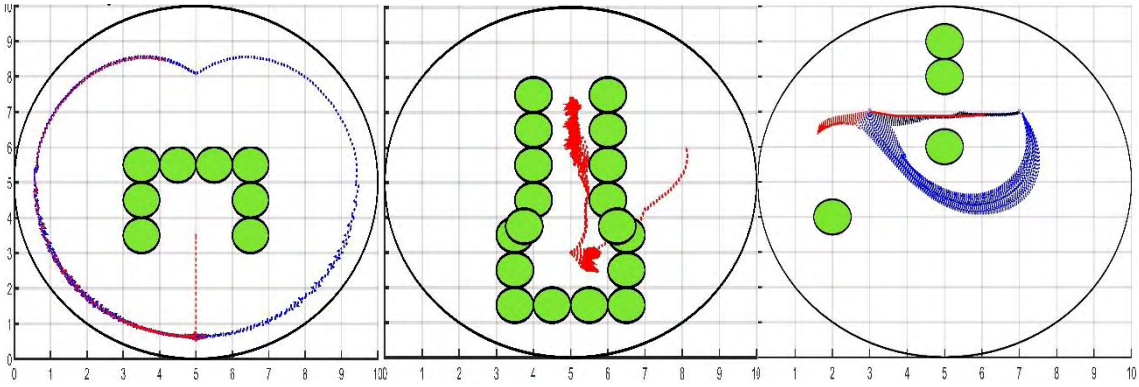
a) Eficacia de la solución encontrada

Aplicación sobre función de navegación

Trampa

En la Figura 45 izquierda se muestran las rutas generadas con la variación del parámetro *k* para el escenario trampa. Se observan cuatro tipos de comportamiento:

Figura 45. Rutas planeadas dependiendo de k. Función de navegación para gradiente descendiente



Fuente: Esta investigación.

El primero corresponde a los aciertos, se puede encontrar cuando $5.997 < k \leq 6.415$ aproximadamente y las rutas generadas se identifican con color azul.

Los siguientes comportamientos no se consideran aciertos y sus rutas generadas se identifican con color rojo:

Para el rango $0 < k \leq 5.997$ la gran mayoría de rutas generadas parecen terminar en un mínimo local ubicado aproximadamente en las coordenadas $(5,0.4)$.

Cuando $6.415 < k \leq 20$ las rutas generadas parten en la dirección de las rutas acertadas hacia la meta, pero no logran alcanzarla, esto se debe al límite de

iteraciones del algoritmo de gradiente descendente que hemos impuesto y que tiene una respuesta exponencial respecto a *k* en el rango de los aciertos. Este comportamiento se adecua para un valor de *k* cercano a 7, después del cual se crea un mínimo local en las cercanías del punto de inicio, siendo éste sigue siendo el comportamiento dominante para los $k > 20$.

Pasaje

En la Figura 45 centro se muestran las rutas generadas con la variación del parámetro *k* en el escenario pasaje estrecho. Se observan tres tipos de comportamiento:

Los siguientes comportamientos no se consideran aciertos y sus rutas se identifican con color rojo:

Para los rangos $0 < k < 8.182$ y $9.645 < k \leq 30$ las rutas generadas mayoritariamente, terminan en mínimos locales ubicados dentro del pasaje. Se puede comprobar que para los $k > 30$, éste sigue siendo el comportamiento dominante.

Cuando $8.182 \leq k \leq 9.645$La mayoría de las rutas generadas conectan los puntos de inicio y meta, pero se crean entre los obstáculos. Nótese que este rango se interseca con la totalidad del rango del primer tipo de comportamiento.

Arreglo

En la Figura 45 derecha, se muestran las rutas generadas con la variación del parámetro *k* en el escenario arreglo de obstáculos. Se observan cuatro tipos de comportamiento:

Los siguientes comportamientos corresponden a los aciertos:

Cuando $2.91 \leq k \leq 3.89$ aproximadamente, las rutas generadas se identifican con color azul.

Para $3.89 < k \leq 6.25$ aproximadamente, las rutas generadas se identifican con color negro.

Los siguientes comportamientos no se consideran aciertos y sus rutas se identifican con color rojo:

Para el rango $0 < k < 2.91$ la gran mayoría de rutas generadas parecen terminar en un mínimo local ubicado aproximadamente en las coordenadas $(1.5,6.5)$.

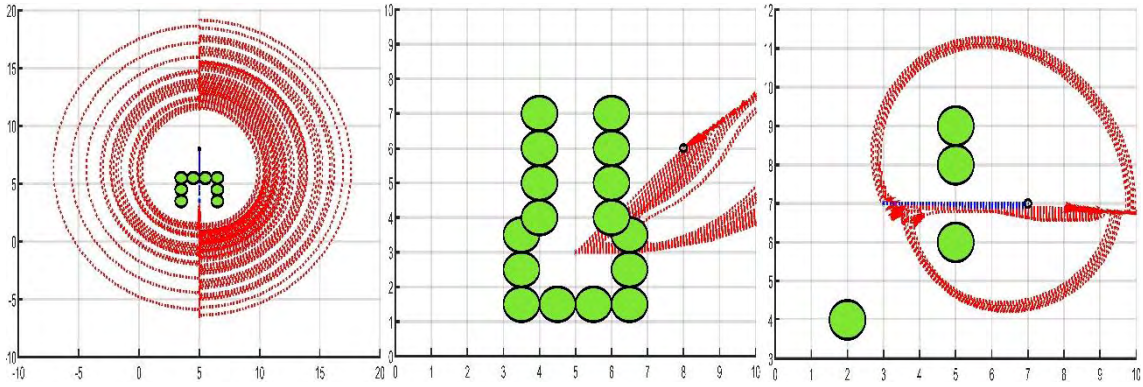
Cuando $6.25 < k \leq 10$ La mayoría de las rutas se generan en dirección a la meta, pero no logran alcanzarla, por la misma razón que en el caso del escenario trampa, las iteraciones no son suficientes. Para $k > 7$ se generan mínimos locales muy cercanos al punto inicial en donde terminan las rutas. Se puede comprobar que para los $k > 10$, éste sigue siendo el comportamiento dominante.

Aplicación sobre campo local

Trampa

En la Figura 46 izquierda se muestran las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario trampa. Se observan dos tipos de comportamiento.

Figura 46. Rutas dependido de λ y ξ. Campo local para gradiente descendiente.



Fuente: Esta investigación.

Los siguientes comportamientos no se consideran como una solución al problema de planificación y sus rutas se identifican con color rojo:

Para el rango de valores $0 < \xi \leq 7.5$ y $-100 < \lambda < -10.9$, la mayoría de las rutas generadas terminan en mínimos *globales* que se alejan de la meta, creciendo tal distancia en el *eje y* conforme el valor de λ aumenta negativamente. Se puede comprobar que para los $\lambda < -100$, éste sigue siendo el comportamiento dominante.

Para el resto de valores en el rango de evaluación, sobre todo para los $\xi > 13$, las rutas generadas conectan los puntos de inicio y meta, pero se crean entre los obstáculos.

Pasaje

En la Figura 46 centro se muestran las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario pasaje estrecho. Se observa que todas las rutas se crean entre los obstáculos.

Arreglo

En la Figura 46 derecha se muestran las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario arreglo de obstáculos. Se observan cuatro tipos de comportamiento:

Se referirá primero $13 < \xi \leq 100$ y $-100 < \lambda < 0$, donde se encuentra la mayor cantidad de aciertos, se puede observar que el éxito de éstas rutas se debe a en

parte a que las rutas generadas atraviesan el círculo de radio 0.5m que se consideró como criterio de parada.

El resto de rutas generadas no se consideran como una solución al problema de planificación y son identificadas con color rojo.

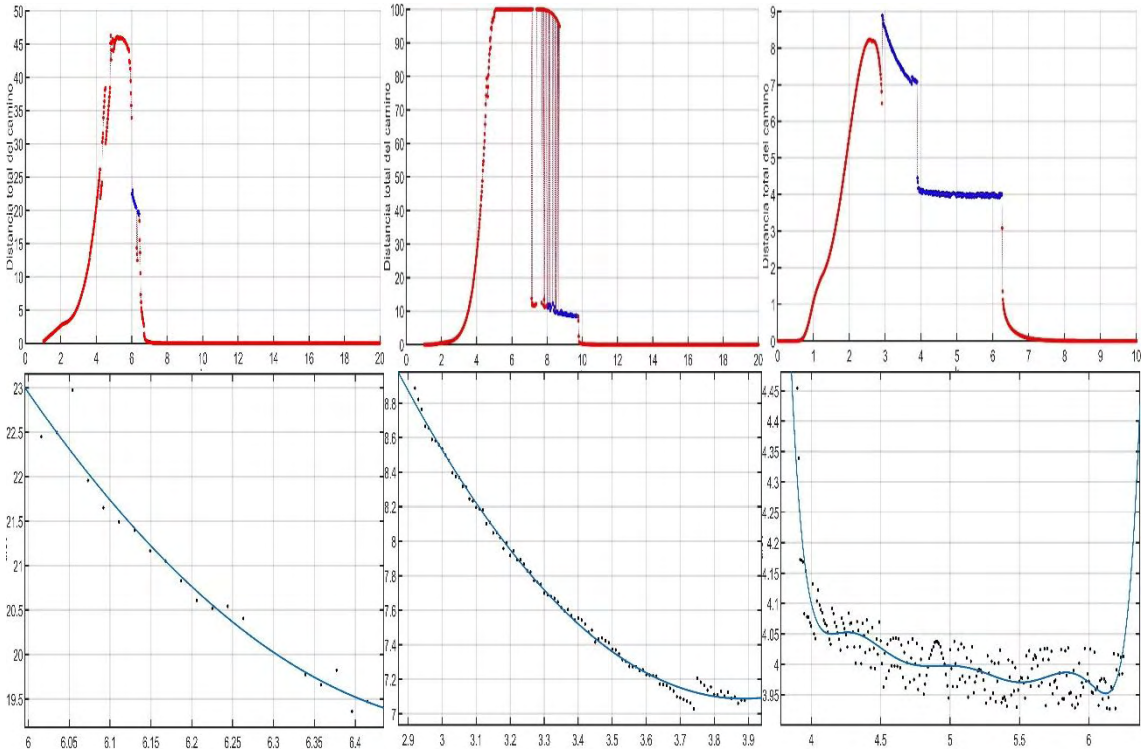
b) Longitud total de la ruta

Aplicación sobre función de navegación global

Trampa

En la Figura 47 superior izquierda se muestra la longitud de las rutas generadas con la variación del parámetro *k* en el escenario trampa. Se puede observar para las rutas acertadas, que la longitud tiende a disminuir cuando el valor de *k* aumenta. Por esta razón, y dada la regularidad de las muestras, se tomara para el análisis, los puntos extremos del rango de aciertos: $k_{lmax} = 6.016$ para evaluar la longitud máxima de la ruta y $k_{lmin} = 6.415$ para la longitud mínima.

Figura 47. Longitud total del camino dependiendo de k para función de navegación. Gradiente descendiente.



Fuente: Esta investigación.

La Figura inferior izquierda se muestra la regresión polinomial de grado 2 para los datos de distancia, donde se logra apreciar el comportamiento decreciente de la longitud respecto a *k*. En la Tabla 9 se observan los resultados de las variables de ajuste y las longitudes mínima y máxima, evaluadas en $k_{lmin}\ y\ k_{lmax}$ respectivamente.

Tabla 9. Variables de ajuste regresión polinomial para función de navegación por gradiente descendiente escenario trampa.

SSE	R^2	RMSE	Longitud mínima	Longitud máxima
0.7739	0.9625	0.2199	19.47	22.74

Pasaje

No existen rutas acertadas para este escenario. Los valores de la Tabla 10, se relacionan con los datos de la Figura 47 inferior centro.

Tabla 10. Variables de ajuste regresión polinomial para función de navegación por gradiente descendiente escenario pasaje estrecho.

$\mathbf{2.92 \leq k \leq 3.89}$			
SSE	R^2	**RMSE**	**Longitud máxima**
0.0934	0.9964	0.03136	8.794

Arreglo

En la Figura 47 superior derecha se muestra la longitud de las rutas generadas con la variación del parámetro *k* en el escenario arreglo. En este caso la elección del k_{lmax} se realiza en el rango donde las distancias son mayores, $2.92 \leq k \leq 3.89$, se puede observar que en este caso las longitudes también disminuyen conforme *k* aumenta, se toma $k_{lmax} = 2.893$. Por otra parte, para el rango de aciertos $3.89 < k \leq 6.25$, no se logra observar ningún tipo de comportamiento creciente o decreciente respecto a *k*, por ello se ha elegido como longitud mínima, a la mínima encontrada en el intervalo (que no se interseca) de la función resultante de la regresión polinomial, con $k_{lmin} = 6.128$.

La Figura 47 inferior derecha muestra la regresión polinomial de grado 2 para los datos de distancia, donde se logra apreciar el comportamiento decreciente de la longitud respecto al intervalo $2.92 \leq k \leq 3.89$ y de grado 8 para $3.89 < k \leq 6.25$. En la Tabla 11 se encuentran los resultados de las variables de ajuste y las longitudes mínima y máxima de la ruta, evaluadas en $k_{lmin}\ y\ k_{lmax}$ respectivamente.

Tabla 11. Variables de ajuste regresión polinomial para función de navegación por gradiente descendiente escenario arreglo de obstáculos.

$3.89 < k \leq 6.25$			
SSE	R^2	**RMSE**	**Longitud mínima**
0.2363	0.7218	0.03226	3.952

Aplicación sobre campo local

Trampa

No existen rutas acertadas para este escenario

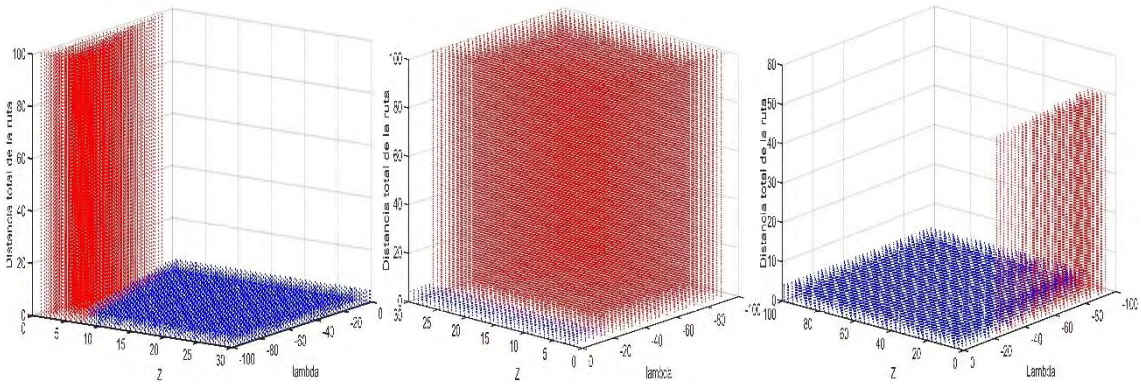
Pasaje

No existen rutas acertadas para este escenario

Arreglo

En la Figura 48 se muestran la longitud de las rutas generadas con la variación de los parámetros λ *y* ξ. Se observa que la longitud de las rutas que aciertan no depende de los parámetros.

Figura 48. Longitud de ruta dependiendo de λ y ξ en campo local. Gradiente descendiente.



Fuente: Esta investigación.

En la Tabla 12 se observan los resultados de la media de los valores de longitud, su desviación estándar y las longitudes mínima y máxima.

Tabla 12. Valores de media, σ y longitud mínima y máxima para distancias de campo local por gradiente descendiente.

Media	$\boldsymbol{Desviación\ estandar}$	Longitud mínima	Longitud máxima
4	0	4	4

c) Seguridad de la ruta

Aplicación sobre función de navegación global

Trampa

El porcentaje de la longitud total de las rutas generadas, que se crea en las "cercanías del obstáculo" (Sección 5.1) en el escenario trampa, para las rutas acertadas, es 0% para todos los *k* del rango.

Además, se puede concluir que para $0 < k \leq 20$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es también del 0%. Aquí se toman tanto los aciertos, como los desaciertos.

Pasaje

En la Figura 49 izquierda se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las "cercanías del obstáculo" con la variación del parámetro *k* en el escenario pasaje. Para las rutas acertadas, es 0% para todos los *k* del rango, puesto que no existen aciertos.

Sin embargo, para $0 < k \leq 20$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 9%. Aquí se toman tanto los aciertos, como los desaciertos.

Arreglo

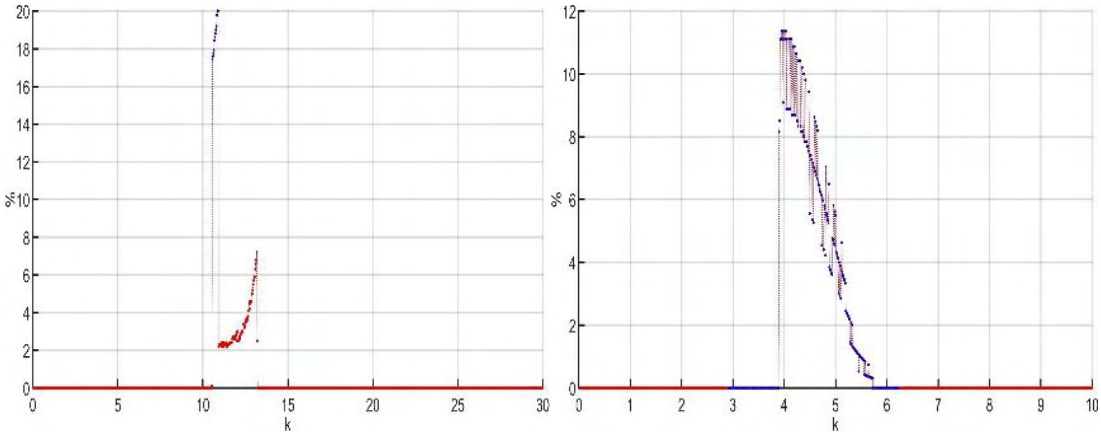
En la Figura 49 derecha se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las "cercanías del obstáculo" con la variación del parámetro *k* en el escenario arreglo. Para las rutas acertadas, la Tabla 13 muestra la media de dichos porcentajes, el porcentaje máximo, mínimo y la desviación estándar.

Tabla 13. Valores de media, σ y porcentaje mínimo y máximo para distancias de función de navegación. Función de navegación para gradiente descendiente.

Media %	$\boldsymbol{Desviación\ estandar}$	Porcentaje mínimo%	Porcentaje máximo%
5.5074	3.5996	0	11.3636

Además, se puede concluir que para $0 < k \leq 10$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 18.3%. Aquí se toman tanto los aciertos, como los desaciertos.

Figura 49. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. Función de navegación para gradiente descendiente.



Fuente: Esta investigación.

OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS

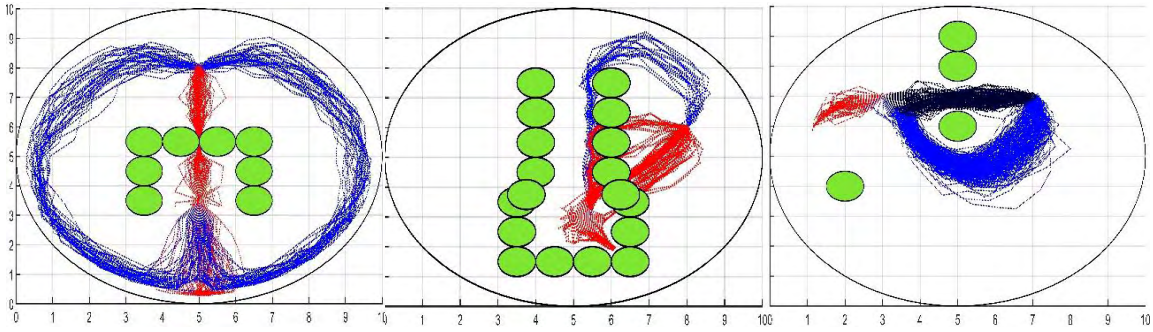
a) Eficacia de la solución encontrada

Aplicación sobre función de navegación global

Trampa

En la Figura 50 izquierda se muestran las rutas generadas con la variación del parámetro *k* en el escenario trampa. Se observan cuatro tipos de comportamiento, el primero corresponde a los aciertos cuando $5.997 < k \leq 6.415$ aproximadamente y las rutas generadas se identifican con color azul. Los comportamientos posteriores no se consideran aciertos y sus rutas generadas se identifican con color rojo.

Figura 50. Rutas dependiendo de k. Función de navegación para PSO



Fuente: Esta investigación.

Para el rango $0 < k \leq 5.997$ gran cantidad de rutas generadas parecen terminar en un mínimo local ubicado aproximadamente en las coordenadas $(5,0.4)$.

Cuando $6.415 < k \leq 20$ las rutas creadas parten en la dirección de la meta aun así no logran alcanzarla. Esto se debe al límite de iteraciones del algoritmo de gradiente descendente que se ha impuesto y que tiene una respuesta exponencial respecto a *k* en el rango de los aciertos. Este comportamiento se adecua para un valor de *k* cercano a 7, después de que se crea un mínimo local en las cercanías del punto de inicio. El comportamiento dominante para los $k > 30$.

Pasaje

En la Figura 50 centro se muestran las rutas generadas con la variación del parámetro *k* en el escenario pasaje. Se observan dos tipos de comportamiento para rutas que no pueden considerarse como aciertos y se grafican en color rojo.

Para los rangos $0 < k < 8.182$ y $9.645 < k \leq 30$ las rutas generadas, terminan en mínimos locales ubicados dentro del pasaje. Se puede comprobar que para los $k > 30$, éste sigue siendo el comportamiento dominante.

Cuando $8.182 \leq k \leq 9.645$ la mayoría de las rutas generadas conectan los puntos de inicio y meta, pero pasan por encima o entre los obstáculos. Nótese que este rango se interseca con la totalidad del rango del primer tipo de comportamiento.

Arreglo

En la Figura 50 derecha se muestran las rutas generadas con la variación del parámetro *k* en el escenario arreglo. Se observan cuatro tipos de comportamiento.

Los siguientes comportamientos corresponden a los aciertos:

Cuando $2.893 \leq k \leq 4.307$ aproximadamente, las rutas generadas se identifican con color azul.

Para $3.773 \leq k \leq 19.88$ aproximadamente, las rutas generadas se identifican con color negro.

Los siguientes comportamientos no se consideran aciertos y sus rutas se identifican con color rojo:

Para el rango $0 < k < 2.893$ la gran mayoría de rutas generadas parecen terminar en un mínimo local ubicado aproximadamente en las coordenadas $(1,6)$.

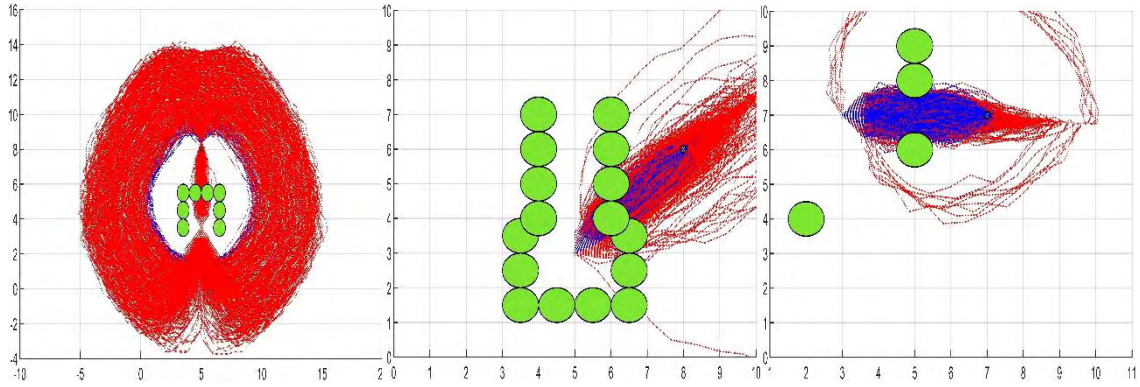
Cuando $17.33 \leq k \leq 20$ la mayoría de las rutas se generan en dirección a la meta, pero no logran alcanzarla terminando en mínimos locales muy cercanos al punto inicial. Se puede comprobar que para los $k > 20$, éste sigue siendo el comportamiento dominante.

Aplicación sobre campo local

Trampa

En la Figura 51 izquierda se muestran las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario trampa. Se observan tres tipos de comportamiento.

Figura 51. Rutas dependido de λ y ξ. Campo local para PSO



Fuente: Esta investigación.

El primero referido al rango $0 < \xi \leq 67.83$ y $-45.55 < \lambda \leq -40.6$, donde se observa que el número de aciertos parece aumentar cuando λ tiende a -40.6 y disminuye cuando ξ crece. Estas rutas son identificadas con color azul.

Los siguientes comportamientos no se consideran como una solución al problema de planificación y sus rutas se identifican con color rojo:

Cuando $0 \leq \xi \leq 100$ y $-40.6 < \lambda < 0$ La mayoría de las rutas generadas conectan los puntos de inicio y meta, pero se crean entre los obstáculos. Se puede comprobar que manteniendo el rango de λ, para los $\xi > 100$, éste sigue siendo el comportamiento dominante.

Para el rango de valores $0 \leq \xi \leq 100$ y $-100 < \lambda < -40.6$, eliminando en esta parte el rango analizado para el primer tipo, la mayoría de las rutas generadas

terminan en mínimos *globales* que se alejan de la meta. La distancia crece en el *eje y* conforme el valor de λ aumenta negativamente. Se puede comprobar que para los $\lambda < -100$ y los $\xi > 100$, éste sigue siendo el comportamiento dominante.

Pasaje

En la Figura 51 centro se muestran las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario pasaje estrecho. Se observa que todas las rutas se crean entre los obstáculos y por lo tanto no existe ningún acierto notable.

Arreglo

En la Figura 51 derecha se muestran las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario arreglo. Se observan cuatro tipos de comportamiento:

Se referirá primero al rango $0 < \xi \leq 100$ y $-28.22 \leq \lambda < 0$ donde se encuentra la mayor cantidad de aciertos. Se observa que el número de aciertos parece aumentar cuando λ disminuye negativamente y ξ crece. Aunque cabe anotar que existen aciertos esporádicamente en todo el rango en el que se realizan las pruebas. Estas rutas se identifican con color azul.

El resto de rutas generadas no se consideran como una solución al problema de planificación y son identificadas con color rojo.

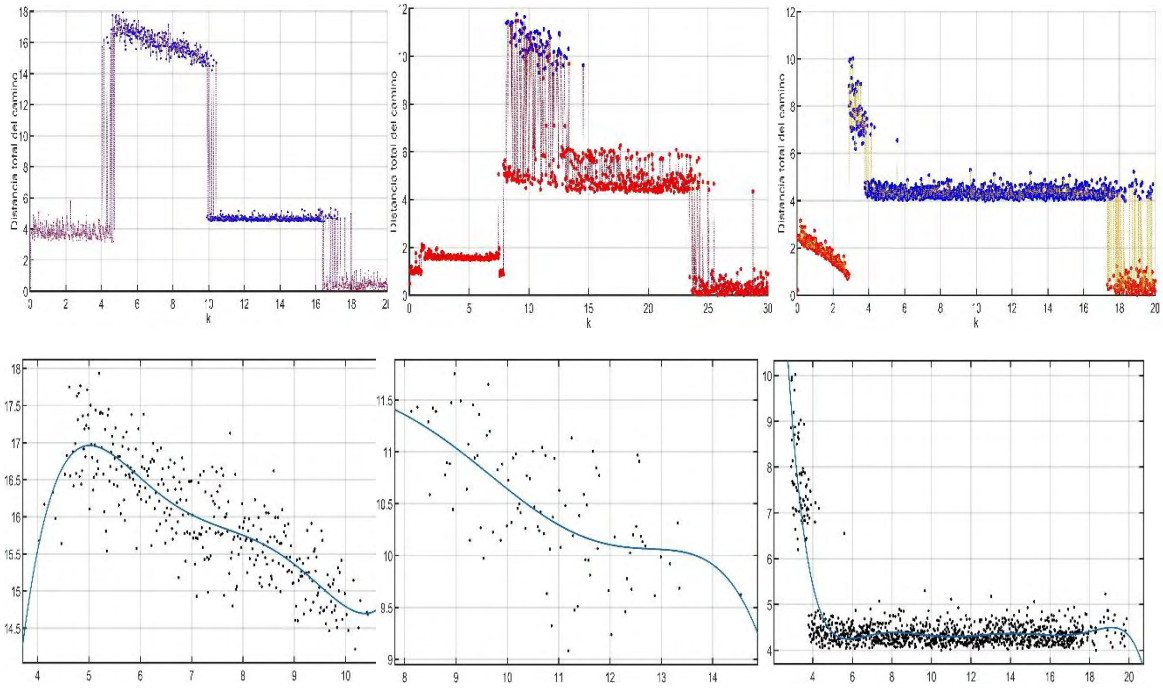
b) Longitud total de la ruta

Aplicación sobre función de navegación global

Trampa

En la Figura 52 superior izquierda se muestra la longitud de las rutas generadas con la variación del parámetro *k* en el escenario trampa. Se puede observar para las rutas acertadas, que la longitud tiende a disminuir cuando el valor de *k* aumenta. Por esta razón, para referirse a las longitudes máxima y mínima de los aciertos, se retoma los rangos de intersección de la variable *k* dónde ocurren, específicamente cuando $4.033 \leq k \leq 4.717$ y $9.9 \leq k \leq 10.43$. Para el análisis se toma los puntos medios: $k_{lmax} = 4.375$ para evaluar la longitud máxima de la ruta y $k_{lmin} = 10.165$ para la longitud mínima.

Figura 52. Longitud total del camino dependiendo de k. Función de navegación para PSO.



Fuente: Esta investigación.

La Figura 52 inferior izquierda muestra la regresión polinomial de grado 5 para los datos de distancia, donde se aprecia el comportamiento decreciente de la longitud respecto a *k*. En la Tabla 14 se observan los resultados de las variables de ajuste y las longitudes mínima y máxima, evaluadas en k_{lmin} y k_{lmax} respectivamente.

Tabla 14. Variables de ajuste regresión polinomial para función de navegación PSO escenario arreglo trampa.

SSE	R^2	RMSE	Longitud mínima	Longitud máxima
57.83	0.7037	0.4212	14.74	16.49

Pasaje

En la Figura 52 superior centro se muestra la longitud de las rutas generadas con la variación del parámetro *k* en el escenario pasaje. En este caso el rango de aciertos ($8.25 \leq k \leq 14.55$) está contenido totalmente en un rango de desaciertos ($7.47 \leq k \leq 25.47$). En el rango de aciertos la distancia nuevamente parece disminuir cuando aumenta el valor de k. Por lo anterior, para la evaluación se toman los límites del rango de aciertos, $k_{lmax} = 8.25$ para evaluar la longitud máxima de la ruta y $k_{lmin} = 14.55$ para la longitud mínima.

La Figura 52 inferior centro muestra la regresión polinomial de grado 5 para los datos de distancia donde se observa el comportamiento decreciente de la longitud respecto a *k*. En la Tabla 15 se observan los resultados de las variables de ajuste y las longitudes mínima y máxima de la ruta evaluadas en $k_{lmin}\ y\ k_{lmax}$ respectivamente.

Tabla 15. Variables de ajuste regresión polinomial para función de navegación PSO escenario pasaje estrecho.

SSE	R^2	RMSE	Longitud mínima	Longitud máxima
19.1	0.4163	0.4741	9.597	11.29

Arreglo

En la Figura 52 superior derecha se muestra la longitud de las rutas generadas con la variación del parámetro *k* en el escenario arreglo. En este caso la elección del k_{lmax} se realiza en el rango donde las distancias son mayores $2.893 \leq k \leq 4.307$. Se puede observar que las longitudes también disminuyen conforme *k* aumenta; dado que aquí el rango de aciertos no tiene intersecciones se toma $k_{lmax} = 2.893$. Por otra parte, para el rango de aciertos $3.773 \leq k \leq 19.88$ no se observa ningún tipo de comportamiento creciente o decreciente respecto a *k*, por ello se ha elegido como longitud mínima, a la mínima encontrada en el intervalo (que no se interseca) de la función resultante de la regresión polinomial, con $k_{lmin} = 5.717$.

La Figura 52 inferior derecha muestra la regresión polinomial de grado 7 para los datos de distancia, en donde se aprecia el comportamiento decreciente de la longitud respecto a *k*. En la Tabla 16 se observan los resultados de las variables de ajuste y las longitudes mínima y máxima de la ruta, evaluadas en $k_{lmin}\ y\ k_{lmax}$ respectivamente.

Tabla 16. Variables de ajuste regresión polinomial para función de navegación PSO escenario arreglo de obstáculos.

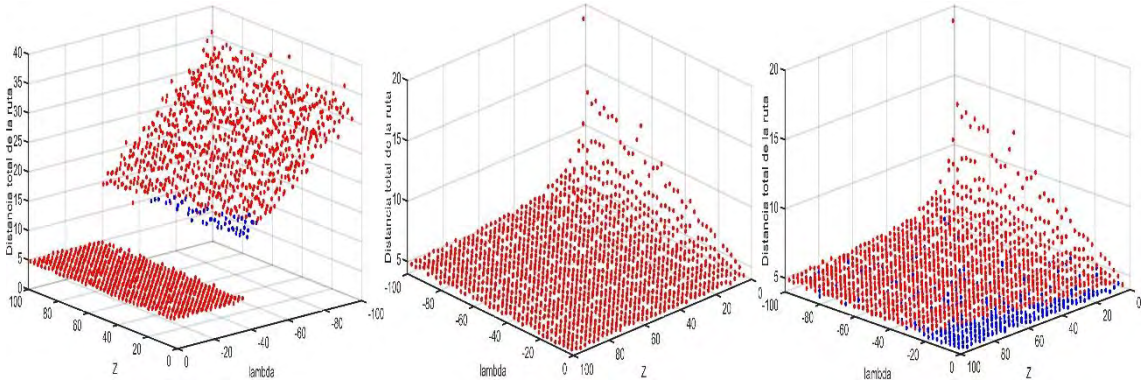
SSE	R^2	RMSE	Longitud mínima	Longitud máxima
168.8	0.8035	0.3857	4.246	9.381

Aplicación sobre campo local

Trampa

En la Figura 53 izquierda se muestra la longitud de las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario trampa. Se observa que la longitud de la ruta depende exclusivamente del parámetro λ, aumentando cuando este último crece negativamente. Esto se debe a que el mínimo *global* se aleja de la meta con el aumento de éste valor, puesto que las funciones que definen los objetos se acentúan cada vez más.

Figura 53. Longitud de ruta dependiendo de λ y ξ. Campo local para PSO.



Fuente: Esta investigación.

En la Tabla 17 se observan los resultados de la media de los valores de longitud, su desviación estándar y las longitudes mínima y máxima.

Tabla 17. Valores de media, σ y longitud mínima y máxima para distancias de campo local para PSO, escenario trampa.

Media	$\boldsymbol{Desviación\ estandar}$	Longitud mínima	Longitud máxima
15.9465	0.8243	14.4340	18.2085

Arreglo

En la Figura 53 derecha se muestran la longitud de las rutas generadas con la variación de los parámetros λ *y* ξ en el escenario arreglo. Se observa que la longitud de la ruta aumenta con forme λ crece negativamente y ξ crece positivamente.

En la Tabla 18 se observan los resultados de la media de los valores de longitud, su desviación estándar y las longitudes mínima y máxima.

Tabla 18. Valores de media, σ y longitud mínima y máxima para distancias de campo local para PSO, escenario arreglo de obstáculos.

Media	$\boldsymbol{Desviación\ estandar}$	Longitud mínima	Longitud máxima
4.4496	0.2006	4.044	5.3310

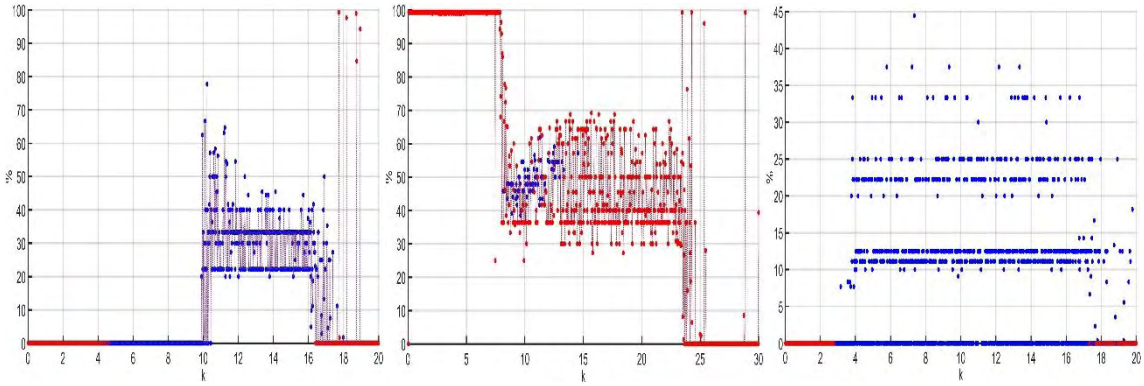
c) Seguridad de la ruta

Aplicación sobre función de navegación global

Trampa

En la Figura 54 izquierda se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las "cercanías del obstáculo" con la variación del parámetro *k* en el escenario trampa. Para las rutas acertadas, el porcentaje es 0% para todos los *k* del rango.

Figura 54. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. Función de navegación para PSO.



Fuente: Esta investigación.

Además, se puede concluir que para $0 < k \leq 20$ el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 34%. Aquí se toman tanto los aciertos, como los desaciertos.

Pasaje

En la Figura 54 centro se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las "cercanías del obstáculo" en relación al parámetro *k* en escenario pasaje. Para las rutas acertadas, la Tabla 19 muestra la media de los

porcentajes de longitud de ruta, el porcentaje máximo, mínimo y la desviación estándar.

Tabla 19. Valores de media, σ y % mínimo y máximo para distancias de función de navegación para PSO, escenario pasaje estrecho.

Media %	$\boldsymbol{Desviación\ estandar}$	Porcentaje mínimo%	Porcentaje máximo%
49.3507	4.6283	38.4615	61.9048

Además, se puede concluir que para $0 < k \leq 20$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 99.84%. Aquí se toman tanto los aciertos, como los desaciertos.

Arreglo

En la Figura 54 derecha se muestra el porcentaje de la longitud total de las rutas generadas que se crea en las "cercanías del obstáculo" con la variación del parámetro *k* en el escenario arreglo. Para las rutas acertadas, la Tabla 20 muestra la media de los porcentajes de longitud de ruta, el porcentaje máximo, mínimo y la desviación estándar.

Tabla 20. Valores de media, σ y % mínimo y máximo para distancias de función de navegación para PSO, escenario arreglo.

Media %	$\boldsymbol{Desviación\ estandar}$	Porcentaje mínimo%	Porcentaje máximo%
9.7056	9.5248	0	44.44

Además, se puede concluir que para $0 < k \leq 10$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 45.67%. Aquí se toman tanto los aciertos, como los desaciertos.

Variación de parámetros del algoritmo

En la Figura 55 superior izquierda y derecha se muestran las rutas generadas con la variación del Número de partículas *n* y el radio de dispersión de las mismas *r* en el escenario tipo trampa. Las rutas que no logran conectar el inicio y la meta, se identifican con color rojo, aquí no se ha identificado con este mismo color a las rutas que se crean entre los obstáculos.

Se puede observar que: La cantidad de aciertos aumenta conforme *r* crece y basta con que $n > 15$ para aumentar a posibilidad de que se generen rutas acertadas, en este caso. En tanto a los caminos creados entre los obstáculos, se puede observar que estos son identificados con un color azul verdoso, que corresponde a las regiones cercanas al límite $r = 3.5$ y $n = 150$.

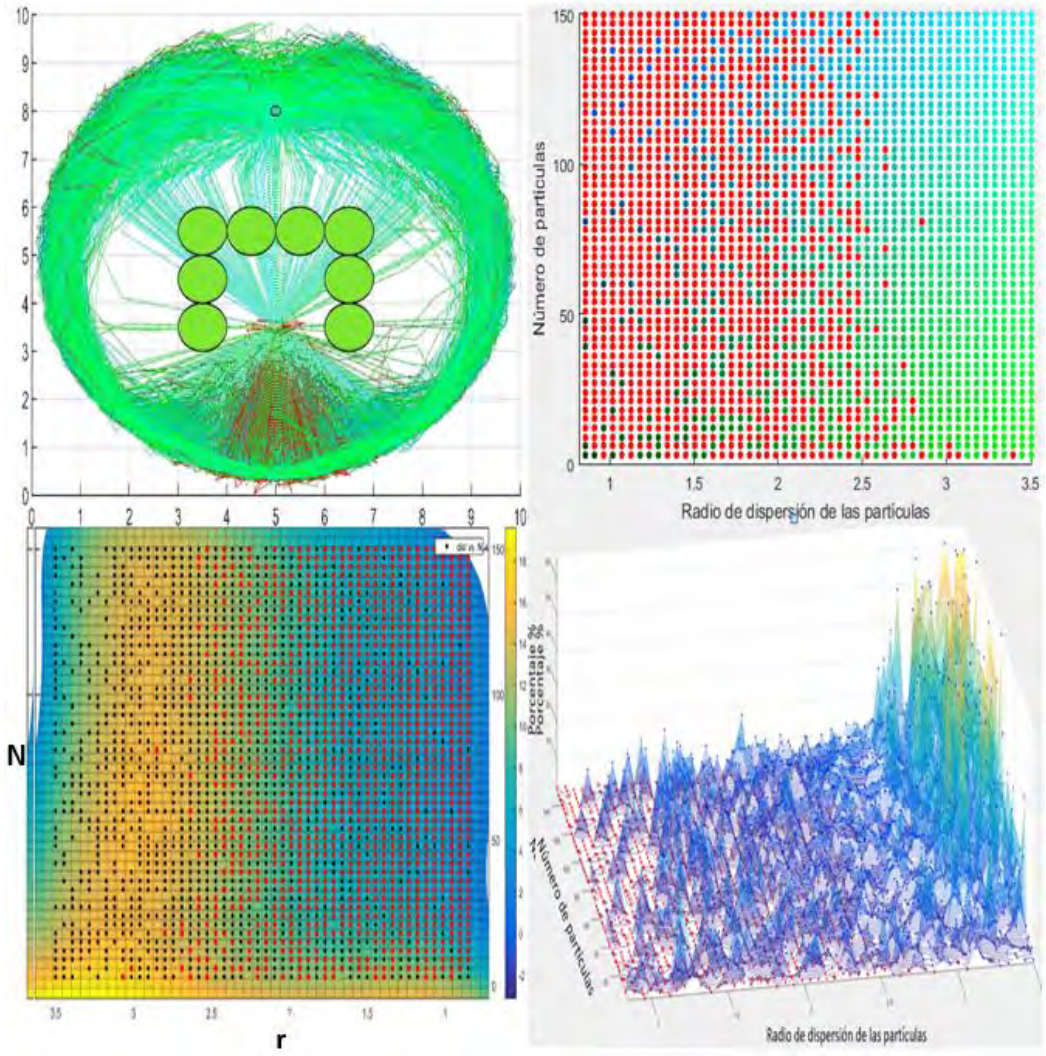
Se observa además que las rutas generadas con la variación del número de partículas *n* y el radio de dispersión de las mismas *r* en el escenario tipo trampa. Las rutas que no logran conectar el inicio y la meta, se identifican con color rojo, aquí no se ha identificado con este mismo color a las rutas que se crean entre los obstáculos.

La cantidad de aciertos aumenta conforme *r* crece y basta con que $n > 15$ para aumentar a posibilidad de que se generen rutas acertadas, en este caso. En tanto a los caminos creados entre los obstáculos, se puede observar que estos son identificados con un color azul verdoso, que corresponde a las regiones cercanas al límite $r = 3.5$ y $n = 150$.

Por otra parte, en la Figura 55 inferior izquierda se muestra la longitud de las rutas generadas con la variación del Número de partículas *n* y el radio de dispersión de las mismas *r* en el escenario tipo trampa. Se observa como las distancias aumentan cuando el valor de *r* se acerca de 0 a 3.5 y disminuye nuevamente después de este valor. Por otro lado, cuando el número de partículas disminuye, la distancia aumenta, en sentido contrario se podría inferir que el aumentar el número de partículas disminuye la distancia de la ruta, esto último solo ocurre significativamente cuando $n < 20$.

En Figura 55 inferior derecha se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las “cercanías del obstáculo” con la variación del Número de partículas *n* y el radio de dispersión de las mismas *r* en el escenario tipo trampa. Se puede observar que el crecimiento del radio de dispersión, provoca considerablemente la generación de las rutas en las cercanías de los obstáculos, hasta el punto de generarse rutas entre ellos. El número de partículas también influye en la cercanía (en el límite de los rangos evaluados) así se observa que para valores de *r* cercanos a 3.5, el porcentaje de la longitud total de la ruta aumenta, conforme *n* aumenta.

Figura 55. Rutas generadas para variaciones de número de partículas y radio de dispersión. Escenario trampa PSO



Fuente: Esta investigación.

Exploración rápida de árboles aleatorios

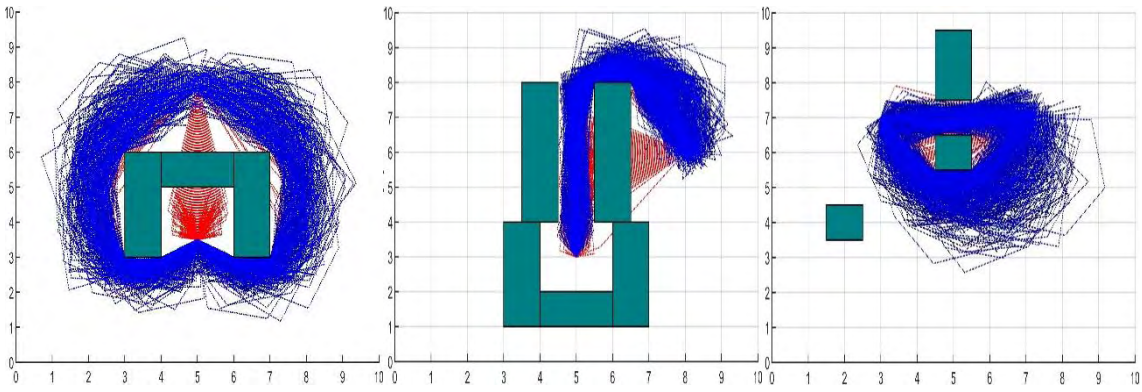
a) Eficacia de la solución encontrada

Variación del número de nodos

En la Figura 56 se muestran las rutas generadas con la variación del número de nodos Nod_{max} en los tres escenarios. La Tabla 21 muestra los valores de Nod_{max}

para los que empiezan a existir aciertos y el valor en el que dejan de existir desaciertos, en cada uno de los escenarios.

Figura 56. Rutas variando el número de nodos



Fuente: Esta investigación.

Tabla 21. Valores de $\boldsymbol{Nod_{max}}$ que determinan aciertos y fracasos en la planeación

	Valores de $\boldsymbol{Nod_{max}}$ para los que empiezan a existir aciertos	**Valores de $\boldsymbol{Nod_{max}}$ para los que dejan de existir los desaciertos**
Trampa	$Nod_{max} \geq 46.3$	$Nod_{max} \geq 166.9$
Pasaje	$Nod_{max} \geq 112$	$Nod_{max} \geq 439$
Arreglo	$Nod_{max} \geq 5$	$Nod_{max} \geq 56$

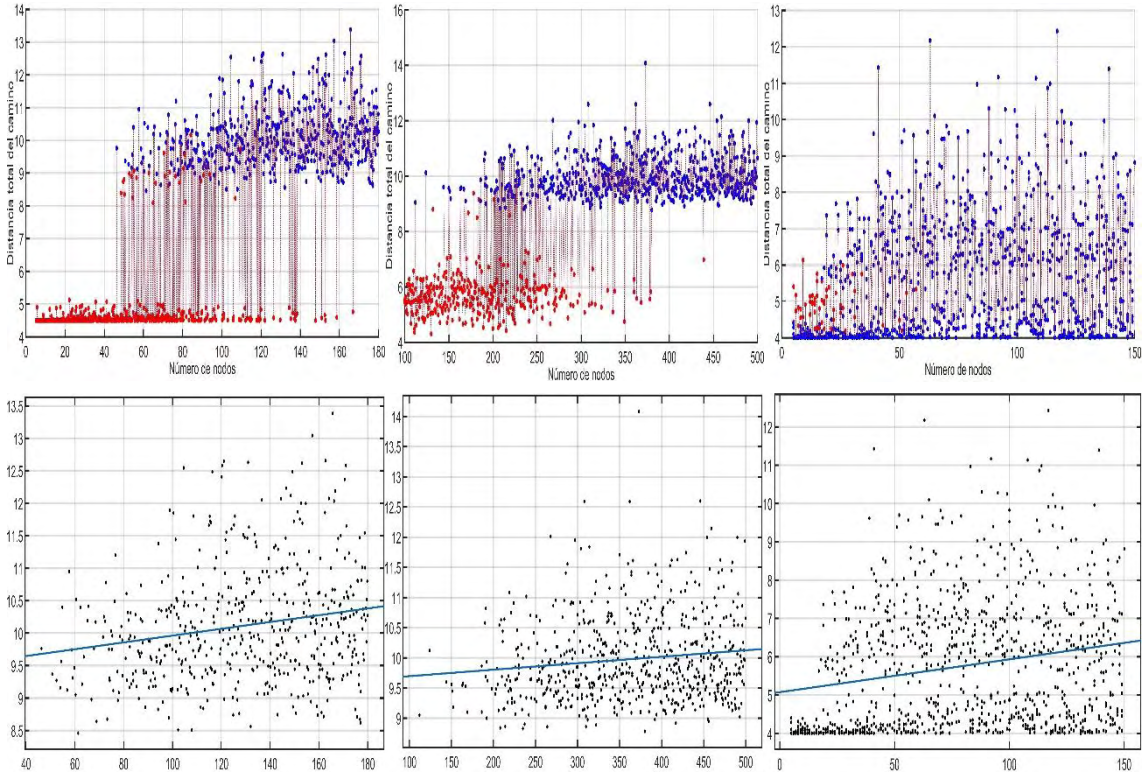
b) Longitud total de la ruta

Variación del número de nodos

Trampa

En la Figura 57 superior izquierda se muestra la longitud de las rutas generadas con la variación del parámetro Nod_{max} en el escenario trampa.

Figura 57. Longitud total del camino dependiendo de Nod_{max}. RRT*



Fuente: Esta investigación.

La Figura 57 inferior izquierda muestra la regresión polinomial de grado 1 para los datos de distancia. En la Tabla 22 se observan los resultados de las variables de ajuste y las longitudes mínima para $Nod_{max} = 46$ y máxima para $Nod_{max} = 180$. Aquí se logra apreciar, que aunque los resultados de la regresión sugieren que el aumento del número de nodos genera distancias más largas, la cercanía del valor de $\boldsymbol{R^2}$ a 0 implica que no existe una buena relación entre la longitud y Nod_{max}.

Tabla 22. Variables de ajuste regresión polinomial para función escenario trampa en RRT*.

SSE	$\boldsymbol{R^2}$	RMSE	Longitud mínima	Longitud máxima
385.1	0.0384	0.8548	9.68	10.39

Pasaje

En la Figura 57 superior centro se muestra la longitud de las rutas generadas con la variación del parámetro Nod_{max} en el escenario pasaje.

La Figura 57 inferior centro muestra la regresión polinomial de grado 1 para los datos de distancia. En la tabla Tabla 23 se observan los resultados de las variables de ajuste y las longitudes mínima para $Nod_{max} = 112$ y máxima para $Nod_{max} = 500$. Dado que el valor de R^2 se acerca a 0, no existe una buena relación entre la longitud y Nod_{max}.

Tabla 23. Variables de ajuste regresión polinomial para función escenario pasaje estrecho en RRT*.

SSE	$\boldsymbol{R^2}$	**RMSE**	**Longitud mínima**	**Longitud máxima**
313	0.01731	0.6971	9.709	10.12

Arreglo

En la Figura 57 superior derecha se muestra la longitud de las rutas generadas con la variación del parámetro Nod_{max} en el escenario arreglo.

La Figura 57 inferior derecha muestra la regresión polinomial de grado 1 para los datos de distancia. En la tabla Tabla 24 se observan los resultados de las variables de ajuste y las longitudes mínima para $Nod_{max} = 5$ y máxima para $Nod_{max} = 150$. Dado que el valor de R^2 se acerca a 0, no existe una buena relación entre la longitud y Nod_{max}.

Tabla 24. Variables de ajuste regresión polinomial para función escenario arreglo de obstáculos en RRT*.

SSE	$\boldsymbol{R^2}$	RMSE	Longitud mínima	Longitud máxima
2479	0.0435	1.626	5.118	6.36

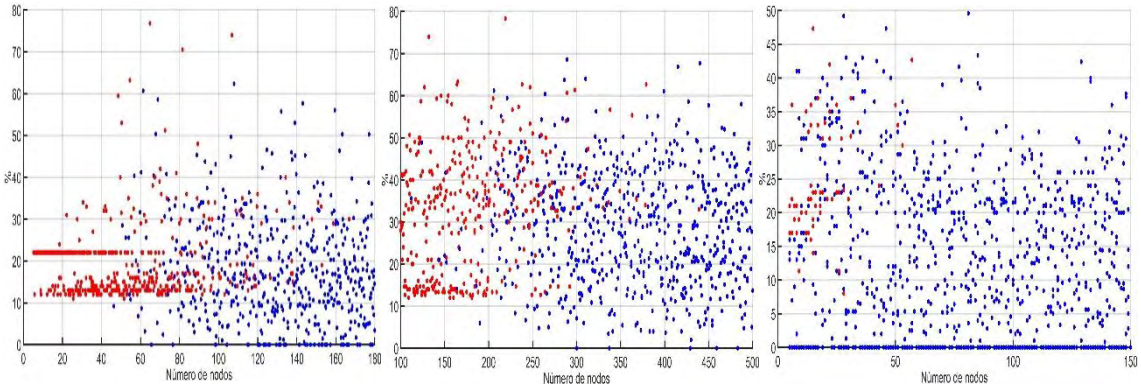
c) Seguridad de la ruta

Variación del número de nodos

Trampa

En la Figura 58 izquierda se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las “cercanías del obstáculo” con la variación del número de nodos Nod_{max} en el escenario trampa. Para las rutas acertadas, la Tabla 25 muestra la media de los porcentajes de longitud de ruta, el porcentaje máximo, mínimo y la desviación estándar.

Figura 58. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. RRT*.



Fuente: Esta investigación.

Tabla 25. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. RRT* escenario trampa.

Media %	$\boldsymbol{Desviación\ estandar}$	Porcentaje mínimo%	Porcentaje máximo%
17.6468	12.2498	0	62.33

Además para $5 < k \leq 180$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 52.9%. Aquí se toman tanto los aciertos, como los desaciertos. La gran dispersión de los datos demuestra que no existe una dependencia de este porcentaje y el número de nodos.

Pasaje

En la Figura 58 centro se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las “cercanías del obstáculo” con la variación del número de nodos Nod_{max} en el escenario pasaje. Para las rutas acertadas, la Tabla 26 muestra la media de los porcentajes de longitud de ruta, el porcentaje máximo, mínimo y la desviación estándar.

Tabla 26. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. RRT* escenario pasaje estrecho.

Media %	$\boldsymbol{Desviación\ estandar}$	Porcentaje mínimo%	Porcentaje máximo%
28.6278	13.37	0	68.5714

Además para $100 < k \leq 500$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 64.6%. Aquí se toman tanto los aciertos, como los desaciertos. La gran dispersión de los datos demuestra que no existe una dependencia de este porcentaje y el número de nodos.

Arreglo

En la Figura 58 derecha se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las "cercanías del obstáculo" con la variación del número de nodos Nod_{max} en el escenario arreglo. Para las rutas acertadas, la Tabla 27 muestra la media de los porcentajes de longitud de ruta, el porcentaje máximo, mínimo y la desviación estándar.

Tabla 27. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. RRT* escenario arreglo de obstáculos.

Media %	$Desviación\ estandar$	Porcentaje mínimo%	Porcentaje máximo%
12.638	11.7517	0	49.6

Además para $5 < k \leq 150$, el porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos es del 94%. Aquí se toman tanto los aciertos, como los desaciertos. La gran dispersión de los datos demuestra que no existe una dependencia de este porcentaje y el número de nodos.

Variación de parámetros del algoritmo

En la Figura 59 superior izquierda y derecha se muestran las rutas generadas con la variación del Avance *EPS* y el radio de búsqueda *r* en el escenario tipo pasaje. Las rutas que se crean entre los obstáculos, son identificadas con color rojo.

Aquí, La cantidad de aciertos parece depender exclusivamente de *EPS*. Los aciertos comienzan a existir para los $EPS \geq 0.35$ y los desaciertos dejan de existir en su mayoría para los $EPS \leq 0.55$.

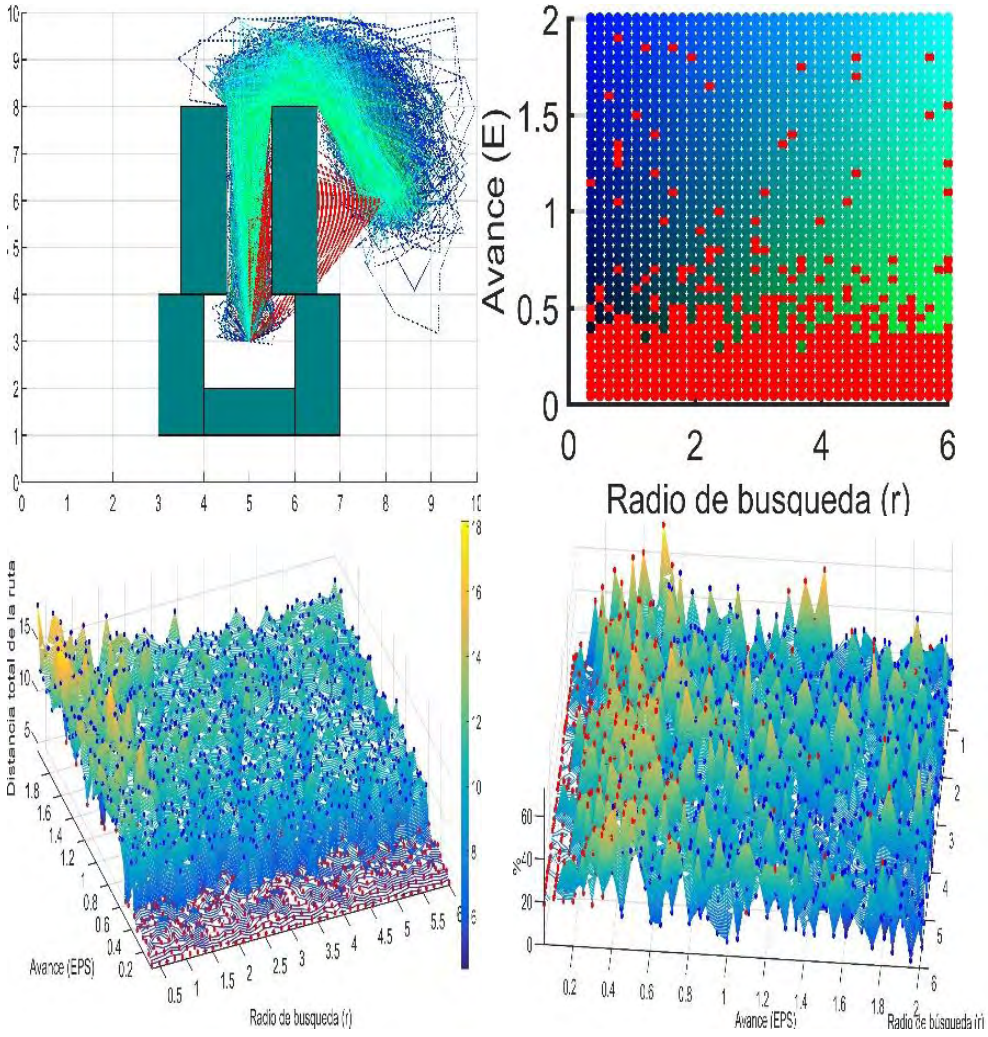
En la Figura 59 inferior izquierda se muestran las rutas generadas con la variación del Avance *EPS* y el radio de búsqueda *r* en el escenario tipo pasaje. Las rutas que se crean entre los obstáculos, son identificadas con color rojo.

Se puede observar como la longitud aumenta al crecer EPS y al disminuir *r*. Dónde la longitud menor 8.8256m se da cuando $EPS = 0.4$ y $r = 5.711$ y la mayor 182.933m cuando $EPS = 2$ y $r = 0.3450$.

En la Figura 59 inferior derecha se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las "cercanías del obstáculo" con la variación del Número de partículas *n* y el radio de dispersión de las mismas en el escenario tipo

trampa. Se puede observar que el crecimiento de *EPS*, disminuye considerablemente la generación de las rutas en las cercanías de los obstáculos. El radio *r* no parece ser muy influyente en este ítem.

Figura 59. Variación de parámetros caso particular RRT*.



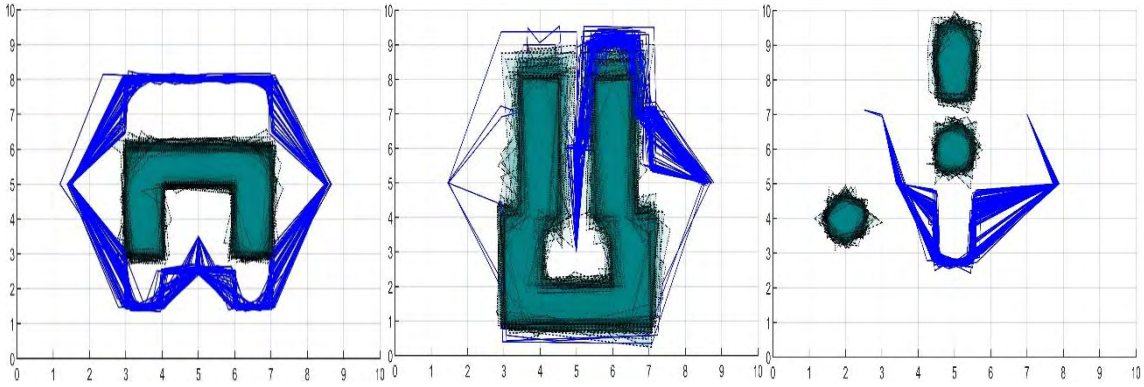
Fuente: Esta investigación.

Descomposición por celdas trapezoidales

a) Eficacia de la solución encontrada

En la Figura 60 se muestran las rutas generadas con la variación de la forma de los obstáculos, en los tres escenarios. La naturaleza del algoritmo CD le permite encontrar una solución, siempre que exista una y que el espacio de trabajo se defina correctamente.

Figura 60. Rutas variando la forma de los obstáculos en CD.

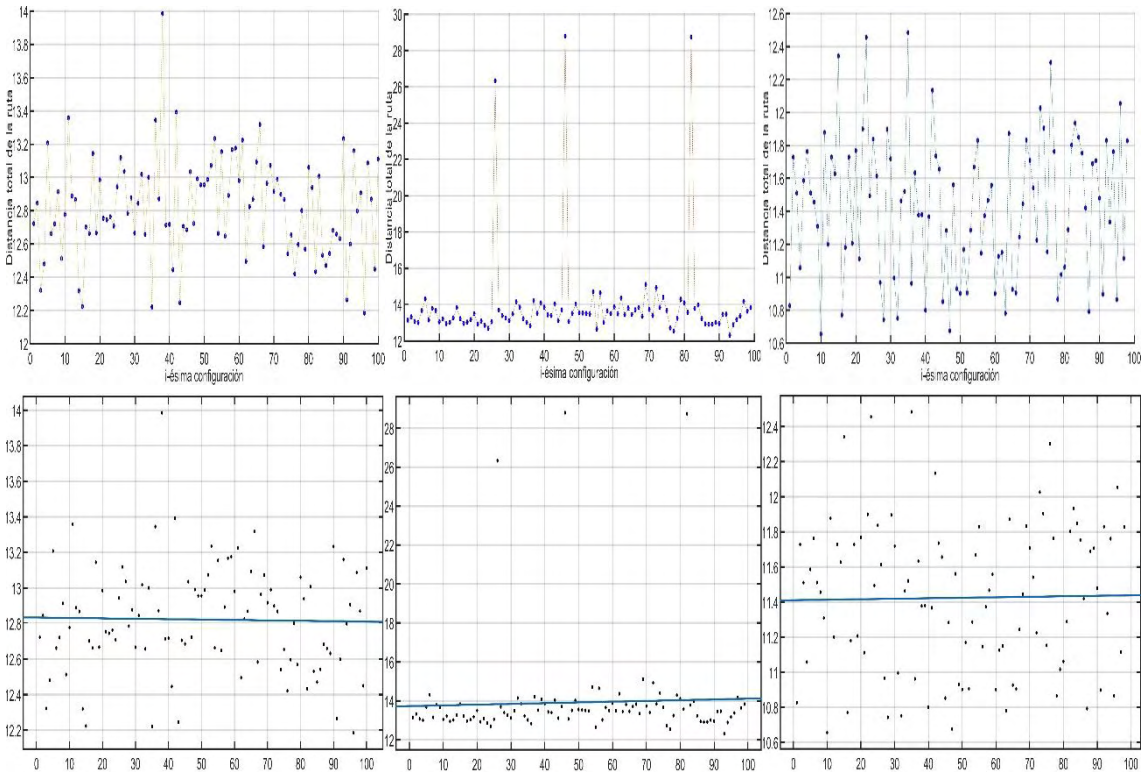


Fuente: Esta investigación.

b) Longitud total de la ruta

En la Figura 61 superior se muestra la longitud de las rutas generadas con la variación de la forma de los obstáculos, en los tres escenarios.

Figura 61. Longitud de ruta para variaciones de forma de los obstáculos. CD.



Fuente: Esta investigación.

La Figura 61 inferior se muestra la regresión polinomial para los datos de distancia en los tres escenarios. En la tabla Tabla 28 se observan los resultados de las variables de ajuste, las longitudes mínimas y máximas, para un grado específico, en cada escenario.

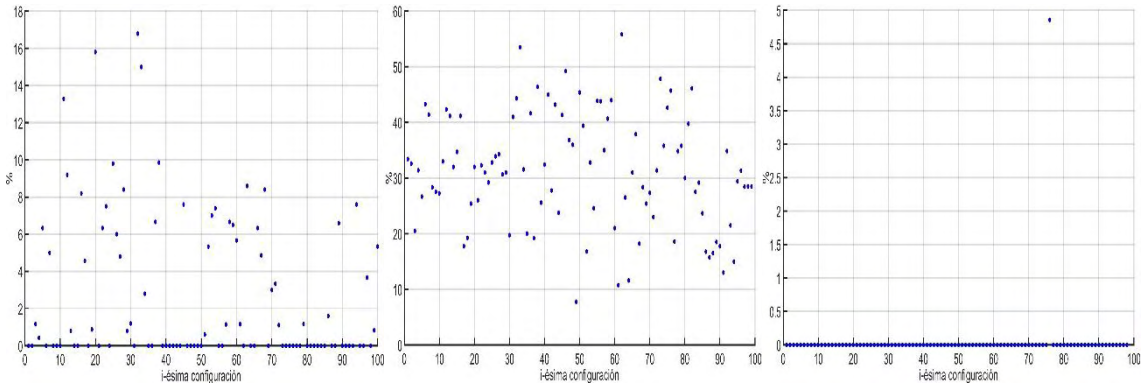
Tabla 28. Variables de ajuste regresión lineal en CD.

	Grado	SSE	R^2	RMSE	Longitud mínima	Longitud máxima
Trampa	1	8.867	0	0.3008	12.81	12.83
Pasaje	1	638.3	0	2.565	13.73	14.11
Arreglo	1	17.97	0	0.43	11.41	11.44

c) Seguridad de la ruta

En la Figura 62 se muestra el porcentaje de la longitud total de las rutas generadas, que se crea en las “cercanías del obstáculo” con la variación de la configuración de los obstáculos en todos los escenarios. Para las rutas acertadas, la Tabla 29 muestra la media de los porcentajes de longitud de ruta, el porcentaje máximo, mínimo y la desviación estándar, además del porcentaje de rutas generadas que tienen porciones creadas en las cercanías de los obstáculos.

Figura 62. Porcentaje de rutas cuyos tramos se encuentran en las cercanías de los obstáculos. CD.



Fuente: Esta investigación.

Tabla 29. Media de porcentajes de longitud de ruta, σ y porcentaje mínimo y máximo. CD.

	Media %	$\boldsymbol{Desviación\ estandar}$	Porcentaje mínimo%	Porcentaje máximo%	Porcentaje de rutas creadas en las cercanías…
Trampa	5.6902	4.0811	0	16.8	48
Pasaje	31.1745	10.1379	7.75	55.833	100
Arreglo	4.8571	0	0	4.8571	1.02

5.1.2. ESTUDIO DE COMPARACIÓN

Con los resultados obtenidos anteriormente se analiza cada método en torno a los criterios de comparación establecidos, se referirá a ellos jerárquicamente, iniciando con los métodos que mejores resultados presentan en cada uno de los índices y finalizando con los que resultan peores para el análisis.

a) Eficacia

Como se dijo con anterioridad, la naturaleza del método CD le permite encontrar una solución al problema de planificación de trayectorias siempre que exista una. Esto depende, únicamente, de qué tan bien se sigan las normas con las que el espacio de trabajo es definido, por ejemplo, para grafos en posición general se requiere que la coordenada x de los vértices no se repita, y que las aristas no se crucen.

En cuanto al Método RRT*, se vio que basta con elegir una buena configuración de los parámetros *EPS* y *r* y un número suficiente de nodos, para que exista una solución en cualquier tipo de escenario, siempre y cuando esta exista.

Para el método de campos potenciales, se puedo apreciar la dependencia de la solución a los parámetros que definen el espacio de trabajo, para los dos tipos de funciones, locales y globales, siendo estas últimas, con las que se generan las trayectorias acertadas para la mayoría de los escenarios. Nótese que, con la función de campo local, con ninguna de las técnicas (Descenso de gradiente y PSO) se logra una ruta acertada para el escenario tipo trampa, y que, en el caso de la técnica de gradiente descendente, solo se logran soluciones para el escenario de arreglo de obstáculos.

Se logró apreciar las ventajas de la técnica PSO, sobre la de Gradiente descendente. En la función de navegación, para los escenarios de arreglo de obstáculos y trampa, el rango de la variable *k* es más grande para PSO que para Gradiente descendente, esto se debe a que la velocidad del móvil no está limitada por la forma de la función, en el método de gradiente descendente este problema

hace que el número de iteraciones, para poder encontrar una solución, crezca exponencialmente Sección 5.1.1 para gradiente descendiente y PSO. En el escenario pasaje se puede observar como la exploración de las partículas en el círculo de dispersión logra que algunas rutas superen los mínimos locales, en los que se detienen las rutas generadas por la técnica de Gradiente descendente.

Por último, para la función de campo local, se logró apreciar como PSO logra encontrar más rutas acertadas en los rangos de dispersión, para el escenario de arreglo de obstáculos y encontrar algunas soluciones, que no puedo encontrar la técnica de gradiente descendente, en el escenario trampa.

b) Longitud total de la ruta

En la Tabla 30 se resume las tablas de la Sección 5.1.1. Aquí se hace énfasis en la Longitud mínima de la ruta encontrada en cada una de las regresiones y el *RMSE* para identificar la ruta con mejores resultados en términos de longitud.

Tabla 30. Resumen valores de regresión polinomial de las muestras.

							Rango con 2 RMSE 95%	
	ESCENARIO	**SSE**	R^2	**RMSE**	**Longitud mínima**	**Límite inferior**	**Límite superior**	**2*RMS**
Campos	**Trampa**	0.7739	0.9625	0,2199	19,47	19,03	19,9098	0,4398
	Pasaje	-	-	-	-	-	-	-
	Arreglo	0,0934 0,2363	0,9964 0,72118	0,03136 0,03226	3,952	3,889	4,01472	0,06272
PSO	**Trampa**	57,83	0,7037	0,4212	14,74	13,9	15,5824	0,8424
	Pasaje	19,1	0,4163	0,4741	10	8,649	10,5452	0,9482
	Arreglo	168,8	0,8035	0,3857	4	3,475	5,0174	0,7714
RRT*	**Trampa**	385,1	0,0384	0,8548	9,68	7,97	11,3896	1,7096
	Pasaje	313	0,01731	0,6971	10	8,315	11,1032	1,3942
	Arreglo	2479	0,0435	2	5	1,866	8,37	3,252
CD	**Trampa**	9	0	0,3008	12,81	12,21	13,4116	0,6016
	Pasaje	638,3	0	3	13,73	8,6	18,86	5,13
	Arreglo	179,7	0	0,43	11,41	10,55	12,27	0,86

En la Tabla 30 se puede observar que con el rango de *2*RMS* (Donde se encuentra alrededor del 95% de los datos) aplicado a la Longitud mínima en cada uno de los métodos, se puede apreciar que RRT* posee el menor límite inferior de la dispersión, en cada uno de los escenarios, siendo 1.866m para Arreglo, 7.97m para Trampa y 8.315m para Pasaje. Esto le permite generar rutas muy cortas, aun cuando la media de distancias no sea siempre la mejor, en este caso solo es mejor para el escenario Trampa. Sin embargo, también es probable que en estos rangos

se generen rutas muy largas, aunque el límite superior de la dispersión no sea excesivamente grande, de hecho, es el mejor en el escenario Trampa con 11.3896m, debido a la enorme dispersión de los datos con un *RMSE* que oscila entre 2 y 0.7. Todo lo anterior se debe a la enorme disposición de los nodos en el espacio de trabajo, que, aunque como se vio, el número de nodos no influye en la media de la longitud, es directamente proporcional a la dispersión de los datos.

PSO es un método que muestra muy buenos resultados en términos de distancia, los límites inferiores de su dispersión están muy cerca de ocupar el segundo lugar en los escenarios de Trampa y Pasaje, ocupando el segundo mejor en el de Arreglo y su límite superior, aún con la gran dispersión de sus datos, es el tercero mejor para Trampa, el mejor para Pasaje y el segundo mejor para arreglo. Lo anterior se debe a la velocidad, en comparación con el método de gradiente descendente, no depende de la dirección de disminución de la función de potencial, la enorme dispersión de las partículas permite encontrar mejores rutas que las señaladas por el gradiente.

Aunque el método de descomposición por celdas tenga dos de los mejores límites inferiores de dispersión, cuyos valores están muy a la par del método PSO, se puede apreciar, que posee una enorme desventaja en términos de longitud en el escenario de arreglos, esto se debe a que el método no es capaz de generar la ruta que los demás algoritmos generan entre dos de los obstáculos Figura 60. En ocasiones este problema se asocia a la definición del espacio de trabajo, los tramos de ruta innecesarios o la ubicación no muy óptima de los nodos del mapa de ruta. Fíjese, por ejemplo, en el escenario de tipo arreglo Figura 60 derecha, si el objeto cuyo centroide se ubica en la coordenada (2,4), allí se crearía un enorme trapezoide, cuyo centro se puede convertir en paso obligatorio, para las rutas generadas.

Por último, el método de Gradiente descendente, resulta particularmente mejor en este índice cuando la función de potencial que relaciona el inicio y la meta, logra conectarlos con suficiente afinidad. Siendo el mejor en términos de las longitud mínima medida en la función de dispersión.

c) Seguridad de la ruta

Los datos correspondientes a cada uno de los escenarios en tanto a este índice reflejan que la técnica que tiende a acercarse más a los obstáculos es la de RRT* debido a su aleatoriedad y destacada dependencia del número máximo de nodos. Esto se discute a continuación para cada escenario: En trampa el algoritmo de gradiente descendiente genera rutas que nunca ingresan al rango de colisión para todo el rango de variación de k, por su parte la optimización por enjambre debido a la aleatoriedad de las muestras, define rutas que se acercan a los obstáculos por debajo de la mitad de las pruebas para un total de 34%. Las rutas creadas por CD se acercan al rango en un 5.69% con una baja dispersión como se ve en la Tabla 29 y en RRT* las rutas de inminente colisión aunque están por debajo de PSO con un 17.65% tienen una alta dispersión alcanzando valores de hasta 62.33% del tramo de ruta que entra en colisión, según Tabla 25.

En pasaje estrecho por gradiente descendiente el porcentaje de rutas dentro del rango es despreciable en comparación a los valores de 49.35%, 28.62% y 31.17% de PSO, RRT* y CD respectivamente. Cabe aclarar que en PSO según Tabla 19 el tope es alto, igual a 61.9% debido a las rutas que no son consideradas como aciertos, es decir son aquellas que pasan por encima de los obstáculos. En CD la desviación es medianamente alta igual al 10.14, sin embargo en RRT* la desviación es alta, luego que los datos son muy dispersos y el tope alcanzado de porcentaje de ruta dentro del rango de colisión es de 68.5%, especialmente para valores pequeños en el número máximo de nodos (caso en el que el algoritmo desprecia la presencia de obstáculos y une el último nodo con el punto de llegada).

Finalmente para el escenario de arreglo, según Tabla 13 el porcentaje de rutas dentro del rango de colisión es bajo al igual que el tope tal y como en PSO y CD donde los valores promedio y la desviación estándar no superan el 10%. Por su parte el algoritmo RRT* supera esta límite con un promedio de porcentajes igual al 12.6% y una desviación del 11.75 según la Tabla 27.

Así el resultado los resultados de las pruebas son desfavorables para técnica de rápida exploración de árboles aleatorios* como consecuencia de su aleatoriedad inherente y necesidad intrínseca de unir el punto final de llegada con el de último nodo definido para completar el grafo de búsqueda. Como contraste, la técnica apropiada para brindar seguridad en el camino, de acuerdo a los resultados obtenidos es PSO que ofrece los mejores resultados en los tres casos de estudio con la desventaja de no encontrar la solución para una cantidad de pruebas realizadas. Para evitar este inconveniente es importante definir muy bien la función a optimizar para evitar que las partículas salten por encima de los obstáculos o se dispersen por valles extensos.

Como un acercamiento al estudio de la suavidad de las rutas se propone este subíndice, en relación a las tablas presentes en la Sección 5.1.1 para el índice de longitud de la ruta específicamente a los valores de dispersión de la ruta.

Nótese que en estas tablas, el método con menores valores asignados al *RMSE*, es el método de Gradiente descendente. Esto significa, que las rutas generadas por este método, respecto a la variación de los parámetros, son muy uniformes. Esto, por supuesto, no significa que las curvas generadas, sean suaves, pero nos da índices de ellos. En este caso las curvas generadas por el método de gradiente descendente, si son las más suaves, puesto que son las únicas que garantizan la existencia, al menos, de la primera derivada.

Las segundas mejores, respecto al *RMSE,* son las curvas generadas por CD, esto se debe a que las ubicaciones de los nodos en el mapa de ruta no tienen una gran variación con el cambio de la forma de los objetos.

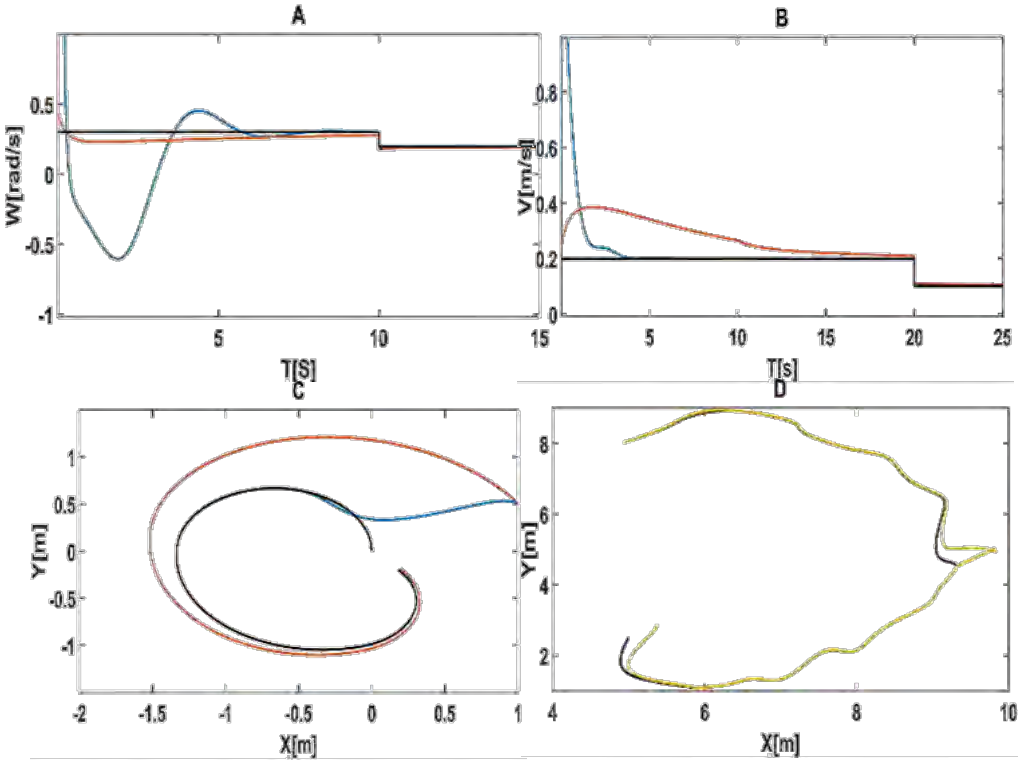
Por último, los métodos basados en distribuciones aleatorias, como RRT y PSO, siempre serán los que mayor dispersión presenten en la recolección de datos.

5.2. APROXIMACIÓN AL TRABAJO FUTURO

5.2.1. CONTROL DE MOVIMIENTOS.

La Figura 63 muestra el comportamiento del control frente a distintas entradas con el uso de constantes arbitrarias (línea naranja) y bajo la técnica de escalado de velocidad (línea azul) para el seguimiento de una referencia (línea negra). Las gráficas superiores corresponden a entrada escalón unitario y una perturbación, y las gráficas inferiores a una entrada de trayectoria de referencia.

Figura 63. Resultados de control aplicado al robot de tracción diferencial.



En negro se muestran las referencias deseadas, en azul los resultados para la técnica de “escalado de velocidad”, en naranja la respuesta una sintonización con constantes arbitrarias y en amarillo la trayectoria obtenida por el control para un camino generado por la técnica PSO. **Fuente:** Esta investigación.

Como se observa el control cumple con la función de estabilizar el vehículo sobre la trayectoria de referencia bien sea con constantes arbitrarias o aplicando la técnica de escalado de velocidad, no obstante cabe mencionar que las oscilaciones y la

disminución del tiempo de establecimiento que ofrece la técnica de escalado de velocidad, son importantes para un seguimiento mucho más efectivo de la ruta.

En la Figura 63 se muestran los resultados de simulación para el control del vehículo utilizando la técnica de control descrita en la Sección 2.1.8. En la gráfica superior izquierda se tiene la respuesta de la velocidad angular ante un escalón unitario y una perturbación. Se puede observar que el seguimiento de la referencia con constantes arbitrarias (línea naranja), no presenta oscilaciones marcadas en comparación con la técnica de escalado de velocidad (línea azul). Sin embargo, el tiempo de establecimiento del control se incrementa, efecto que es más evidente con la respuesta de velocidad lineal presentada en la gráfica superior derecha.

Las oscilaciones y la reducción del tiempo de establecimiento debido al control exigente de la técnica de escalado de velocidad se manifiestan en un seguimiento mucho más eficiente y preciso de la ruta (Figura 63C), dónde la línea negra representa la trayectoria deseada que lleve al vehículo.

Por último, en la Figura 63D se muestra que es posible aplicar el modelo inverso a una curva continua y suave (al menos en su primera derivada) para obtener trayectorias viables en la aplicación del control propuesto. En línea negra se presenta la trayectoria obtenida después del suavizado con la técnica Spline para un camino elaborado por el método PSO en el arreglo de obstáculos tipo trampa. En línea naranja se presenta la trayectoria obtenida por el control con condiciones iniciales distintas y una perturbación en un tiempo arbitrario. Se nota que el control logra seguir la referencia convenientemente y que se recupera de la perturbación que puede sufrir el vehículo en su recorrido.

Estas condiciones simulan ambientes reales a los que puede estar sometido un robot de tracción diferencial y el comportamiento visible de la respuesta demuestra que partiendo de las trayectorias dadas por los métodos, es posible la implementación de cada técnica en plataformas físicas.

5.2.2. GENERACIÓN DE TRAYECTORIAS

i) Interpoladores B-Spline y Spline

Los resultados de las Figuras 64-66 muestran las trayectorias obtenidas por el método de planeación (color negro) y su respectivo post-procesado, es decir, aplicando los interpoladores Spline (color rojo) y B-Spline (color azul). Se añaden las Tablas 31-33 que presentan el cálculo de la distancia en metros para las trayectorias originales $Dist$ y aquellas generadas $Sdist$ y $BSdist$.

Figura 64. Rutas planeadas en escenario tipo trampa e interpolaciones Spline y BSpline

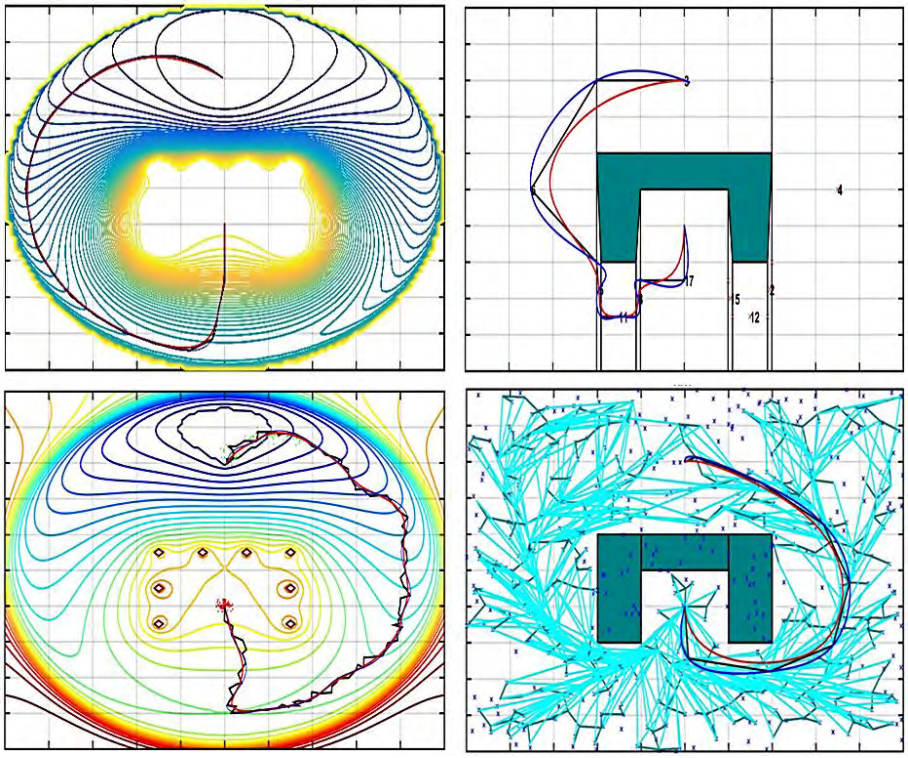


Imagen superior izquierda: Función de navegación campos potenciales artificiales. Imagen superior derecha: Descomposición por celdas trapezoidales. Imagen inferior izquierda: Optimización por enjambre de partículas. Imagen inferior derecha: Exploración rápida de árbol de azar. **Fuente:** Esta investigación.

Tabla 31. Longitud de distancia original y por interpolaciones. Escenario trampa.

MÉTODO	$Dist$	$BDist$	$SDist$
PF	16.93	16.72	17.01
PSO	16.54	16.07	16.72
RRT*	12.21	11.26	12.52
CD	13.66	12.05	14.63

Figura 65. Rutas planeadas en escenario tipo pasaje estrecho e interpolaciones Spline y BSpline

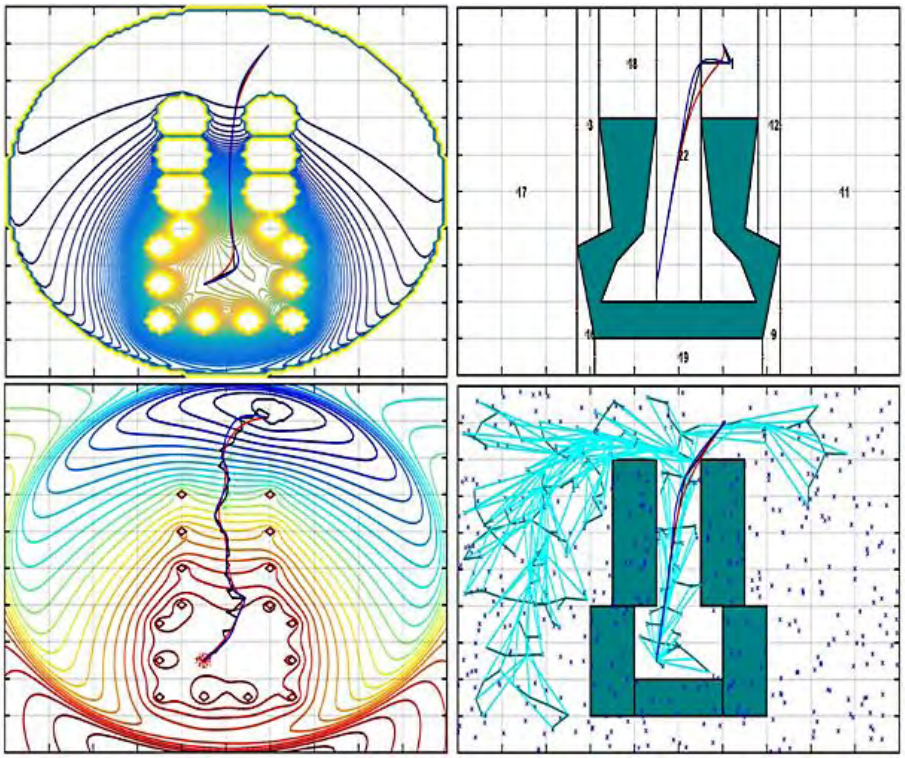


Imagen superior izquierda: Función de navegación campos potenciales artificiales. Imagen superior derecha: Descomposición por celdas trapezoidales. Imagen inferior izquierda: Optimización por enjambre de partículas. Imagen inferior derecha: Exploración rápida de árbol de azar. **Fuente:** Esta investigación.

Tabla 32. Longitud de distancia original y por interpolaciones. Escenario pasaje estrecho.

MÉTODO	$Dist$	$BDist$	$SDist$
PF	7.01	6.87	7.06
PSO	7.81	7.46	7.92
RRT*	6.77	6.74	6.92
CD	7.25	6.74	7.33

Figura 66. Rutas planeadas en escenario tipo arreglo de obstáculos e interpolaciones Spline y BSpline.

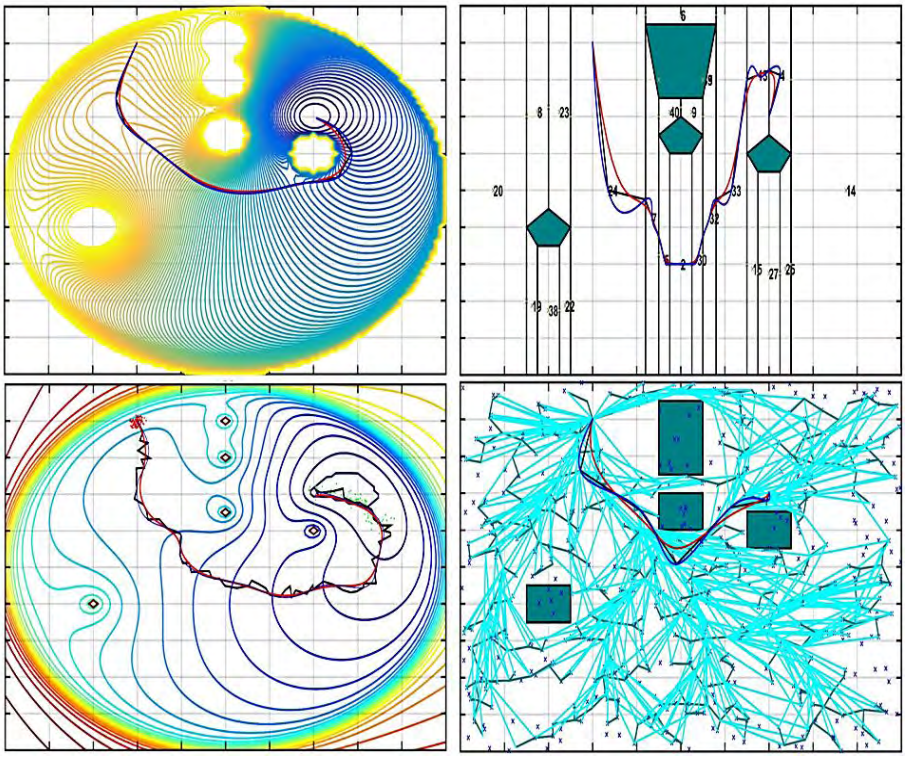


Imagen superior izquierda: Función de navegación campos potenciales artificiales. Imagen superior derecha: Descomposición por celdas trapezoidales. Imagen inferior izquierda: Optimización por enjambre de partículas. Imagen inferior derecha: Exploración rápida de árbol de azar. **Fuente:** Esta investigación.

Tabla 33. Longitud de distancia original y por interpolaciones. Escenario pasaje estrecho.

MÉTODO	$Dist$	$BDist$	$SDist$
PF	9.88	9.58	9.97
PSO	12.44	11.96	12.64
RRT*	7.84	6.82	8.06
CD	14.55	13.80	16.14

Los interpoladores cumplen la función de suavizar todas las rutas efectivamente. En cuanto a la distancia, el método B-Spline presenta los mejores resultados, puesto que no pasa necesariamente por todos los puntos de control asociados a la curva de referencia. Al contrario, Spline abarca todos los puntos de control asociados a precio de acercarse mucho a los objetos en algunos tramos de curva o generar curvas de amplitud innecesaria, como se puede observar.

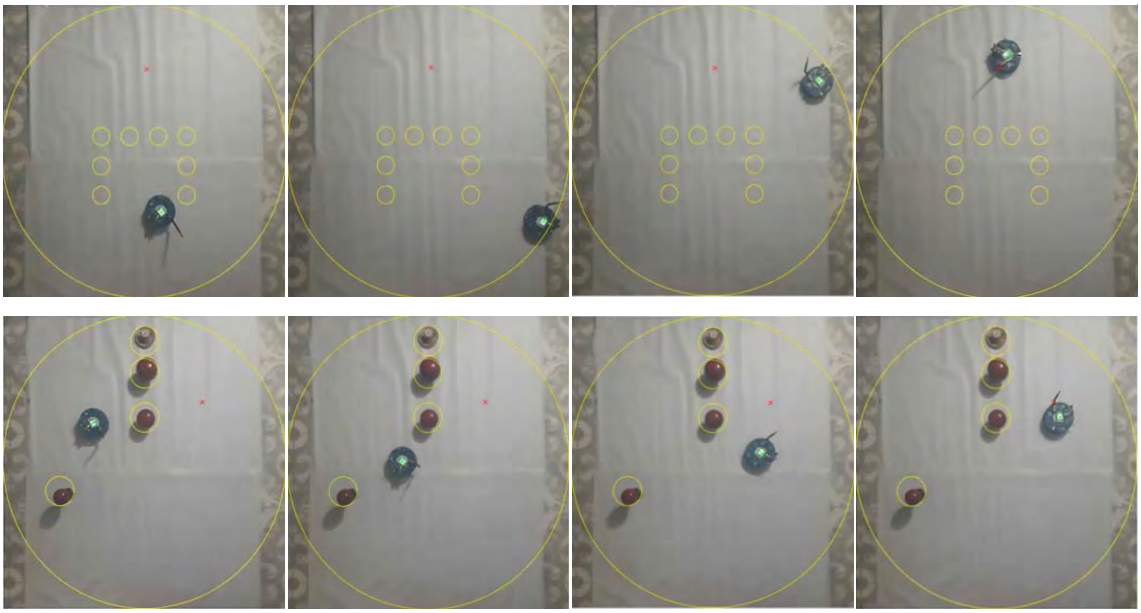
Sin embargo, para efectos de implementación de los algoritmos, B-Spline tiene mayor tiempo de ejecución y menor posibilidad de programación en paralelo, puesto que para la evaluación de la curva paramétrica resultante se debe calcular una sumatoria de todos los puntos de control asociados.

5.2.3. IMPLEMENTACIÓN PLATAFORMA FÍSICA

Como se estudió en la Sección 2.1.8. y según los resultados para la parte de control de movimiento, la implementación de la esta técnica en la plataforma Arduino-Robot se hace retroalimentando la posición del robot a partir del algoritmo propuesto de visión por computador y comunicaciones. La señal de salida es el valor de velocidad angular de cada una de las ruedas que aplicada al modelo (91) envía a los motores los valores de PWM apropiados para lograr aproximar el posicionamiento del vehículo al punto de control de la trayectoria creada.

La Figura 67 muestra la solución que encuentra la plataforma robótica frente al problema de planear una trayectoria óptima y libre de colisión en el escenario trampa y arreglo de obstáculos modificado. La Tabla 34 da a conocer los valores medidos experimentalmente para tiempo y distancia sobre la trayectoria.

Figura 67. Resultados de planeación de trayectoria implementando PF.



El punto color rojo indica la posición objetivo y el punto color cian la posición del robot. Imágenes superiores de izquierda a derecha: Recorrido del robot, escenario trampa. Imágenes inferiores de izquierda a derecha: Recorrido del robot, escenario arreglo de obstáculos. **Fuente:** Esta investigación.

Tabla 34. Índices de comparación implementación.

ESCENARIO	TIEMPO	DISTANCIA
TRAMPA	90	2.37
ARREGLO DE OBSTÁCULOS	130	0.78

Análisis respecto implementación:

Los resultados obtenidos luego de la implementación del algoritmo de planeación basado en campos de potencial artificial, revelan la aplicabilidad de este tipo de técnicas en tiempo real en plataformas físicas, ya que como se observa en la Figura 67, en las dos situaciones implementadas el robot planea su trayectoria y consigue llegar al punto deseado desde una posición de partida. Esto se logra desde una perspectiva descentralizada, ya que el robot de forma automática, calcula el gradiente de su función programada y navega por el entorno, recibiendo únicamente retroalimentación de su posición desde la estación base.

No obstante, cabe resaltar que los resultados de tiempo y distancia de la Tabla 34 son relativos a las condiciones de implementación, es decir, al equipamiento utilizado y la programación realizada. Para este caso en particular, son relevantes los siguientes aspectos: *i)* El tiempo requerido para procesar la imagen y enviar el dato de posición al robot toma de 0.3 a 0.5 segundos, tiempo en el que el robot se detiene, y se moviliza una vez ha sido calculada la señal de control. Este hecho

incrementa el tiempo base que le tomaría al robot desarrollar su navegación, si el tiempo considerado para estos cálculos fuese despreciable. *ii)* Los motores propios de la plataforma realizan arranques inesperados y hacen que las llantas resbalen desviando el robot de su curso. Como consecuencia, se invierte tiempo en la corrección de ruta y las distancias recorridas se incrementan.

En este orden de ideas, es claro que PF y PSO, son viables para implementación en tiempo real bajo la limitante de la definición de una función multidimensional no muy compleja y programable. Entonces se debe considerar que para mejorar tiempo y distancias recorridas se debe trabajar con una alta calidad en los dispositivos, precisión en las mediciones adquiridas y rapidez en el procesamiento embebido y envió de datos.

En cuanto al algoritmo de CD, puede llevarse a la práctica acompañado de un interpolador a fin de generar trayectorias suaves que pueda seguir el robot diferencial. No obstante lo mencionado sobre tiempo y distancia para PF se asocia a CD y a RRT*, quienes requieren un enfoque centralizado. Es decir, se debe realizar en primer lugar el cálculo previo de la trayectoria y su respectivo suavizado, para luego transmitirla al robot paso a paso.

5.2.4. CASO DE MÚLTIPLES ROBOTS – AMBIENTE DINÁMICO

En esta sección se presentan los resultados tras la implementación virtual del método de obstáculos de velocidad recíproca y las dos extensiones propuestas a los algoritmos de campos de potencial artificial y descomposición por celdas. Mediante figuras y tablas se muestran las trayectorias recorridas por cada uno de los robots y el cálculo de distancias recorridas en metros, para las rutas planeadas por el método original *Dist* y aquellas que son generadas por *Spline (SDist)* y *BSpline (BDist).*

La Figura 68 y la Tabla 35 indican los resultados conseguidos considerando los índices de tiempo y distancia que le llevo al equipo de robots en conjunto cumplir con la tarea de navegar por un mismo entorno sin colisión.

i) Plataforma virtual

Tabla 35. Índices de comparación múltiples robots.

Robot	***Dist***	***BDist***	***SDist***
-EXTENSIÓN CAMPOS DE POTENCIAL ARTIFICIAL-			
1	5.84	5.80	5.86
2	9.94	9.89	9.99
3	9.87	9.78	9.92
-EXTENSIÓN DESCOMPOSICIÓN POR CELDAS-			

1	4.39	4.39	6.03
2	10.65	9.69	13.19
3	4.97	4.97	6.71
OBSTÁCULOS DE VELOCIDAD RECÍPROCA			
1	11.92	11.83	11.97
2	5.44	5.42	5.51
3	7.07	7.06	7.12

Figura 68. Rutas planeadas para múltiples robots en un escenario de navegación controlado.

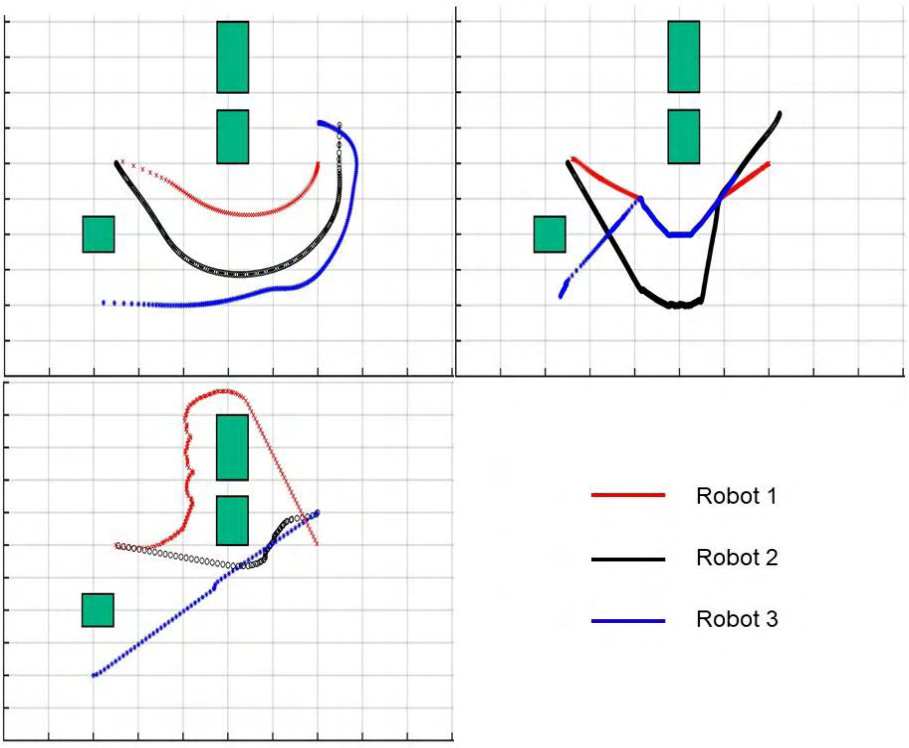


Imagen superior izquierda: Rutas planeadas por la extensión propuesta para campos de potencial artificial. Imagen superior derecha: Rutas planeadas por la extensión propuesta para descomposición por celdas. Imagen inferior izquierda: Rutas planeadas por el método de obstáculos de velocidad recíproca. **Fuente:** Esta investigación.

Análisis respecto índices de comparación múltiples robots.

i) Eficacia de la solución encontrada:

La Figura 68 muestra que la propuesta de extensión de los algoritmos y RVO, cumplen con la función de planear una trayectoria desde un punto de partida a uno de llegada, libre de colisión con los obstáculos estáticos y con los robots presentes

en el entorno. Cabe resaltar que en la extensión para CD y el algoritmo RVO es claro el cruce de rutas por el recurso de espacio compartido. No obstante, este recurso es aprovechado por un solo robot a la vez gracias a las reglas de transito establecidas para CD, y en el caso de RVO, gracias al estudio de tiempo de colisión que evita que dos robots estén presentes en una misma posición o en aquella que involucre el choque de sus armazones.

Estos resultados son favorables debido a las condiciones controladas de desarrollo. Tal como se refirió sobre los resultados de PF para un único robot, para la extensión de campos de potencial artificial se necesita desarrollar una función multidimensional para cada robot, que es una tarea difícil de completar. El no definir correctamente las funciones hará que uno u otro robot no llegue a su objetivo final y posiblemente se estanque en un mínimo local. Para el método CD extendido esto no es un problema, puesto que las ventajas asociadas al desarrollo para un solo robot también lo son para el caso de múltiples, en donde la creación del grafo de conectividad es la misma para todos los robots involucrados.

En tanto a RVO, siempre y cuando el número de robots no sea significativamente elevado (tipo enjambre) y el entorno de navegación no esté limitado a un área determinada, se podrá realizar el cálculo de velocidades óptimas por fuera de los obstáculos de velocidad creados por cada agente. En caso contrario, RVO no converge rápidamente a la solución buscada, debido al enorme espacio ocupado por la densidad de obstáculos de velocidad, que prohíben el desplazamiento y libre movilidad de los vehículos.

ii) Evasión y Ruta establecida:

Para el análisis de múltiples robots navegado en un mismo entorno, según los datos registrados en la Tabla 35 es claro que no es posible determinar un método que planee las rutas más cortas. Si bien la extensión para CD ofrece la ruta más corta para el robot 1, no lo hace para el robot 2, de cuya trayectoria se puede decir que es la menos eficiente. Por este mismo hecho, tampoco es posible afirmar, cuál de las técnicas ofrece las rutas con las peores distancias recorridas.

Lo dicho resulta consecuente con la tarea de evitar colisión. Cada uno de los algoritmos programados busca establecer un camino de conexión entre dos puntos sin colisión con obstáculos estáticos, y al tiempo, reaccionar ante la presencia de un agente móvil. Es aquí precisamente, en donde el criterio de distancia recorrida pierde prioridad en comparación a un choque inminente, e índices como el tiempo, también es degradado.

Por otra parte, en la extensión propuesta para PF y CD, la aplicación de reglas para el transito jerarquizado hacen que un solo robot se mueva rápidamente por el entorno mientras los demás “esperan” su turno correspondiente. En función de esto, se observa en la Figura 68 que se tiende a formar las rutas más óptimas para el robot 1 y 2, puesto que en las rutas programadas por CD, el robot 3 espera a que el

robot 1 pase por el punto (4,5), siendo así, el último en completar la tarea de navegación.

Por su parte, RVO no prioriza el tránsito de uno u otro robot mientras planea la navegación evitando colisión con todos los agentes presentes en el entorno. Para esto tiene en cuenta la suma de los obstáculos de velocidad al momento de permitir el avance en una dirección. Sin embargo, a pesar del uso de la técnica de optimización para el cálculo de velocidades, se observa que la ruta planeada para el robot 1 presenta oscilaciones, y además, por el hecho de planear una ruta sin colisión con otros agentes, evade de una manera no óptima los obstáculos (modelados como agentes con velocidad cero), haciendo que este recorra un trayecto innecesario, pero muy seguro. Por su parte, PF y CD extendido planean una ruta más eficiente para el robot 1, que se caracteriza por poseer las ventajas y desventajas mencionadas para el caso de un único robot.

Además, PF genera rutas suaves que fácilmente pueden ser seguidas por un robot con restricciones no holonómicas y CD une segmentos de línea recta a los que necesariamente debe aplicarse un post-procesado para generación de trayectoria (como los ya expuestos BSline y Spline cuyas distancias no discrepan mucho de las distancias de las rutas originales para cada robot, como se ve en la Tabla 35). En cuanto a RVO, considera las restricciones cinemáticas dentro de sus cálculos. A pesar de ello, las oscilaciones presentes no se pueden obviar, ya que para implementaciones en plataformas físicas, estas fluctuaciones son causantes de deslizamientos y arranques inesperados que alteran el tránsito de ruta e incluso pueden provocar colisiones.

iii) Tiempo de ejecución:

En tanto al tiempo de ejecución, el método RVO presenta las mayores desventajas puesto que el cálculo de la velocidad debe realizarse en cada una de las iteraciones para cada uno de los robots. Por otro lado, la complejidad de los métodos de minimización a los que incurre PSO y ABC, depende del número de individuos o partículas. Combinar estos dos factores, resulta en un método bastante costoso computacionalmente y por ende difícil de implementar en tiempo real.

El enfoque de CD para múltiples robots requiere de un tiempo de computación bastante similar a su caso de único robot puesto que solo cambia el número de nodos creados y sus debidas conexiones. Esto quiere decir que el método sigue siendo centralizado y offline, puesto que requiere de la previa planeación de la ruta. Sin embargo, el método de evasión propuesto, es reactivo y viabiliza mucho más la implementación del método en tiempo real.

Por último, el enfoque PF para múltiples robots es completamente reactivo tanto en la planificación de trayectoria como en la evasión. Así, su complejidad depende únicamente, del tiempo que le tome al algoritmo de gradiente descendente, encontrar una ruta óptima.

6. CONCLUSIONES

Se hizo una comparación de diferentes métodos de planeación de trayectorias con distintas características fundamentales. Al proponer diferentes escenarios y realizar simulaciones de cada uno de los algoritmos, se establecen varios índices que permiten clasificar los métodos y evaluar su desempeño.

Se resaltó que a cada técnica se le asocian ventajas y desventajas, frente a problemas de planeación particulares. Lo más conveniente resulta en determinar las características y condiciones que el problema imponga para asociar estos estudios a las ventajas de las técnicas analizadas.

Haciendo uso de interpoladores se establece una alternativa ante el problema de restricciones no holonómicas en la implementación de las trayectorias, ya que se generan curvas suaves que pueden ser seguidas por controles cinemáticos simples como el presentado.

En tanto a los desarrollos propuestos para múltiples robots, el hecho de establecer reglas de tránsito para el tráfico de robots presentes en un mismo entorno mejora la suavidad de las rutas, evitando oscilaciones. Pese a que se aumenta el tiempo o distancia recorridas, se garantiza el cumplimiento de la tarea de navegación para cada robot.

Como aporte a la comunidad científica se socializó los resultados de esta investigación con un artículo científico en la tercera versión de la conferencia colombiana de automatización y control y se desarrolló una página web que incluye documentos, códigos, vídeos y manuales de usuario con el fin de facilitar y promover los estudios posteriores en robótica móvil.

7. TRABAJO FUTURO

Cabe anotar que aún es tarea de investigación incluir las restricciones no-holonómicas dentro de la formulación de los enfoques de planeación de trayectorias, evitando así la tarea de suavizado que en este trabajo se sustentó a partir de interpoladores.

Por otra parte, es posible establecer una relación matemática entre la planeación global de trayectoria y el control de movimientos de tal manera que se pueda optimizar el comportamiento del sistema de navegación para mejorar la distancia total recorrida, la reacción ante perturbaciones y el tiempo sobre la trayectoria.

En tanto al modelado de escenarios de navegación, la representación de formas complejas de obstáculos reales es un problema inherente a todos los algoritmos de planificación de trayectorias.

Es pertinente aumentar la complejidad de los problemas para métodos en más de dos dimensiones, teniendo en cuenta elevación, tiempo, implementaciones online o que consideren aplicaciones con enjambres de robots.

Para futuras implementaciones en línea, es pertinente el análisis de las técnicas bajo la metodología basada en sensores, adecuado así las plataformas para desarrollos mucho más robustos en robótica cooperativa.

Finalmente el estudio de posicionamiento en el plano de forma automática o asistida es indispensable para evitar correcciones de ruta, por eso aún es tema de investigación el desarrollo de filtros de predicción y técnicas de rastreo que disminuyan la incertidumbre en las medidas.

BIBLIOGRAFÍA

[1] ALLAWI, Ziyad, ABDADLLA, Turki. PSO-Optimized Reciprocal Velocity Obstacles Algorithm for Navigation of Multiple Mobile Robots. In: International Journal of Robotics and Automation. Vol. 4, Nº 1, March 2015. p 31-40.

[2] ALLAWI, Ziyad. ABDADLLA, Turki. An ABC-Optimized Reciprocal Velocity Obstacles Algorithm for Navigation of Multiple Mobile Robots. En: IJCCCE, Mayo 2015. Vol.15, No.2, p 45- 57.

[3] ARDUINO ROBOT. The first Arduino on wheels, the easiest way to get started with robotics. [en línea]. [Citado el 23 de Enero de 2018].

[4] BERG, Mark et al. Algoritmos de Geometría Computacional y Aplicaciones. 3 ed. Berlin: Springer, 2008.3386p

[5] BLACKWELL Mike, The URANUS Mobile Robot, Carnegie Mellon University, 1990 p: Robotics Institute. CMU-RI-TR-91-06.

[6] BREITENMOSER, Andrea's et al. Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots. In: Distributed Autonomous Robotic Systems, Vol 83, 2013; p. 203-216.

[7] BULO, Francesco. CORTÉS, Jorge. MARTINEZ, Sonia. Control Distribuido de Redes Robóticas.1ed. New Jersey: Princenton University Press, 2009, 319p.

[8] CABRERA, Ronald .Planeamiento de trayectoria y control de un robot móvil marítimo aplicando optimización por colonia de hormigas. Lima, 2016, 99 p. Trabajo de grado (Maestría en Control y Automatización). Pontificia Universidad Católica de Perú.

[9] CAPOZZI J. Evolution-based path planning and management for autonomous vehicles. Washington 2001 Tesis Doctoral (Aeronáutica y Astronáutica). University of Washington.

[10] CÁRDENAS, Edwin Francis. A Computational Application of Trajectory Generation and Obstacle Avoidance for Different Manipulative Robots with Analysis in a Real System. Bogotá, 2009, 125p. Degree Work. (Master of Engineering). Universidad Nacional de Colombia. Faculty of Engineering.

[11] CLAES, Daniel et al. Collision Avoidance under bounded Localization Uncertainty, En: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, 2012, p 1192-1198.

[12] CONNOLLY, Christopher. BURNS Julian. Path planning using Laplace's equation, IEEE Conference on Robotics and Automation, 1994. Vol.3. p. 2102-2106

[13] DE LOS SANTOS DE LA ROSA Erik, Heurística para la generación de configuraciones en pasajes estrechos aplicada al problema de los clavos. Puebla, 2004. Trabajo de grado. (Maestría en Ciencias con Especialidad en Ingeniería en Sistemas Computacionales) Universidad de las Américas

[14] ESPITIA, Helbert. Propuesta de un algoritmo para la planeación de trayectorias de robots móviles empleando campos potenciales y enjambres de partículas activas brownianas. Bogotá, 2011,137 p. Trabajo de grado (Maestría en Ingeniería-Ingeniería Mecánica) Universidad Nacional de Colombia. Facultad de Ingeniería.

[15] FIORINI, Paolo. SHILLER, Zvi. Motion Planning in Dynamic Environments Using Velocity Obstacles. En: International Journal of Robotic Research, Vol 17, Nº 7, Julio 1998; p 760-772.

[16] GARRIDO, Santiago et al. General Path Planning Methodology for Leader-Follower Robot Formations. En: International Journal of Advanced Robotic System. Enero, 2013. Vol 10 Nº 1, p 1-10.

[17] GE, Sam. CUI, Yu Jun. Dynamic Motion Planning for mobile robots using potential field method, En: Autonomous Robots. Noviembre 2002, Vol 13, Nº3, p 207–222.

[18] GE, Sam. CUI, Yu Jun. New potential functions for mobile robot path planning, IEEE Transactions on Robotics and Automation, Oct 2000. Vol. 16, Nº 5, p. 615-620.

[19] GEMEINDER, March. GERKE, Markus. GA-based path planning for mobile robot systems employing an active search algorithm, Applied Soft Computing, Junio 2003. Vol. 3, Nº 2, p. 149–158.

[20] GUEVARA, Brayan Ricardo .Generación de trayectos en un entorno dinámico usando una plataforma robótica diferencial, Bogotá, 2017, 99p. Trabajo de grado. (Ingeniería electrónica), Fundación Universitaria los Libertadores, Facultad de ingeniería.

[21] GUY, Stephen et al. Reciprocal n-body Collision Avoidance. En 14th International Symposium of Robotic Research, ISRR 2009 - Lucerne, Switzerland. Junio 2011 Vol 70, Nº3, p 3-19.

[22] GUZMAN, Jaime; ARANGO, Rafael; JIMÉNEZ, Leidy. Búsqueda de la ruta óptima mediante los algoritmos: genético y dijkstra utilizando mapas de

visibilidad. En: Scientia et Technica, Agosto, 2012. Vol. Nº 5, p. 107-112, 2012.

[23] HOWARD, Andrew. MATARIE, Maja. SUKHATME, Gaurav. Mobile Sensor Network Deployment using Potential Field a Distributed, Scalable Solution to the Area Coverage Problem. En: Submitted to the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02) Fukuoka Japan, 2002. 11p.

[24] JANARDANAN, Jayasri. Evitación de Colisión Descentralizada, Universidad de Nebraska-Lincoln, Lincon, Nebraska, 2013, 85 p. Trabajo de grado. (Maestría en Ciencia). Universidad de Nebraska – LincoIn. Departamento de Ciencia Computacional.

[25] JIMÉNEZ, Jovani, VALLEJO, Marcela. OCHOA, John. Methodology for the Analysis and Design of Multi-Agent Robotic Systems: MAD-Smart. In: Journal Advances in Systems and Computing. Sep 2007.Vol4, Nº 2, p 61-70.

[26] JOHANSSON, Ronnie .Intelligent Motion Planning for a Multi-Robot System. 1 ed. Stockholm: Royal Institute of Technology, 2001, 83 p.

[27] JOHNSONBAUGH, Richard. Matemáticas Discretas. 6 ed. Mexico: Person, Capítulo 8, p 347-352.

[28] JUR VAN, Berg, LIN, Ming. DINESH Manocha. Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation. En: IEEE International Conference on Robotics and Automation, Pasadena, CA, 2008, p 1928-1935.

[29] JUR VAN, Berg. WILKIE, David. MANOCHA, Dinesh. Generalized Velocity Obstacles. En: IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, 2009, p. 5573-5578.

[30] KARAMAN, Sertac. FRAZZOLI, Emilio. Incremental sampling-based algorithms for optimal motion planning, En: Robotics: Science and Systems (RSS), 2010.p. 3-8, 2010.

[31] KARAMAN, Sertac. FRAZZOLI, Emilio. Sampling-based algorithms for optimal motion planning, En: International Journal of Robotics Research. Sept 2011.Vol 30, Nº 7, p. 846–894,

[32] KAVRAKI Lydia, LATOMBE Jean, OVERMARS Marc. Probabilistic roadmap for path planning in high dimensional configuration spaces. IEEE Transactions on Robotics and Automation, Aug 1996, Vol. 12, Nº 4, p. 566-580.

[33] KENNEDY, James. EBERHART, Russell. SHI, Yuhui The Particle Swarm. En: Swarm Intelligence, San Francisco, CA: Morgan Kaufmann Publishers. 2001. Ch.7, pp. 287-325.

[34] KHOSLA, Pradeep. Real-time obstacle avoidance using harmonic potential functions, IEEE Conference on Robotics and Automation, Sacramento, CA, Vol. 8, No. 3, Jun 1992; p. 338-349,

[35] KHOSLA, Pradeep. VOLPE, Ricardo. Super quadric artificial potentials for obstacle avoidance and approach, IEEE International Conference on Robotics and Automation, Philadelphia, PA, 1988. Vol 3; p 1778-1784.

[36] LEE Leng-Feng, Decentralized motion planning within an artificial potential framework (APF) for cooperative payload transport by multi-robot collectives. New York, 2004, 194p. Degree Work, (Master of Science). University of New York, Department of Mechanical and Aerospace Engineering.

[37] LINDEMANN, Stephen, VALLE, Steven. Steps toward derandomizing RRTs. In: IEEE Fourth International Workshop on Robot Motion and Control (Puszczykowo, Poland), 2004; p. 271-277.

[38] LIU, Zheng; ANG, Marcelo; SEAH, Winston. A potential field based approach for multi-robot tracking of multiple moving targets. In: The First Humanoid, Nanotechnology, Information Technology, Communication and Control. Environment and Management. International Conference March 27-30, 2003, Manila, Philippines. 8 p.

[39] LÓPEZ GARCÍA, Diego. Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos.Huelva, 2012, 294 p. Tesis doctoral. Universidad de Huelva. Departamento de Ingeniería Electrónica, de Sistemas Informáticos y Automática.

[40] MASEHIAN, Ellips; SEDIGHIZADEH, Davoud. Classic and heuristic approaches in robot motion planning—a chronological review. En: Proc. World Academy of Science, Engineering and Technology, Septiembre 2007.vol. 23, p. 101-106,

[41] MCLURKING, James. Analysis and Implementation of Distributed Algorithms for Multi-Robot Systems. Massachusetts, 2008, 166p. Tesis Doctoral. (Philosophy in Computer Science). Massachusetts institute of technology. Department of Electrical Engineering and Computer Science.

[42] MORALES GUERRERO, Rodolfo. Construcción y localización de un robot móvil diferencial para la evasión de obstáculos mediante campos potenciales artificiales y control automático. México, 2010, 166p. Trabajo de grado

(Maestría en tecnología de cómputo). Instituto politécnico nacional. Área de mecatrónica.

[43] NOUROLLAH, Ali; BAZZAZ, Saeed; MEYBODI, Mohammad. Cell Decomposition Algorithm Using Pseudo Triangulation. En: Proceedings of the International Conference on Foundations of Computer Science (FCS), 2012. 6 p.

[44] OLLERO, Aníbal. Robótica Móvil. 1 ed. Barcelona: Marcombo, S.A, 2001. 464p.

[45] OROZCO, Ulises. Planeación de trayectoria mediante campos potenciales bacteriológicos para robótica móvil. Tijuana, 2013, 124 p. Trabajo de grado (Maestría en Ciencias en Sistemas Digitales) Instituto Politécnico Nacional.

[46] OUSSMA, Khatib. Real time obstacle avoidance for manipulators and mobile robots. En: The international journal of robotics research.Vol 5, Nº 3, 1986. p. 90-98.

[47] PARKER, Lynee. SCHNEIDER, Frank. SCHULTZ, Alan. Multi-Robot Systems - From Swarms to Intelligent Automata, 3 ed. Holanda: Springer, 2003, 285p.

[48] PASSINO, Kevin. GAZI, Veysel. A class of Attraction/repulsion functions for stable swarm aggregations. En: International Journal of Control.Vol 77, Nº18.Enero, 2004; p 1567-1579.

[49] PASSINO, Kevin. Elements of decision making. En: Biomimicry for optimization, control, and automation, London, UK: Springer-Verlag, Junio2005. Ch. 2, p. 255-261.

[50] PORTA GARCÍA, Miguel. Planeación de trayectorias para robótica móvil mediante optimización por colonia de hormigas. Tijuana México, 2007, 137 p. Tesis de Maestría en Ciencias en Sistemas Digitales, Instituto Politécnico Nacional.

[51] PRAUTZSCH Harmut, BOEHM Wolfgang, PALUSZNY Marco. Métodos de Bézier and B-spline. Heidelberg: Springer-Verlag Berlin, 2002.

[52] RAJA, p, PUGAZHENTHI, s. Optimal path planning of mobile robots. A review. In: International Journal of Physical Sciences. Vol. 7, Nº 9, (Feb. 2012); p. 1314-1320.

[53] REZA, Alam. RAFIQUE, Nasir. KHAN, Planificación de Ruta de Robot Móvil en Entornos Estáticos Mediante la Optimización de Enjambre de Partículas.

Revista Internacional de Ciencias de la Computación e Ingeniería Electrónica (IJCSEE). 2015. Vol.3 Nº 3; p 253-258

[54] RIMON, Elon. KODITCHEK, Daniel. Exact robot navigation using artificial potential functions, IEEE Transactions on Robotics and Automation, Oct 1992.Vol 8, Nº 5. p. 501-518.

[55] ROGERS, David; ADAMS, James. Mathematical Elements for Computer Graphics Capítulo 5. 1 ed. Ciudad: McGraw-Hill, 1990 - p, 116-155

[56] ROURKE, Joseph. Computational Geometry in C. 2 ed. Reino Unido: Cambridge University press 2014 Capítulo 2, p 44-55.

[57] SANTILI, Corrado et al. Nonholonomic, bounded curvature path planning in cluttered environments. En: DSEA- Dipartimento di Sistemi Elettrici ed Automatione, Italia, 1995, p 363-372.

[58] SCHNEIDER, Frank. Formation Navigation and Relative Localisation of Multi-Robot Systemsh. Bonn, 2012. 132p. Tesis doctoral. Facultad de Matemáticas y Ciencias Naturales de la Rheinische Friedrich-Wilhelms-Universität zu Bonn.

[59] SCHWAGER Mac. RUS, Daniela. SLOTINE, Jacques. Unifying Geometric, Probabilistic, and Potential Field Approaches to multi-robot deployment MIT. Sept, 2010, Vol 30 nº 3, p 371-383.

[60] SEKI, Hiroaki; KAMIYA, Yoshitsugu; HIKIZU, Masatoshi. Real-time Obstacle Avoidance Using Potential Field for a Nonholonomic Vehicle. En: INTECH. Japon: Kanazawa University. P 523-542.

[61] SHAMOS, Michael. Computational Geometry. Connecticut, 1978, p 92-108 Trabajo de grado.(Doctor of Philosophy). Faculty of the Graduate School of Yale University

[62] SIERAKOWSKI, Coelho. AUGUSTO, Cesar. Bacteria colony approaches with variable velocity applied to path optimization of mobile robots, ABCM Symposium Series in Mechatronics, Volume 2. Enero 2006; p15-25.

[63] SILVA ORTIGOZA, Ramón, et al. “Una panorámica de los robots móviles”, En Telematique: Revista de la Universidad Rafael Belloso Chacin Zuli. Vol. 6, No. 3 (2007); p. 1-14

[64] SLEUMER. Nora. TSCHICHOLD, Gurman. Exact cell decomposition of arrangements used for path planning in robotics, Swiss Federal Institute of Technology, Sept 1999. 14p.

[65] SNAPE, Jaime et al. Independent Navigation of Multiple Mobile Robots with Hybrid Reciprocal Velocity Obstacles. En: IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, p 5917-5922.

[66] SNAPE, Jaime et al. The Hybrid Reciprocal Velocity Obstacle, En: IEEE Transactions on Robotics, Aug. 2011.vol. 27, no. 4, pp. 696-706.

[67] SONG, Peng; KUMAR, Vijay. A Potential Field Based Approach to Multi-Robot Manipulation. En: Departmental Papers (MEAM).En: IEEE International Conference on Robotics and Automation, Mayo, 2002.Vol 2, p1217-1222.

[68] SPONG, Mark. HUTCHINSON, Seth. Robot Modeling and Control BOOK, 3 ed. New York: John Wiley and Sons, 2005, 496 p.

[69] TANG, Lei, et al. A novel potential field method for obstacle avoidance and path planning of mobile robot. En: 3rd IEEE Int. Conf. in Computer Science and Information Technology (ICCSIT), 2010, p. 633-637.

[70] TARANILLA, María et al. Una operación entre polígonos: Sumas de minkowski.en: Geometría Computacional, de la Universidad Politécnica de Madrid. [en línea]. [Citado el 23 de Enero de 2018].

[71] TIBADUIZA, Diego et al. Planeamiento de caminos y trayectorias mediante algoritmos genéticos y campos de potencial para un robot móvil. En: Iteckne Vol 8, Nº 2, diciembre 2011.p. 183-192.

[72] UNIVERSIDAD DE GRANADA. Interpolación Spline [en línea]. <http://www.ugr.es/~prodelas/AnNumCaminos/ftp/Tema6.htm> [Citado el 22 de Marzo de 2018].

[73] VALLE, Steven. Motion planning: The essentials. En: IEEE Robot. Autom. Mag., 2011. vol. 18, Nº 1, p 79–89.

[74] VALLE, Steven. Planning algorithms. 1 ed. Cambridge: Cambridge University Press, 2006. 826 p.

[75] VLASSIS, Nikos. A concise Introduction to Multiagent systems and distributed Artificial Intelligence: Synthesis Lectures in Artificial Intelligence and Maching Learning. 1 ed. San Francisco: Morgan & Claypool Publishers, 2007. 84 p.

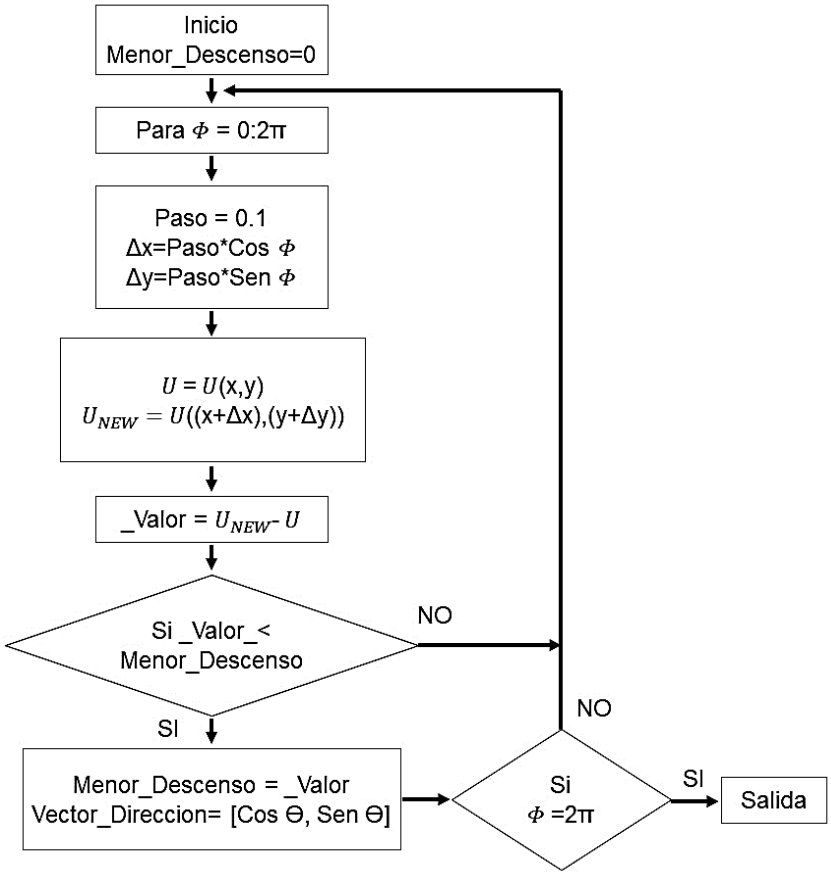
[76] WAQAR Ahmad. Motion Planning of Mobile Robot in Dynamic Enironment Using Potential Field and Roadmap Based Planner. Texas, 2003, 86p.Trabao de grado. (Maestría en Ciencia).Universidad de Texas A&M.

ANEXOS

Esta sección ha sido destinada a los algoritmos que complementan la operación de algunos de los métodos de planeación de trayectorias y a los resultados logrados con esta investigación.

ANEXO 1. CÁLCULO DE GRADIENTE DESCENDIENTE

Este anexo expone el método numérico de gradiente descendiente en diagrama de flujo.



ANEXO 2. PSEUDO-CÓDIGO ALGORITMO A*

Algoritmo *A estrella*
1. Inicialización: Crear dos listas, una para Abiertos (Nodos no evaluados) y otra para Cerrados (Nodos evaluados). Determinar el Nodo de inicio y llegada.

Calcular la función de costo f(n)=g(n); Distancia entre la meta y la posición inicial.

2. **Mientras** n!= Nodo de llegada

3. Eliminar de la lista Abiertos el nodo con la función de menor costo y ponerlo en la lista Cerrados.
4. **Si** n = Nodo de llegada
   Use los punteros para obtener ruta solución.
   Finaliza el algoritmo.
   Caso contrario
   Determine todos los nodos sucesores de n
5. Evaluar la función de costo para sucesores en lista Abiertos
   Calcular g(n): Distancia hasta el nodo n.
   Calcular h(n): Distancia para obtener desde el nodo n el nodo de llegada.
   Función de costo f(n)=g(n)+h(n)
   Fin Si
6. Asociar el costo calculado con cada sucesor que no esté en las listas y colocarlos en la lista de Abiertos, con punteros a n.
7. Asociar a los sucesores que ya estén en lista Abiertos con el menor de los valores de costo reciente y anterior.
   Min(nuevo f(n), viejo f(n))

   Fin Mientras

ANEXO 3. PSEUDO-CÓDIGO ALGORITMO DIJKSTRA

Este algoritmo requiere de un grafo dirigido ponderado de N nodos no aislados, tal como el que genera el método de Descomposición en celdas trapezoidales.

Algoritmo *Dijkstra*

1. Inicialización: Determinar Nodo inicial x.
   Vector D de tamaño N con todas sus distancias igual a inf.
   Exceptuando a x.
2. Nodo actual=x
3. **Mientras** Existan nodos no marcados
4. Se recorre todos los nodos no marcados v_i
5. Cálculo de distancia tentativa dt(v_i)
   dt(v_i)=$D_{actual} + d(actual, v_i)$
6. **Si** dt(v_i)<$D_{Vi} \rightarrow D_{Vi}$= dt($v_i$)
7. **Fin Si**
8. Se marca el Nodo actual como completo

9. Nodo actual = min(D)
   Fin Mientras

ANEXO 4. PSEUDO-CÓDIGO PARTICULE SWARM OPTIMIZATION (PSO) Y ANT BEE COLONY (ABC)

Algoritmo *PSO*

1. Inicialización:
   Determinar posición de inicio y llegada.
   Asignar posiciones y velocidades aleatorias a las partículas.
2. **Mientras** Mejor global ¡= Posición de llegada
   Para cada partícula:
3. Actualizar su velocidad considerando:
   Inercia de la partícula (la hace seguir con la misma velocidad)
   Atracción al mejor local
   Atracción al mejor global
4. Actualizar la posición de la partícula
   Calcular el valor de bienestar en la nueva posición
   Actualizar su mejor local
   Fin para
   Actualizar el mejor global del sistema
   Devolver el mejor global
   Fin Mientras

Algoritmo *ABC*

1. Inicialización:
2. Definir un criterio de parada.
3. Población de soluciones $x_{i,0,}\ i = 1, \dots, SN$
4. Evaluar la población.
5. g=1.
6. **Repetir**
7. Producir nuevas soluciones $v_{i,g}$ para las abejas empleadas y evaluarlas, haciendo:

$$v_{i,g} = x_{ig} + \varphi_{ig}(x_{ig} - x_{kg})$$

k:Índice aleatorio diferente de *i.*

φ: Número real aleatorio, uniformemente distribuido entre -1 y 1

8. Seleccionar las soluciones que serán visitadas por una abeja observadora según su aptitud.
9. Producir nuevas soluciones $v_{i,g}$ para las abejas observadoras y nuevamente evaluarlas.
10. Conservar la mejor solución entre la actual y la candidata.
11. Determinar si existe una fuente abandonada y reemplazarla utilizando una abeja exploradora.

12. Memorizar la mejor solución encontrada hasta el momento.
13. G=g+1.
14. **Hasta** g=Criterio de parada.

ANEXO 5. PSEUDO-CÓDIGO AGRUPAMIENTO K-MEDIAS

Algoritmo *k-medias*

1. Inicialización: Escoger un valor de k (número de grupos), una partición inicial $\mathbf{C}^{(0)}$ con centroides $\mathbf{Q}^{(0)}$ y un número máximo de iteraciones ($niter$).

2. Inicializar un contador $r = 1$.

   Mientras $r < niter$
   Desde j=1 **hasta** k hacer

3. Actualizar centroides

   $q_l{}^{(r)} = \frac{m_l \boldsymbol{q}_l{}^{(r-1)} - \boldsymbol{x}_i}{m_l - 1}$, $q_j{}^{(r)} = \frac{m_j \boldsymbol{q}_j{}^{(r-1)} - \boldsymbol{x}_i}{m_j - 1}$.

4. Calcular el cambio de la función objetivo

   $v_{ij} = \frac{m_j}{m_j+1} f\left(\boldsymbol{q}_j{}^{(r)}, \boldsymbol{x}_i\right) - \frac{m_l}{m_l-1} f\left(q_l{}^{(r)}, \boldsymbol{x}_i\right)$.

   donde $\boldsymbol{x}_i \in \boldsymbol{C}_l{}^{(r)}$
   Terminar **Desde**
   Si $v_{ij} \geq 0$ para todos los i,j

5. El proceso termina y la solución es $\mathbf{C}^{(r)}$

   Caso contrario
   $r = r + 1$
   Fin Si
   Fin Mientras

ANEXO 6. ALGORITMO PARA RASTERO DEL ROBOT

```
clear
close all
clc
cam=ipcam('http://192.168.0.100:8080/videofeed'); %Lectura de cámara IP
```

```
preview(cam)
%% Kmeans, encuentra el centroide para la aplicación del filtro
figure(2)
img=snapshot(cam);
imshow(img);
[cc,range]=kmfunction(img);
%% Configura el puerto serial
delete(instrfind({'port'},{'COM9'}));
s=serial('COM9','BaudRate',9600,'Terminator','CR/LF');
fopen(s);
%Referencia
V=[5 5];
steps=80;
posref=zeros(steps,3);
posref(1,:)=[1.2 0.15 pi/4];
T=0.2;
for i=2:steps
    if i>30
        V=[7 4];
    end
    posref(i,:)=updatepos(posref(i-1,:),V,T);
end
%% Evío de datos
K=(1.75/720);
figure(2)
for i=1:10000
    hold on
    BW=filtro(img,cc,range);%Filtro con centro en cc y rango "range"
    BW(1,1)=1;
    stat=regionprops(BW);
    [~,I]=max([stat.Area]); %Encuentra de las regiones encontradas, el índice de la
de mayor Area
    c=stat(I).Centroid; %El centroide de la región de mayor área.
    plot(c(1),c(2),'Color','g','Marker','x','MarkerSize',10)
 %Gráfica de la referencia
    plot(posref(1:steps,1)/K,720-posref(1:steps,2)/K,'Color','r')
    hold off
    pause(1);

    img=snapshot(cam);
    imshow(img);
    %Cálculo y envío de la posición en metros
    pos=1000*[c(1)*K,(720-c(2))*K];
    %Envió de la posición
    fprintf(s,'%d',floor(floor(pos(1))*10000+pos(2)));
```

```
end
%%
fclose(s);
```

ANEXO 7. ARTÍCULO: A QUANTITATIVE COMPARISON OF PATH PLANNING METHODS IN MOBILE ROBOTICS

Este anexo contiene el artículo seleccionado para sustentación en modalidad ponencia oral y publicación en la conferencia IEEE COLOMBIAN CONFERENCE ON AUTOMATIC CONTROL celebrada en Cartagena-Colombia del 18 al 20 de octubre de 2017.

A Quantitative Comparison of Path Planning Methods in Mobile Robotics

Jesús Alberto Getial Barragán
Departamento de Electrónica
Universidad de Nariño
Pasto, Colombia
jesusgetial@udenar.edu.co

Edwin David Erazo Caicedo
Departamento de Electrónica
Universidad de Nariño
Pasto, Colombia
eddavid95@hotmail.com

Andrés Pantoja
Departamento de Electrónica
Universidad de Nariño
Pasto, Colombia
ad_pantoja@udenar.edu.co

***Abstract*— Path planning and motion control are primary goals in mobile robotics since they are the basis of autonomous navigation. This work presents a quantitative methodology to compare different algorithms that solve the path-planning problem, taking into account their basic concepts and results obtained in three simulation scenarios. The resulting trajectories are smoothed by Spline and B-Spline interpolation methods in order to propose a solution to the non-holonomic constraints of a differential-traction robot with a tracking controller. The comparison is based on simulations to calculate indexes and evaluate different performance criteria.**

Keywords*— *Cell descomposition, Heuristic algorithms, Mobile robotics, Path planning, Potential fields.

I. INTRODUCCIÓN

La planificación de trayectorias define cómo lograr que un robot móvil encuentre una ruta óptima y libre de obstáculos desde un punto inicial hasta un punto deseado en diferentes escenarios. Este problema ha sido resuelto por técnicas y algoritmos que utilizan principios diversos tales como la minimización de funciones, teoría de grafos o métodos heurísticos, entre otros [10], [11].

Cada una de las técnicas desarrolladas tiene ventajas y desventajas al compararse en diferentes casos debido a la naturaleza de su algoritmo. Por ejemplo, los métodos reactivos o de respuesta rápida representan el entorno como funciones tridimensionales, en donde los obstáculos tienen valores altos y el vehículo debe minimizar el camino entre un punto elevado de inicio y un punto bajo de llegada. Entre estos métodos se encuentran los denominados *campos potenciales artificiales* [1], [2], [13]. Por otra parte, los métodos heurísticos como el de *optimización por enjambres de partículas* (PSO, por sus siglas en inglés), emulan comportamientos naturales para optimizar una función de bienestar con un campo de búsqueda amplio [6], [7], [14]. Algunos algoritmos como la *descomposición por celdas trapezoidales* hacen uso de la teoría de grafos para establecer trayectos entre espacios característicos del medio uniendo sus centroides con caminos definidos [8], o bien sus vértices como los grafos de visibilidad [12]. Finalmente, existen métodos basados en mapas de ruta probabilísticos, en donde la elección de la mejor trayectoria se realiza teniendo en cuenta la conformación de árboles y una búsqueda sistemática, como en la *exploración rápida de árboles aleatorios* (RRT, por sus siglas en inglés) [4], [5].

En este trabajo, haciendo uso de simulaciones, se evalúan métodos representativos para la generación de trayectorias con el fin de calcular algunos índices de desempeño como tiempo, distancia y eficiencia de las rutas. Dadas las diferentes características de las metodologías programadas, medir el desempeño en tres escenarios típicos de arreglos de obstáculos permite analizar las ventajas de cada algoritmo, orientando la elección de los métodos a la solución de problemas particulares.

Además de la metodología de comparación, se propone una estrategia para resolver el problema de las restricciones no holonómicas de locomoción de un vehículo de tracción diferencial y se presenta una técnica eficiente de control para el seguimiento de rutas del vehículo.

II. ESTRATEGIAS PROGRAMADAS PARA PLANEACIÓN DE TRAYECTORIAS

En este apartado se describen 4 algoritmos reconocidos como representativos de los métodos de campos de potencial artificial, heurísticos y basados en grafos.

A. Función de navegación para campos potenciales artificiales (PF).

En esta técnica se define el escenario como una función tridimensional $U(q)$, igual a la suma de funciones de potencial atrayentes (para el punto final deseado) y repulsivas (para el punto inicial y los obstáculos). El robot se asume como una partícula que encuentra una ruta libre de colisión definida por el gradiente descendiente del potencial. Este método se remonta a 1986, pero modificaciones recientes a la propuesta inicial se observan en [1]. Una de las limitaciones de esta estrategia es que se pueden generar espacios de mínimos locales que inhabilitan al robot para llegar al objetivo (mínimo global). Para contrarrestar este inconveniente, en [3] se introduce un tipo especial de potencial denominado función de navegación, dado por una esfera de radio r_0, centrada en el punto cartesiano inicial q_0 que contiene n obstáculos de radio r_i ($r_{obstáculos}$) centrados en q_i,

2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)

978-1-5386-0398-7/17/$31.00 ©2017 IEEE

$$U(q) = \begin{cases} \frac{\|q-qtar\|^2}{\|q-qtar\|^{2k}+B(q)^{\frac{1}{k}}} & para\ B > 0 \\ 1 & para\ B \leq 0 \end{cases}, \quad (1)$$

donde k corresponde a un factor de suavizado, $qtar$ la posición cartesiana de destino y B determina las posiciones de los obstáculos con la expresión

$$B(q) = B_0(q) \prod_{i=0}^{n} B_i(q), \qquad B_0 = -\|q - qtar\|^2 + {r_0}^2,$$
$$B_i(q) = \|q - q_i\|^2 + {r_i}^2. \quad (2)$$

En [2] se detalla el desarrollo y aplicación de esta técnica en distintos escenarios y casos de estudio.

B. Optimización por enjambre de partículas (PSO)

Este algoritmo es una técnica de búsqueda multi-agente inspirada en el movimiento de grupos de animales como insectos, pájaros y peces, entre otros [7]. En PSO, cada individuo de una población n (enjambre) de posibles soluciones es representado por un vector de posición $x_i(t)$ en un espacio multidimensional de búsqueda. Cada individuo se moviliza iterativamente de acuerdo con

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (3)$$

y una velocidad de desplazamiento v_i dada por

$$v_i(t+1) = v_i(t) + c_1\varphi_1(P_{ibest} - x_i) + c_2\varphi_2(P_{gbest} - x_i) + c_3\varphi_3(P_{nbest} - x_i), \quad (4)$$

donde φ_1, φ_2 y φ_3 son valores de una distribución aleatoria uniforme entre [0,1] que representan los factores cognitivos y sociales que influyen en cada individuo, P_{ibest} y P_{gbest} representan las mejores posiciones obtenidas por el individuo y el enjambre, respectivamente, y c_1, c_2 y c_3 son factores de ponderación.

Aunque en la versión original de PSO solo se toma la mejor posición del enjambre P_{gbest} para influir en la velocidad de cada individuo, en este caso se adiciona P_{nbest}, que representa la mejor posición del vecindario alrededor de un radio determinado [7]. El cálculo de las mejores posiciones se realiza con una función de aptitud o de *fitness* dada por un campo de potencial artificial local similar al usado en la Sección II-A [2, pp. 26-41].

En adición, el algoritmo se implementa con algunas propuestas encaminadas a la planificación de trayectorias:

- Las particulas son inicializadas en posiciones aleatorias dentro de una esfera con radio definido r y centro en la posición inicial.
- Se elimina un número fijo de particulas (nwf) con la peor fitness en cada iteración para mejorar la búsqueda.
- El algoritmo se detiene cuando P_{gbest} se encuentra en una esfera de radio pequeño con centro en la posición de meta.

C. Exploración rápida de árbol de azar (RRT)*

Como una variación al método denominado exploración rápida de grafos aleatorios (RRG, por sus siglas en inglés) [4], RRT* es un algoritmo estocástico basado en muestreo e incluido dentro de los métodos heurísticos. Mantiene la ventaja de completitud probabilística de RRT, es decir, que la probabilidad de encontrar un camino si existe, tiende a 1 exponencialmente con el número de nodos y hace que el costo de la solución encontrada converja casi seguramente al óptimo [5].

El método se puede resumir en los siguientes pasos que se efectúan hasta que se alcance el número máximo de nodos N_{max} o se obtenga el destino objetivo:

- Elección de un nodo aleatorio en el espacio de trabajo.
- Búsqueda del nodo más cercano al nodo aleatorio, entre los nodos de las ramificaciones existentes.
- Avance de una distancia predefinida (dis_p) desde el nodo más cercano hasta el aleatorio, siempre y cuando no exista colisión con algún obstáculo (al nodo más cercano, se le conoce como nodo padre).
- Búsqueda y comprobación en un radio determinado (con centro en el nodo padre) de un nodo incluso más cercano al nodo aleatorio, a partir del cual se logre un avance sin colisión. En caso verdadero, se actualiza el nodo padre.
- Agregar el nodo aleatorio escogido y el nodo padre a las ramificaciones existentes.

Cumplido el criterio de parada, se selecciona la ruta más corta.

D. Descomposicón por celdas (CD)

El método CD recurre a la descomposición vertical o trapezoidal del escenario distinguiendo entre obstáculos y posible área de recorrido, como una aplicación del desarrollo propuesto en [8]. Básicamente, esta técnica busca generar trayectorias en el espacio libre encontrado con la resolución de dos problemas: *i)* descomponer el espacio libre en celdas y *ii)* construir el grafo de conectividad. Así, primero se descompone el espacio en un mapa trapezoidal que se forma al extender rectas verticales sobre los vértices de cada objeto del escenario. Estas extensiones se detienen si interceptan un eje del grafo o los límites del espacio de trabajo. Luego se realiza un mapa de ruta con los centros de los trapezoides creados y los centros de las extensiones verticales, eliminando del proceso a los trapezoides contenidos por objetos.

Una vez especificado el mapa de ruta, se detecta la celda que contiene el punto objetivo de la ruta, tomando como partida la que contiene el punto inicial. En este caso, para elegir la mejor trayectoria entre estos puntos se aplica el algoritmo de Dijkstra [8].

III. CONTROL DE MOVIMIENTO Y RESTRICCIONES NO HOLONÓMICAS

La planificación de trayectorias no se puede limitar a la creación de un camino libre de obstáculos, sino que debe asegurarse de que dicho camino pueda ser transitado. Para esto, las trayectorias no suaves obtenidas de los algoritmos de la sección anterior (imposibles de seguir por vehículos con restricciones no holonómicas), son suavizadas con la ejecución de interpoladores Spline y B-Spline. Además, se aplica un control de movimiento para un robot de tracción diferencial por medio de realimentación de estados.

A. Interpoladores

Las Spline de tipo cúbicas son curvas paramétricas utilizadas para interpolar puntos consecutivos con polinomios de tercer orden, sin perder suavidad en las junturas o puntos de control. Su utilidad en este caso radica en que garantizan la existencia de hasta la segunda derivada. Por su parte, las B-Spline son curvas paramétricas basadas en las llamadas curvas de Beizer y el algoritmo de Boor, garantizando también la existencia de la segunda derivada.

Una descripción más detallada de ambos interpoladores se encuentra en [9, pp. 300-318].

B. Control del vehículo

El objetivo del control es establecer una estrategia para que el vehículo siga las trayectorias establecidas por los algoritmos anteriores. Para esto se utiliza el método basado en realimentación de estados propuesto en [9, pp. 275-290], en donde se considera un vehículo de tracción diferencial con un modelo cinemático sobre las coordenadas de referencia (x, y, ϕ) dado por

$$x' = -vsin\phi; \quad y' = vcos\phi; \quad \phi' = w, \tag{5}$$

donde v y w son las velocidades lineal y angular del vehículo. El error en las tres coordenadas es entonces

$$\begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} -sin\phi & cos\phi & 0 \\ -cos\phi & -sin\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{ref} - x \\ y_{ref} - x \\ \phi_{ref} - x \end{pmatrix}. \tag{6}$$

Derivando respecto al tiempo y linealizando en el punto de equilibrio $(e = 0, u = 0)$, con $u_1 = -v + v_{ref}\cos(e_3)$ y $u_2 = w_{ref} - \mathrm{w}$, se obtiene

$$e' = \begin{pmatrix} 0 & w_{ref}(t) & 0 \\ -w_{ref}(t) & 0 & v_{ref}(t) \\ 0 & 0 & 0 \end{pmatrix} e + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \tag{7}$$

Aplicando la ley de control

$$u_1 = -k_1 e_1; \quad u_2 = -k_2 sgn(v_{ref}) e_2 - k_3 e_3 \tag{8}$$

resulta en el sistema

$$e' = Ae = \begin{pmatrix} -k_1 & w_{ref} & 0 \\ -w_{ref} & 0 & v_{ref} \\ 0 & -k_2 sgn(v_{ref}) & -k_3 \end{pmatrix} e \tag{9}$$

cuyo polinomio característico linealizado está dado por $(s + 2\delta b)(s^2 + 2\delta bs + b^2) = 0$. En este caso, las constantes k_i se sintonizan con valores arbitrarios de 1, 2 y 4 respectivamente.

Para mejorar la respuesta anterior, ante variaciones de la velocidad de referencia, se aplica la técnica de "escalado de velocidad" también descrita en [9, pp.278], con constantes $\beta = 13.75\ y\ \delta = 0.9$.

Finalmente, las entradas de este controlador son obtenidas de las curvas suavizadas por los interpoladores (i.e., las coordenadas (x, y, φ) de referencia), mientras que las entradas *w* (velocidad angular) y *v* (velocidad lineal) de referencia, se logran a través del modelo inverso de la planta.

IV. MARCO EXPERIMENTAL

A. Diseño de experimentos

Los entornos para los diferentes casos de estudio se definen para garantizar las mismas condiciones en todos los algoritmos al solucionar la planificación de trayectorias. Las metodologías descritas en Sección II se implementan en Matlab® R2015a, en una plataforma con procesador Intel Core i7 de 3.0 GHz y 12 GB de memoria RAM.

Los escenarios considerados en los casos de estudio son los que demandan, en general, mayor dificultad a los métodos propuestos:

1) Tipo Trampa (Fig. 1): Los obstáculos rodean el punto inicial impidiendo una línea directa hacia el objetivo. Puede generar mínimos locales y requiere de rutas que roden la totalidad del objeto, incluso alejandose de la meta.

2) Pasaje estrecho (Fig. 2): A pesar de tener una ruta directa, es posible que no se generen trayectorias seguras o que tengan un alto grado de colisión con los obstáculos.

3) Arreglo de obstáculos (Fig. 3): Es el caso general y permite establecer la adaptabilidad de las metodologías.

B. Metodología para el análisis de resultados

Tras la simulación de los algoritmos se consideran los siguientes criterios para comparar los algoritmos, destacando los aspectos prioritarios a la hora de resolver el problema de planeación.

1) Eficacia de la solución encontrada: Se define como el cumplimiento del objetivo, es decir, generar una trayectoria libre de obstáculos desde un punto inicial a un punto final.

2) Ruta establecida: Se consideran principalmente dos aspectos: i) la seguridad de la ruta (libre de colisión) y ii) la distancia total, definida como la suma de las distancias euclidianas entre puntos de control que definen las curvas para las trayectorias originales (sin suavizado) y con interpolación Spline y B-Spline.

3) Tiempo de cómputo: Corresponde al tiempo de ejecución total del algortimo original en la plataforma utilizada, teniendo en cuenta la eliminación de rutinas de visualización y sentencias innecesarias.

4) Definición del modelo de los escenarios: Establece la dificultad de representar los espacios de trabajo, incluyendo la representación, ubicación y conformación de los obstáculos y los puntos de inicio y llegada del móvil.

C. Interpoladores y Control

Se presentan las relaciones entre el suavizado y el control de movimientos, como aporte a la generación de trayectorias.

1) Interpoladores: Se realiza un análisis comparativo de las estrategias de suavizado con los interpoladores B-Spline y Spline, que permiten establecer medidas cuantitativas como longitudes de rutas establecidas y tiempo de ejecución.

2) Control: Presenta un análisis de la ejecución del control teniendo en cuenta el tiempo de establecimiento y la reacción ante perturbaciones mediante el análisis en dos casos: entrada escalón unitario y ruta establecida por PSO para el escenario tipo trampa.

D. Parámetros de los Algoritmos

La Tabla I muestra los parámetros sintonizados para la programación de los diferentes métodos, obteniendo resultados adecuados luego de validaciones en simulación.

TABLE I. VALOR NUMÉRICO DE LOS PARÁMETROS

PF	PSO	RRT*
$* \boldsymbol{K_{trampa}} = \mathbf{5,1}$ $* \boldsymbol{K_{pasaje\ estrecho}} = \mathbf{6,5}$ $* \boldsymbol{K_{arreglo\ de\ obstáculos}} = \mathbf{4,5}$ $* \boldsymbol{r_{obstáculos}} = \mathbf{0.3}$ $* \boldsymbol{r_0} = \mathbf{5}$	$* \boldsymbol{n} = 100$ $* \boldsymbol{c_1} = 0.01$ $* \boldsymbol{c_2} = 0.025$ $* \boldsymbol{c_3} = 0.5$ $* \boldsymbol{Vel_{max}} = 0.01$ $* \boldsymbol{r} = 0.2$ $* \boldsymbol{nwf} = 5$	$\boldsymbol{N_{max}} = 500$ $\boldsymbol{dis_p} = 0.5$

V. RESULTADOS Y DISCUSIÓN

A. *Resultados*

Los resultados en la ejecución de cada algoritmo en los escenarios propuestos se resumen en las gráficas de las trayectorias obtenidas, junto con una tabla en donde se presenta el tiempo de ejecución en segundos y el cálculo de las distancias en metros para las trayectorias originales (*Dist*) y suavizadas con Spline (*Sdist*) y B-Spline (*BSdist*).

1) Escenario con obstáculo tipo trampa.

TABLE II. ÍNDICES EN OBSTÁCULO TIPO TRAMPA

Método	***Tiempo***	***Dist***	***BSdist***	***Sdist***
PF	0.07	16.93	16.72	17.01
PSO	0.64	16.54	16.07	16.72
RRT*	10.99	12.21	11.26	12.52
CD	0.37	13.66	12.05	14.63

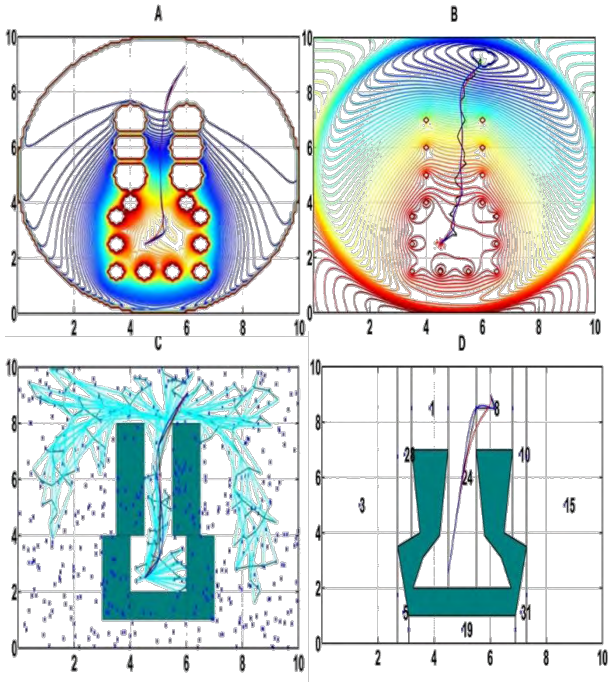


Fig. 1. Rutas obtenidas frente a obstáculo tipo trampa: original (negro), Spline (rojo) y B-Spline (azul). Los algoritmos aplicados son A) PF (superior izquierda) B) PSO (superior derecha) C) RRT* (inferior izquierda) y D) CD (inferior derecha).

2) Escenario con obstáculo de pasaje estrecho

TABLE III. ÍNDICES EN OBSTÁCULO TIPO PASAJE ESTRECHO

Método	***Tiempo***	***Dist***	***BSdist***	***Sdist***
PF	0.05	7.01	6.87	7.06
PSO	0.24	7.81	7.46	7.92
RRT*	2.43	6.77	6.74	6.92
CD	0.58	7.25	6.74	7.33

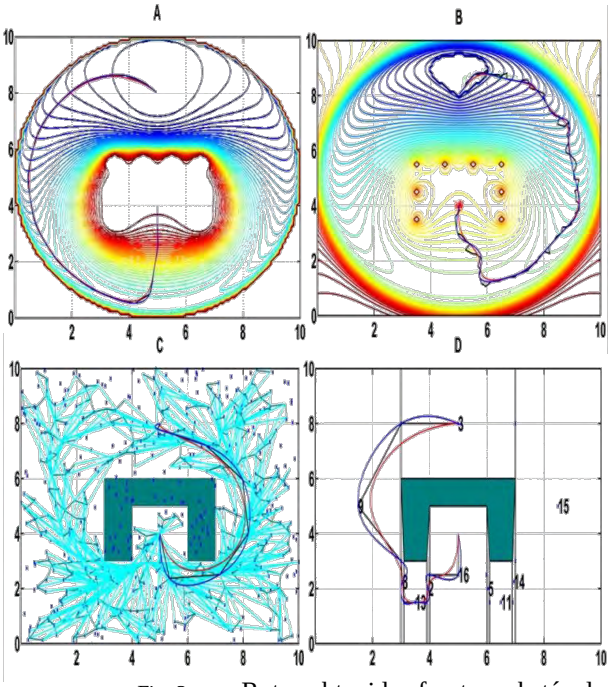


Fig. 2. Rutas obtenidas frente a obstáculo tipo pasaje estrecho: original (negro), Spline (rojo) y B-Spline (azul). Los algoritmos aplicados son A) PF (superior izquierda) B) PSO (superior derecha) C) RRT* (inferior izquierda) y D) CD (inferior derecha).

3) Escenario con arreglo general de obstáculos

TABLE IV. ÍNDICES EN ARREGLO DE OBSTÁCULOS

Método	***Tiempo***	***Dist***	***BSdist***	***Sdist***
PF	0.06	9.88	9.58	9.97
PSO	0.39	12.44	11.96	12.64
RRT*	19.04	7.84	6.82	8.06
CD	0.72	14.55	13.80	16.14

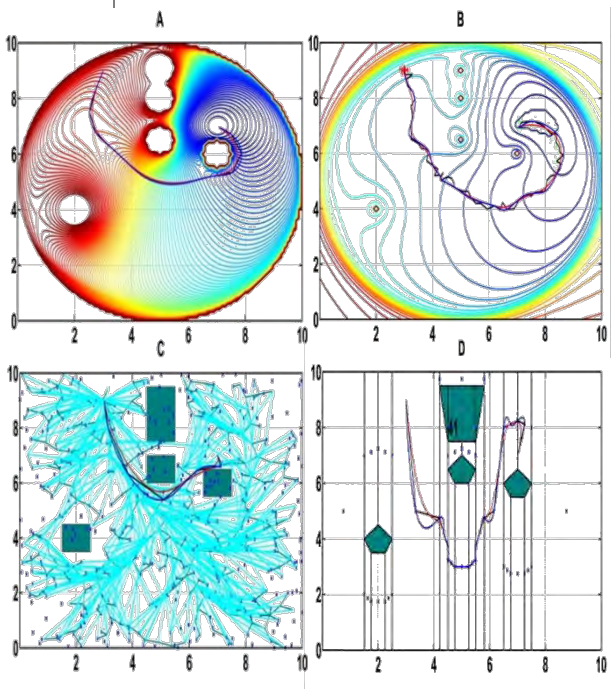


Fig. 3. Rutas obtenidas frente a un arreglo general de obstáculos: original (negro), Spline (rojo) y B-Spline

(azul). Los algoritmos aplicados son A) PF (superior izquierda) B) PSO (superior derecha) C) RRT* (inferior izquierda) y D) CD (inferior derecha).

B. Análisis de resultados

Con los resultados mostrados se analiza cada método en torno a los criterios de comparación establecidos, considerando todos los escenarios en cada subsección.

1) Eficacia

Aunque para cada una de las situaciones planteadas en este trabajo se encuentra una solución con todos los algoritmos, no se puede generalizar que para cualquier escenario se vaya a poder solucionar el problema.

En los métodos que requieren de una función multidimensional para describir el escenario (como en el caso PF), es necesario que esta función se defina evitando mínimos locales, valles extensos, y que además los obstáculos y la relación inicio-meta estén muy bien definidos. En consecuencia, en el algoritmo de PSO se pueden presentar mínimos locales, sin embargo, el aumento del número de partículas presentes en el espacio de trabajo disminuye la probabilidad de caer en mínimos locales y garantiza una mayor evaluación del entorno a costa de una mayor complejidad computacional.

En cambio, el algoritmo RRT* es capaz de encontrar una solución siempre y cuando tenga las ramas suficientes en su mapa de ruta para unir el punto inicial y final. Por último, para algoritmos basados en grafos como el CD, al construir rutas en el espacio libre, siempre se hallará una solución al problema, si es que existe.

2) Ruta establecida

Según los datos de las Tablas y Figuras 1 a 3 para todos los escenarios, el método RRT* genera la distancia más corta de los cuatro algoritmos probados debido a la gran cantidad de ramas que se especificaron para la creación del mapa de ruta. Así, se generan trayectos muy cercanos a los obstáculos, dado que a primera vista, el camino que rodea los límites del objeto es el mejor posible. Estas trayectorias pueden representar un problema en la implementación en ambientes y vehículos reales, puesto que se pueden generar colisiones.

Por su parte, el método de CD requiere de la descomposición trapezoidal del espacio de trabajo y los puntos que definen la trayectoria están situados en el centro y en los extremos derecho e izquierdo de cada trapezoide. Entonces, la distancia de los tramos de ruta depende de cómo se cree el mapa trapezoidal (dependiendo mucho del escenario) y en ocasiones produciendo tramos de ruta muy cercanos o muy lejanos a los objetos. Como se observa en la Figura 1, se pueden ocasionar colisiones, o, al contrario, como en la Figura 3, dar lugar a tramos de ruta innecesariamente extensos.

El método PF tiene las segundas mejores distancias recorridas en 2 de los 3 casos analizados. Este efecto no es inherente a la naturaleza del algoritmo sino a la definición de la función de potencial, pues el camino seguido desde cualquier punto depende del descenso del gradiente. De esta manera, una buena representación del ambiente supone una ventaja de flexibilidad ante los dos algoritmos RRT* y CD, ventaja que, por las mismas razones, también se asocia al método PSO.

En cuanto a la suavidad de la ruta, se observa que los caminos originales más suaves se encuentran con el método PF dado el cálculo por gradiente descendente, que garantiza la existencia de al menos la primera derivada de la curva. PSO en cambio genera trayectorias con bastantes fluctuaciones, ya que la ruta es definida por varios individuos del enjambre. En los métodos CD y RRT*, las rutas originales están representadas por segmentos de líneas rectas, obteniéndose trayectorias continuas, pero nunca suaves.

3) Tiempo total de computación

Los datos de las Tablas 1 a 3 corroboran que PF es el algoritmo con menor tiempo computacional por ser un método reactivo (tiene definida previamente la función de potencial). Sin embargo, para mejorar aspectos como eficacia y suavidad de la ruta, es preciso incrementar el tiempo de ejecución, puesto que estas variables están relacionadas con la magnitud del paso, el radio de muestreo y los puntos del radio de muestreo en el cálculo del gradiente descendiente. Los resultados de la Sección V-A.1 se obtuvieron con una magnitud de paso igual a 0.05 y un radio y puntos de muestreo igual a 0.1 y 18 respectivamente, valores con los cuales se encuentran los mejores resultados luego de una serie de ejecuciones del algoritmo.

El tiempo de ejecución de PSO está sujeto a la cantidad de individuos necesarios para la solución del problema, cuyo aumento repercute en el aumento progresivo del tiempo de ejecución.

Por su parte, el tiempo que toma la ejecución del algoritmo CD depende de la complejidad del escenario. Así, para escenarios con muchos vértices en los obstáculos, el tiempo tenderá a incrementarse. Esto se observa comparando el escenario de trampa que tiene 8 vértices y tarda 0.37 s; el de pasaje estrecho con 15 vértices y 0.58 s, y el de arreglo de obstáculos con 18 vértices y 0.72 s.

El método RRT* a su vez, tiene el mayor tiempo de ejecución debido a que se deben producir la mayor cantidad de rutas posibles esparcidas por todo el espacio de trabajo, bajo la consideración de que cada una de ellas sea segura y con el objetivo de elegir al final la mejor de ellas.

4) Definición de espacio de trabajo

Aunque se haya logrado la representación de todos los escenarios propuestos, cabe resaltar que hay escenarios que implican mayores dificultades de modelado debido a la cantidad de obstáculos y la relación entre estos y los puntos de inicio y llegada del móvil. Los métodos de mayor dificultad de representación son aquellos que requieren la definición de una función multidimensional, como PF, que necesitan de la sintonización de múltiples parámetros para disminuir la formación de mínimos locales y encontrar una buena relación de gradiente descendiente entre el punto inicial y final para evitar colisiones. Esta sintonización puede resultar no factible en particular cuando el número de obstáculos crece.

Aunque PSO también requiere una función de representación del escenario, la sintonización de parámetros no es exigente, puesto que por su naturaleza de múltiples individuos y con componente estocástica, puede escapar de los mínimos locales mediante la evaluación de muchos puntos alrededor del móvil.

El método de CD tiene un grado de dificultad alto a la hora de definir los polígonos, de tal manera que algunas coordenadas de los vértices no coincidan con las de otros puntos de interés. Aunque existen algoritmos que evitan estas repeticiones, su implementación depende del escenario.

A diferencia de los anteriores, RRT* representa fácilmente cualquier tipo de escenarios sin tener en cuenta su complejidad.

Finalmente, la simulación de la forma compleja de obstáculos reales, es un problema inherente a todos los algoritmos de planificación de trayectorias.

C. Interpoladores y control

1) Interpoladores B-Spline y Spline

Teniendo en cuenta las Figuras 1 a 3, se observa que los interpoladores cumplen la función de suavizar todas las rutas efectivamente. En cuanto a la distancia, el método B-Spline presenta los mejores resultados (Tablas 1 a 3), puesto que no pasa necesariamente por todos los puntos de control asociados a la curva de referencia. Al contrario, Spline abarca todos los puntos de control asociados, a precio de acercarse mucho a los objetos en algunos tramos de curva o generar curvas de amplitud innecesaria, como se puede observar Figura 1D.

Sin embargo, para efectos de implementación de los algoritmos, B-Spline tiene mayor tiempo de ejecución y menor posibilidad de programación en paralelo, puesto que para la evaluación de la curva paramétrica resultante se debe calcular una sumatoria de todos los puntos de control asociados.

2) Control por retroalimentación de estados.

En la Figura 4 se muestran los resultados para el control del vehículo utilizando la técnica de control descrita en la Sección III-B. En la gráfica superior izquierda se tiene la respuesta de la velocidad angular ante un escalón unitario y una perturbación. Se puede observar que el seguimiento de la referencia con constantes arbitrarias (línea naranja), no presenta oscilaciones marcadas en comparación con la técnica de escalado de velocidad (línea azul). Sin embargo, el tiempo de establecimiento del control se incrementa, efecto que es más evidente con la respuesta de velocidad lineal presentada en la gráfica superior derecha.

Las oscilaciones y la reducción del tiempo de establecimiento debido al control exigente de la técnica de escalado de velocidad, se manifiestan en un seguimiento mucho más eficiente y preciso de la ruta (Figura 4C) dónde la línea negra representa la trayectoria deseada que lleve al vehículo.

Por último, en la Figura 4D se muestra que es posible aplicar el modelo inverso a una curva continua y suave (al menos en su primera derivada), para obtener trayectorias viables en la aplicación del control propuesto. En línea negra se presenta la trayectoria obtenida después del suavizado con

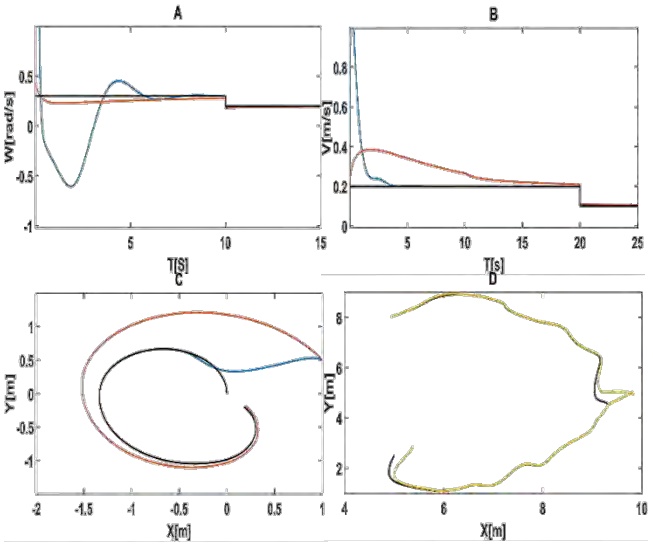


Fig. 4. Resultados del control aplicado al robot de tracción diferencial. En negro se muestran las referencias deseadas, en azul los resultados para la técnica de "escalado de velocidad", en naranja la respuesta una sintonización con constantes arbitrarias y en amarillo la trayectoria obtenida por el control para un camino generado por la técnica PSO.

la técnica Spline para un camino elaborado por el método PSO en el arreglo de obstáculos tipo trampa. En línea naranja se presenta la trayectoria obtenida por el control con condiciones iniciales distintas y una perturbación en un tiempo arbitrario. Se nota que el control logra seguir la referencia convenientemente y que se recupera de la perturbación que puede sufrir el vehículo en su recorrido.

Estas condiciones simulan ambientes reales a los que puede estar sometido un robot de tracción diferencial y el comportamiento visible de la respuesta demuestra que, partiendo de las trayectorias dadas por los métodos, es posible la implementación de cada técnica en plataformas físicas.

VI. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se presenta una comparación cuantitativa de diferentes métodos de planeación de trayectorias con distintas características fundamentales. Al proponer diferentes escenarios y realizar simulaciones de cada uno de los algoritmos, se establecen varios índices que permiten clasificar los métodos y evaluar su desempeño.

Finalmente, haciedo uso de interpoladores se establece una alternativa ante el problema de restricciones no holonómicas en la implementación de las trayectorias, ya que se generan curvas suaves que pueden ser seguidas por controles cinemáticos simples como el presentado.

Para trabajos futuros es pertinente aumentar la complejidad de los problemas para métodos en más de dos dimensiones, teniendo en cuenta elevación, tiempo, implementaciones online o que consideren aplicaciones con múltiples robots.

REFERENCIAS

[1] O. Lei, et al. “A novel potential field method for obstacle avoidance and path planning of mobile robot,” in *3rd IEEE Int. Conf. in Computer Science and Information Technology (ICCSIT)*, 2010, pp. 633-637.

[2] L. Leng-Feng., “Decentralized motion planning within an artificial potential framework (APF) for cooperative payload transport by multi-robot collectives,” M.S Thesis, Dep. Mechanical and Aerospace Eng., University of New York at Buffalo, U.S, 2004.

[3] E. Rimon and D. E. Koditchek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation.*, vol. 8, no. 5, pp. 501-518, 1992.

[4] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *International Journal of Robotics Research*., vol 30, no. 7, pp. 846–894, 2011.

[5] S. Karaman and E. Frazzoli, “Incremental sampling-based algorithms for optimal motion planning,” in *Robotics: Science and Systems (RSS)*, pp. 3-8, 2010.

[6] J. Kennedy. “Particle swarm optimization,” in *Encyclopedia of machine learning*, 2 ed, U.S: Springer, pp. 760-766, 2011.

[7] M. Alam, M. Rafique, and M. Khan, “Mobile Robot Path Planning in Static Environments using Particle Swarm Optimization,” *International Journal of Computer Science and Electronics Engineering (IJCSEE)*, vol. 3, no. 3, 2015.

[8] De Berg. Mark, et al. “Robot motion planning,” in *Computational geometry: algorithms and applications*, 3th ed, Berlin, DE: Springer, ch. 13, pp. 283-306, 2000.

[9] A. Ollero, *Robótica. Manipuladores y robots móviles*, Barcelona, ES: Marcombo, 2001.

[10] S. M. LaValle, ‘‘Motion planning: The essentials,’’ *IEEE Robot. Autom. Mag*., vol. 18, no. 1, pp. 79–89, 2011.

[11] Y. Raja, S. Pugazhenthi, “Optimal path planning of mobile robots: A review,” *International Journal of Physical Sciences*, vol. 7, no. 9, pp. 1314-1320, 2012.

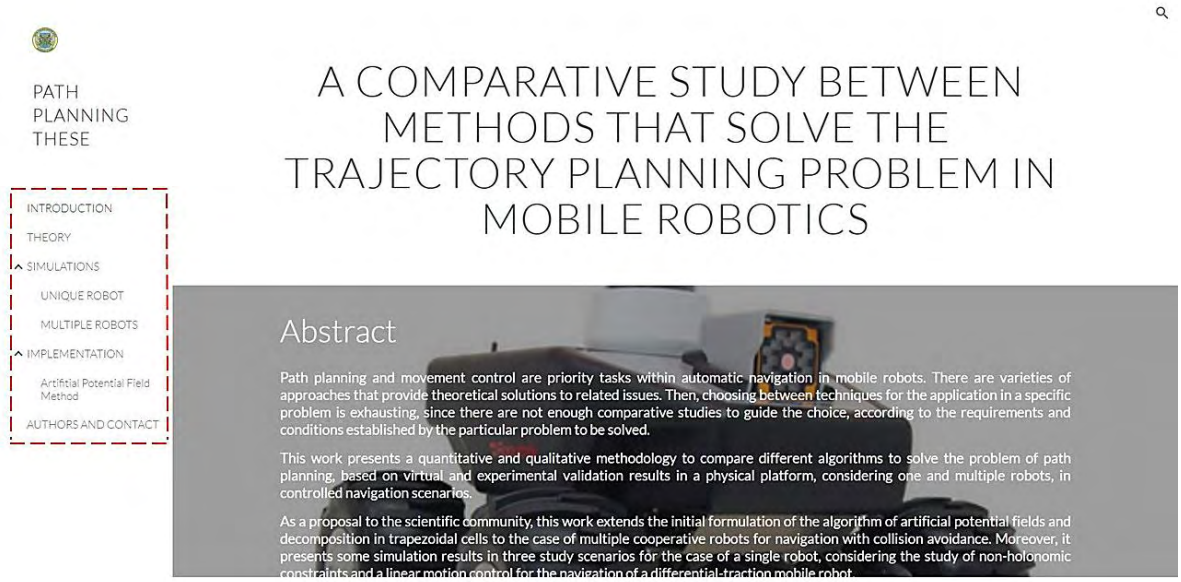
[12] J. Luna, R. Sánchez and L. Pinzón, “Finding path through the algorithms: genetic and dijkstra using maps of visibility,” *Scientia et Technica*, vol. 2, no. 5, pp. 107-112, 2012.

[13] K. Passino., “Elements of decision making,” in *Biomimicry for optimization, control, and automation*, London, UK: Springer-Verlag, ch. 2, pp. 255-261 2005.

[14] E. Masehian, D. Sedighizadeh, “Classic and heuristic approaches in robot motion planning—a chronological review,” in *Proc. World Academy of Science, Engineering and Technology*, vol. 23, pp. 101-106, 2007.

ANEXO 8. PÁGINA WEB

Con el fin de lograr gran difusión de los resultados de esta investigación, se realiza una página web en Google Sites, que permite la libre descarga de los códigos realizados tanto para validaciones virtuales como de implementación en plataforma física. De igual manera, presenta información relativa al proyecto, guiando al internauta en la temática de planeación de trayectorias, facilitándole los documentos derivados de este estudio y manuales de usuario para cada algoritmo.



El link para su búsqueda es:

<https://sites.google.com/view/pathplanningthese>