Vol. 22, n.° 3, pp. 55-68, 2023
Revista UIS Ingenierías
Página de la revista: https://revistas.uis.edu.co/index.php/revistauisingenierias

Enrutado polilineal basado en geometría para la planeación de movimiento en ordenamiento de objetos

Geometry-based polylinal routing for motion planning in object sorting

Pedro Alejandro Montaño-Herrera [1a], Juan Pablo Sosa-Esquivel [1b], Marco Antonio Jinete-Gómez [1c]

[1] Semillero de Investigación Open-Creator, Facultad de ingenierías, Universidad Piloto de Colombia, Colombia. Orcid: 0000-0003-1060-15231 [a], 0000-0002-7474-4286 [b], 0000-0002-2591-1308 [c]
Correos electrónicos: pedro-montano@upc.edu.co [a], juan-sosa@upc.edu.co [b], marco-jinete@upc.edu.co [c]

Recibido: 24 octubre, 2022. Aceptado: 26 mayo, 2023. Versión final: 1 julio, 2023.

Resumen

En esta investigación se propone el método de enrutado polilineal basado en geometría como una solución a la planificación de movimiento en ejercicios de clasificación de objetos para procesos de manufactura. Este algoritmo se basa en propiedades geométricas que surgen de la interacción entre los objetos dentro del espacio de configuración. El método propuesto en este trabajo, en su fase experimental, logró generar rutas suaves con un tiempo de procesamiento de entre 62.5-125 ms en un ordenador equipado con un procesador AMD Ryzen 7 2700X Eight-Core 3.70 GHz y 16 GB de memoria RAM. En comparación con el algoritmo RRT, se observa una mayor eficiencia del 38% al 48%, lo que se traduce en una reducción de los procesos iterativos y un tiempo de respuesta más corto. Por lo tanto, el método planteado es una solución viable para resolver escenarios de planificación de movimiento en el ejercicio de ordenamiento de objetos.

Palabras clave: Robodk; robótica; planeación de movimiento; simulación; automatización; visión artificial.

Abstract

This paper proposes the geometry-based polylinal routing method as a solution for motion planning in object sorting exercises in manufacturing processes. This algorithm is based on geometric properties that arise from the interaction among objects within the configuration space. The method proposed in this paper, during its experimental phase, successfully generated smooth routes with a processing time ranging from 62.5-125 ms on a computer equipped with an AMD Ryzen 7 2700X Eight-Core 3.70 GHz processor and 16 GB of RAM. When compared to the RRT algorithm, it exhibits a higher efficiency of 38% to 48%, resulting in a reduction in iterative processes and a shorter response time. Therefore, the proposed method presents a viable solution for addressing motion planning scenarios in object sorting exercises.

Keywords: Robodk; robotics; motion planing; simulation; automation; artificial vision.

ISSN impreso: 1657 - 4583. ISSN en línea: 2145 – 8456, **CC BY-ND 4.0**
Como citar: P. A. Montaño-Herrera, J. P. Sosa-Esquivel, M. A. Jinete-Gómez, "Enrutado polilineal basado en geometría para la planeación de movimiento en ordenamiento de objetos," *Rev. UIS Ing.*, vol. 22, no. 3, pp. 55-68, 2023, doi: https://doi.org/10.18273/revuin.v22n3-2023005

1. Introducción

La cuarta revolución industrial busca lograr mejoras en productividad, calidad, automatización, transporte, seguridad y flexibilidad [1]. Esto se ha estado realizando a través de datos tomados de la industria manufacturera, opinión de expertos. Apoyándose en las continuas mejoras de infraestructuras de comunicación, computación y control, se impulsa a la industria consintiendo la integración de nuevas máquinas, métodos de aprendizaje e infraestructuras de fabricación [2]. Los ambientes de simulación han desempeñado un papel importante, haciendo predicciones de fenómenos del mundo real que, aunque exista una complejidad de variables e incertidumbres, se acercan y permiten hacer análisis profundos, diseños robustos, además de ser un gran apoyo a procesos educativos. Un buen ejemplo sobre la utilidad de las simulaciones está en el diseño y análisis de sistemas de fabricación, con la creación de modelos 3D y visualizaciones realistas [3]. Es decir, que es una herramienta útil para afrontar los desafíos del área, entre los que están la optimización de diseños de ingeniería, sistemas de fabricación, control de plantas de producción, mantenimiento y la mejora de calidad [4].

En torno a esta área, el uso de técnicas relacionadas con el procesamiento de imágenes, han estado estrechamente relacionadas con diferentes problemáticas en el planeamiento y la ejecución de movimientos en robots. Con ello, la industria ha pasado a una etapa conocida como 4.0, en donde el procesamiento de imágenes en labores de visión artificial y uso de métodos estadísticos, han permitido el surgimiento e implementación de nuevas y eficientes técnicas relacionadas con la automatización [5]. Nuevas herramientas inteligentes, integrando múltiples instrumentos como lo son sensores, cámaras o aplicaciones en técnicas de visión e inteligencia artificiales [6], útiles para la extracción de datos y toma de decisiones que rodean el entorno de una herramienta, buscan que una tarea repetitiva o monótona sea mucho más fácil de realizar y se reduzca cada vez más el riesgo de algún accidente, un ejemplo reciente en la industria de hoy en día son los vehículos autónomos [7]. De la misma manera, en aplicaciones más enfocadas a la automatización y manufactura se encuentran los robots manipuladores, creados para realizar tareas como desplazar y sostener objetos. Uno de los ejercicios en los que más se ocupan los manipuladores en configuraciones industriales como mantenimiento de máquinas, ensamblaje o selección de contenedores es el de pick and place [8], [9]. Tarea que se logra a través de múltiples técnicas como la identificación de objetos, obstáculos, planeación de movimiento, cálculo de trayectorias y el control de sistemas dinámicos.

La planeación de movimiento es uno de los temas más importantes en la robótica hoy en día. Debido a que, las necesidades actuales de los procesos de manufactura son cada vez más variadas y complejas en sus aplicaciones. Un ejemplo de esto lo señala J. Zhong [10] con las tareas de soldaduras complejas en la industria naval que, debido a la demanda de soldadura flexible y de alta eficiencia, se ve con urgencia que este proceso sea llevado a cabo por medio de la planificación de rutas. Para ello, se define el espacio de configuración (c-space) el cual, dicta las restricciones, ya sean de velocidad, aceleración o alcances, que se deben cumplir antes de elaborar una ruta libre de colisiones para que el manipulador vaya del punto inicial al punto final.

En la actualidad algoritmos como el método de hoja de ruta probabilística (PRM) [11] y el método de árbol aleatorio de exploración rápida (RRT) [12], han demostrado ser eficientes debido a su complejidad computacional y su flexibilidad para encontrar soluciones en distintos tipos de escenarios. Por lo cual, se han tomado como base relevante para el desarrollo de otros algoritmos o de mejoras de estos mismos. A través del estudio [10], se puede ver la influencia de estos algoritmos, en donde para realizar la planeación de rutas de un manipulador en un entorno semicerrado y estrecho, se utilizó un algoritmo Deep-RL sin modelo DDPG para entrenar la navegación en el espacio de configuración. Por otra parte, en [13] se desarrolla una metodología basada en el muestreo para tratar lo que denominaron fuentes críticas y aplicando dos algoritmos Local Critical Sourse - RRT (LCS-RRT) y Critical Source - RRT (CS-RRT), que consiguen atravesar pasajes estrechos. También, en [14] proponen un algoritmo RRT mejorado para robots industriales que operan en entornos complejos que ejecutan la planificación de rutas con un algoritmo de búsqueda incremental, el cual busca generar rutas fiables en un menor tiempo. Asimismo, [15] plantea aplicar un algoritmo RRT mejorado en conjunto con un algoritmo genético y un método de suavizado para conseguir que la planeación de rutas sea más limpia y fácil de realizar.

A través de lo expuesto, el presente trabajo se centra en el desarrollo de un algoritmo de planeación de movimiento. El método de enrutado polilíneal basado en geometría (EPBG) tiene como base encontrar soluciones a la evasión de obstáculos para encontrar una ruta eficiente a un punto objetivo, por medio de propiedades geométricas que surgen entre la interacción de los artículos dentro del espacio de configuración, teniendo en cuenta espacios de trabajo, objetos y objetivos. El escenario de su desarrollo es un ambiente simulado basado en un sistema automatizado de una planta de producción. Dicho escenario se concentra en la ejecución

de un estudio enfocado a una escena real, en donde muchas informaciones relacionadas con el referenciamiento espacial en una escena 3D, planificación de movimiento y posicionamiento de objetos son captadas mediante un sistema de monitoreo simulado basado en tan solo una cámara, con el cual brazo manipulador ABB IRB 120-0.6 [16] se ubicará y realizará las rutas que el algoritmo encuentre.

Para explicar el algoritmo y su funcionamiento, este artículo se divide en las siguientes partes: Espacio de configuración, donde se explica cómo se obtiene el espacio de configuración para el requerimiento de ordenamiento de objetos. La sección del algoritmo propuesto, donde se explican a profundidad los métodos principales que desarrollan el algoritmo. La sección del montaje experimental, que muestra el montaje realizado para la evaluación del algoritmo. Resultados y discusiones, donde se compara el rendimiento del algoritmo con respecto a RRT y se hablan de sus puntos fuertes y débiles. Por último, se finaliza con la sección de conclusiones.

2. Espacio de configuración

El espacio de configuración (c-space), guarda toda la información relevante para que el algoritmo de planeación de movimiento pueda plantear una ruta. Dentro de este, está toda la información espacial de los objetos, obstáculos y límites físicos del robot manipulador. Aunque, para poder ser determinado, se debe tener en cuenta la configuración de la cámara, extraer los parámetros de las imágenes, y también, determinar todas las ubicaciones y orientaciones donde el robot manipulador se puede ubicar, teniendo en cuenta el ambiente de trabajo.

Para este caso de estudio, se planteó captar la escena se utiliza una cámara ubicada sobre la mesa de trabajo y el robot como se muestra en las Figuras 1 y 2. La cámara es una cámara afín, es decir una cámara que no posee errores no lineales como lo pueden ser lentes distorsionados, esta información debe ser aclarada principalmente porque es a partir de la información de la matriz de transformación de la cámara 1 y de su inversa, que se traducen las coordenadas halladas por la trayectoria al espacio 3D y viceversa.

Las imágenes tomadas por la cámara, para poder ser interpretadas, necesitan pasar por un proceso de segmentación, para esto se utilizó una herramienta de anotación llamada Js segment annotator basada en SLIC super pixels Figura 3, haciendo una segmentación interactiva coarse-to-fine, ajustando la resolución de los superpixels a lo necesario para marcar segmentos pequeños y aplicando suavizados con morfologías para remover artefactos [17]. La salida de este anotador son unas mascaras con etiquetas, para este ejercicio se crearon 11 clases las cuales se definen entre herramientas, recipientes y fondo, como se muestra en la Tabla 1 y la Figura 4.

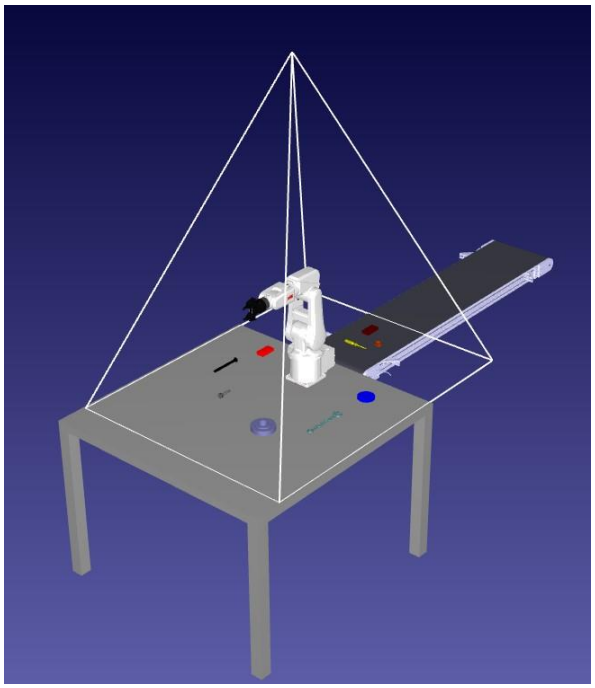

Figura 1. Imagen de representación de ubicación de la cámara respecto a la escena. Fuente: elaboración propia.

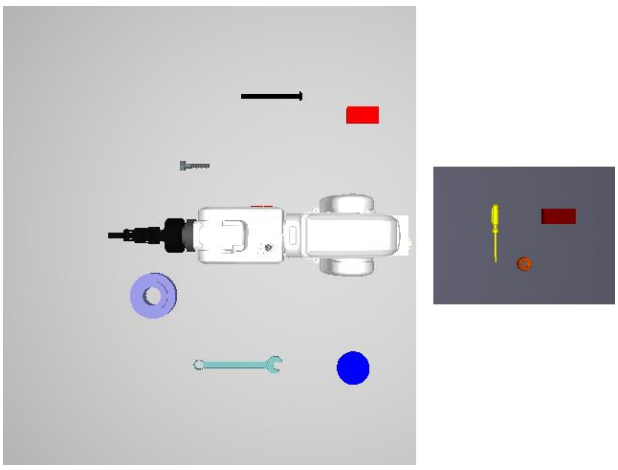

Figura 2. Vista superior tomada por la cámara afín. Fuente: elaboración propia.

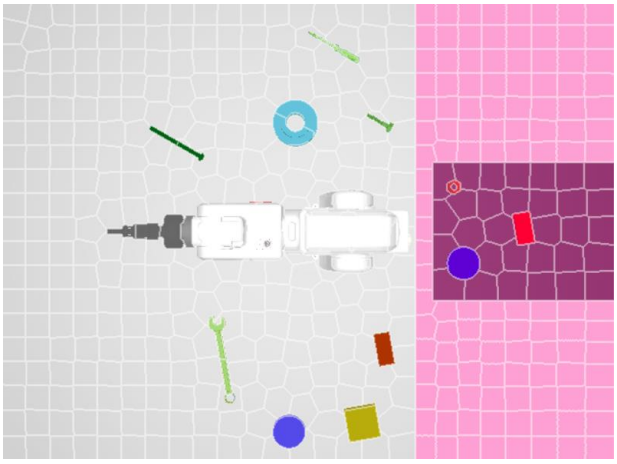

Figura 3. Herramienta de anotación js segment annotator con la cuadrícula de SLIC super pixels. Fuente: elaboración propia.

Para evitar choques entre los objetos y la herramienta del robot y, además, mantener los límites de movimiento del robot manipulador. Es importante definir para cada objeto y para el manipulador unos radios de seguridad. Los criterios para estos radios de seguridad se basan primero en definir los límites físicos inferiores del espacio de trabajo del robot, Figura 5, y segundo en definir para cada objeto un radio mínimo de seguridad.

Tabla 1. Clases presentes en la línea de producción

Clase 1	**Id**
Recipiente 1 (Cilindro azul)	0
Recipiente 2 (Cubo amarillo)	1
Recipiente 3 (Ortoedro rojo)	2
Recipiente 4 (Ortoedro café)	3
Llave	4
Tornillo	5
Pasador	6
Cuña o Cojinete de balinera	7
Destornillador	8
Tuerca	9
Fondo o fuera de la mesa	10

Fuente: elaboración propia.

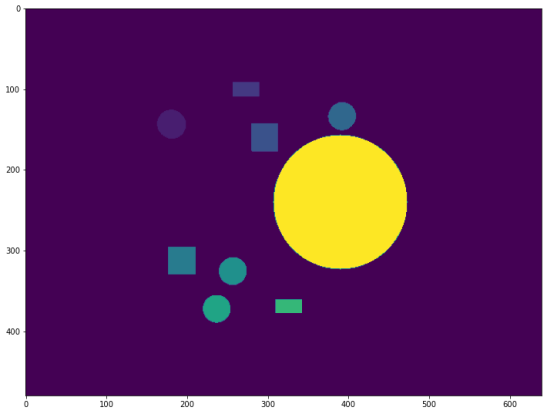

Figura 4. Mascara de salida del anotador.

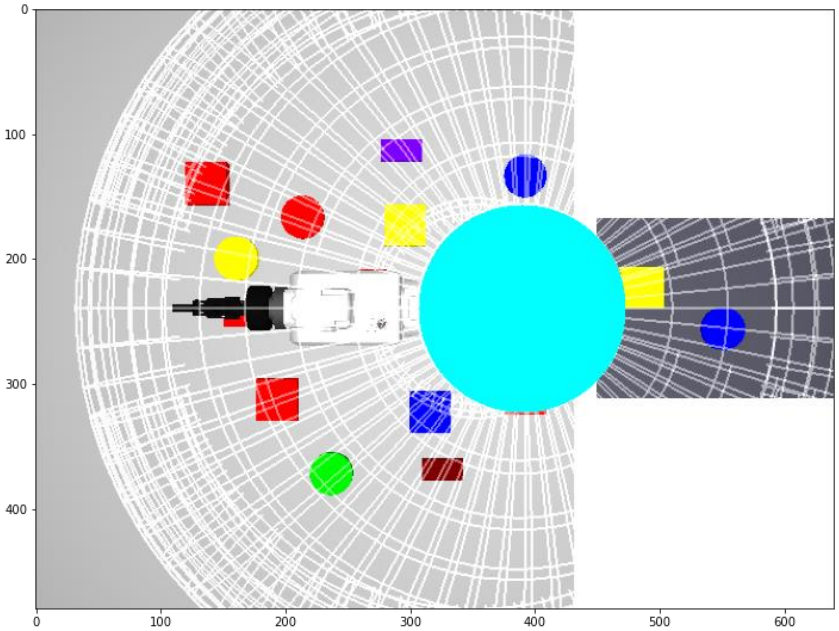

Figura 5. Espacio de trabajo del robot manipulador, el límite inferior representado por el color cian y los límites exteriores por la malla esférica alrededor. Fuente: elaboración propia.

Para esto se define un área cuadrada para cada objeto y a partir de esa área se toma como criterio la relación que existe de un cuadrado inscrito en una circunferencia. La relación que existe dentro de un cuadrado inscrito en una circunferencia es de $R = \sqrt{\frac{L^2}{2}}$, donde L es la distancia de un lado del cuadrado y R es el radio de la circunferencia, Figura 6.

Como resultado se obtiene la información espacial de cada objeto con su respectiva corrección de radio de seguridad y los límites inferiores del robot dentro del espacio de configuración Figura 7.

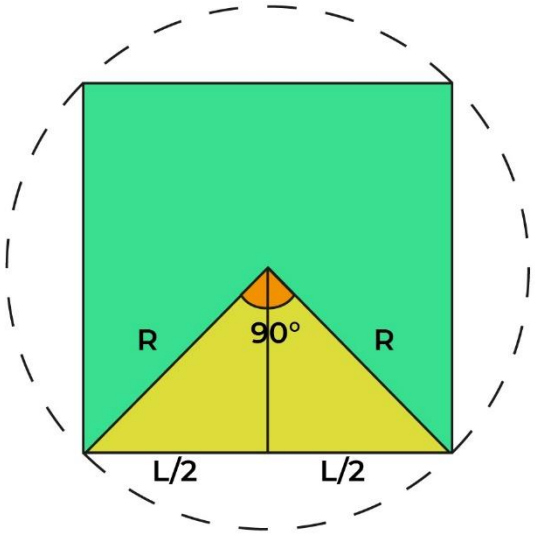

Figura 6. Representación de la relación de un cuadrado inscrito en una circunferencia. Fuente: elaboración propia.

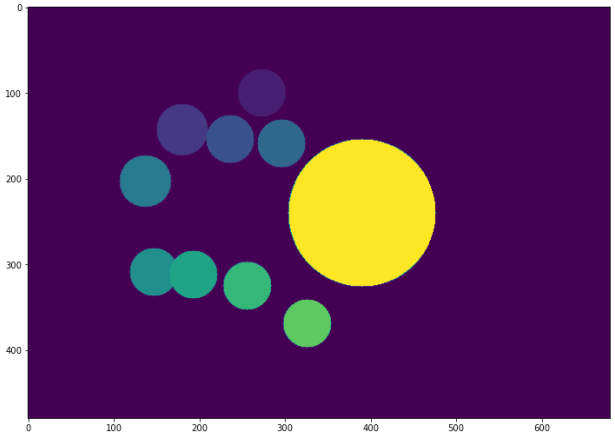

Figura 7. Mascara de representación del espacio de configuración con los radios de seguridad definidos para cada objeto. Fuente: elaboración propia.

3. Algoritmo propuesto

A partir de lo expuesto en el caso de estudio, el algoritmo propuesto, en adelante EPBG, mantiene la esencia de los algoritmos por muestreo, como lo es RRT [12]. Es decir, que no construye una cuadrícula en el espacio libre (f-space) para posteriormente generar las rutas, como lo hace PRM [11]. De manera general el algoritmo consiste en la obtención de líneas que se forman a través de la interacción de los puntos inicial (Pi) y final (Pf) con los obstáculos, como se observa en la Figura 8. Una vez determinadas se realiza una operación de muestreo en la que se recorre y se selecciona un punto en una recta. El punto seleccionado se comportará como un nuevo punto inicial (Pi) hasta que dejen de existir colisiones y se llegue a una trayectoria continua hasta el punto final (Pf).

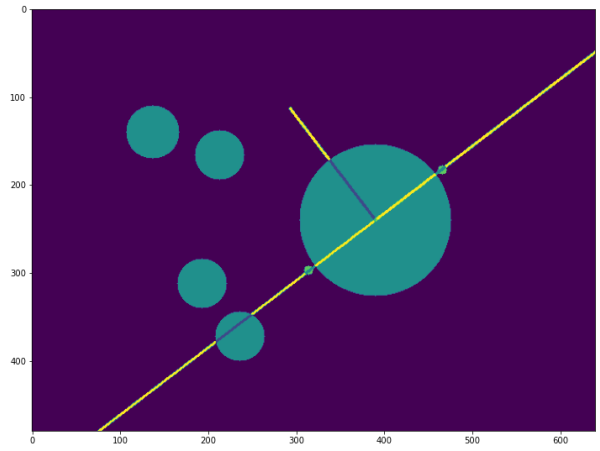

Figura 8. Representación de las líneas formadas a través de interacciones en el espacio de configuración. Fuente: elaboración propia.

Para comprender el algoritmo 1 (EPBG) (Tabla 2), se deben definir todos los parámetros y variables que interactúan dentro de él. Primero, el algoritmo recibe como entrada el punto inicial (Pi), el objetivo final a alcanzar (Pf) y su salida deberá ser una ruta que conecte dichos puntos sin chocar con ningún obstáculo (Ruta). Segundo, la variable de escala de radio de seguridad (EscalaRS), que se declara inicialmente como 1, se refiere a un espacio de seguridad que se le da a cada obstáculo. Al incluir esta variable se busca que no existan choques con la herramienta del manipulador y los obstáculos. Esta variable multiplica el valor que se le da como radio de seguridad a cada obstáculo, y al aumentar su valor modificará la selección de los puntos para la ruta, de modo como se muestra en la Figura 9.

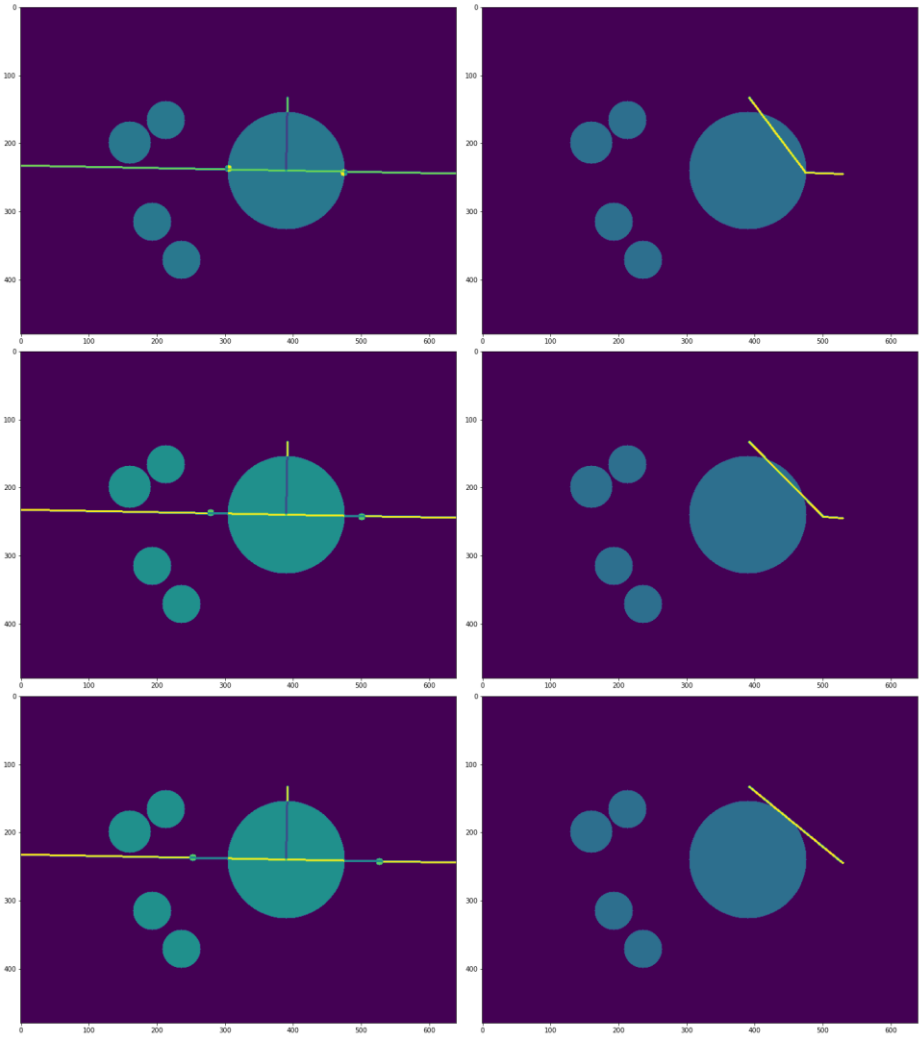

Figura 9. Representación efecto del aumento de la variable de escala de radio de seguridad EscalaRS. Fuente: elaboración propia.

En este artículo se consideró un aumento por ciclo de esta variable de 0,1, y este aumento se hará un número de veces (N) dado por el usuario, pero para este caso de estudio se establece en 10 iteraciones.

Una vez aclaradas las variables, entradas y salidas, se puede hablar de la función principal Ruta ← Planeador (Pi, Pf, EscalaRS), que se muestra en el algoritmo 2 (Tabla 2). Dentro de esta función se encuentran los procesos fundamentales para la generación de una ruta.

Para empezar, el metodo ObjChoque ← DetectarOC determina si existe un choque con algún objeto, generando 2 mascaras, una máscara con el trazado una línea recta ($\overline{PıPf}$), desde el punto inicial (Pi) al punto final (Pf), Figura 10 y otra con la ubicación de los obstáculos y sus radios de seguridad (M0). Tanto la línea como los obstáculos tienen intensidades distintas y al comparar las máscaras ($M_{0l} = M_0 - M_l$), se crea una diferencia de intensidad en los puntos que comparten la línea de principio a fin ($\overline{PıPf}$) y los obstáculos (M0). Al detectar la presencia de obstáculos, se toma como objeto de choque el objeto más cercano al punto inicial (Pi), es decir el primer objeto (ObjChoque) con el que exista una colisión, Figura 10.

Determinado el objeto, inicia la evasión del mismo con la función Mp ← Perpendiuclar (Pi,ObjChoque), obteniendo una máscara (Mp) con la información de línea (Lp) perpendicular a una línea trazada entre el punto inicial (Pi) y el centroide (Pc). La línea perpendicular (Lp ⊥ $\overline{PıPc}$) pasará por el centroide del obstáculo y se expandirá a lo largo de la imagen como se muestra en la Figura 11.

Tabla 2. Algoritmo 1 y 2

Algoritmo 1 Algoritmo EPBG

```
Entrada: Pi = (x,y)
Entrada: Pf = (x,y)
Salida: Ruta
  EscalaRS ← 1
  Choque ← verdadero
  mientras Choque==verdadero O N≤10 hacer
    Ruta ← Planeador(Pi, Pf, EscalaRS)
    Choque ← DetectarChoque(Ruta)
    EscalaRS ← EscalaRS + 0,1
    N ← N + 1
  fin mientras
```

Algoritmo 2 Método planeador

```
Entrada: Pi = (x,y)
Entrada: Pf = (x,y)
Entrada: EscalaRS
Salida: Ruta
  mientras Pi<>Pf hacer
    ObjChoque ← DetectarOC
    Mp ← Perpendicular(Pi,ObjChoque)
    Puntos ← PuntosExtremos(Mp,ObjChoque)
    Pi,Ruta ← NuevoPi(Puntos,Pf)
  fin mientras
```

Posteriormente, el método Puntos ← PuntosExtremos (Mp, ObjChoque), resuelve los puntos (Puntos ← $Obs_{\emptyset} \cap Lp$) que interceptan la circunferencia formada por el radio de seguridad ($Obs_{\emptyset}$) y la línea perpendicular ($Lp \perp \overline{PiPc}$), como se ve en la Figura 12. Una vez definidos, el último método del ciclo Pi,Ruta ← NuevoPi (Puntos,Pf), evalúa los puntos, seleccionando el que se encuentre más cerca al punto final (Pf). Este punto seleccionado se definirá como un nuevo punto inicial (Pi), para ser utilizado en la siguiente iteración y se guardara en la lista que tiene los puntos que conforman la ruta (Ruta).

Por último, en el algoritmo 1 se encuentra el método Choque ← DetectarChoque (Ruta). Este método hace una comprobación similar a la del método ObjChoque ← DetectarOC, en el algoritmo 2. Pero aquí, recibe la lista de puntos Ruta y construye una máscara (M_r) con su información, para luego compararla (M_{or}=M_{ob}-M_r) con la máscara (M_{ob}) que contiene la información de los obstáculos con sus radios de seguridad base, es decir, con el valor de EscalaRS ← 1, como se aprecia en la Figura 13. Al hacer esta operación, se crea una diferencia de intensidad en los puntos que comparten la ruta (M_r) y los objetos (M_{ob}) y si se presenta un choque, entonces la función retornará verdadero.

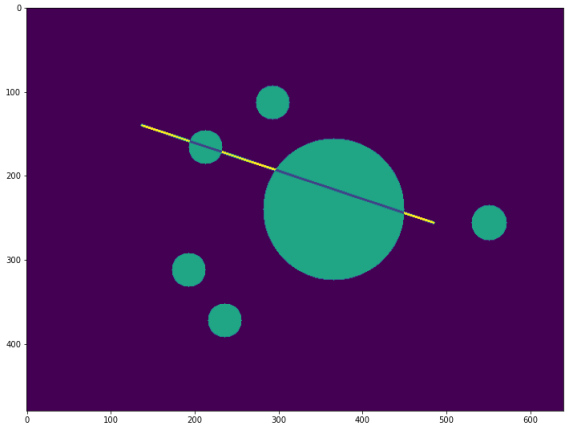

Figura 10. Detección de objeto de choque para una ruta trazada desde el punto inicial al punto final. Fuente: elaboración propia.

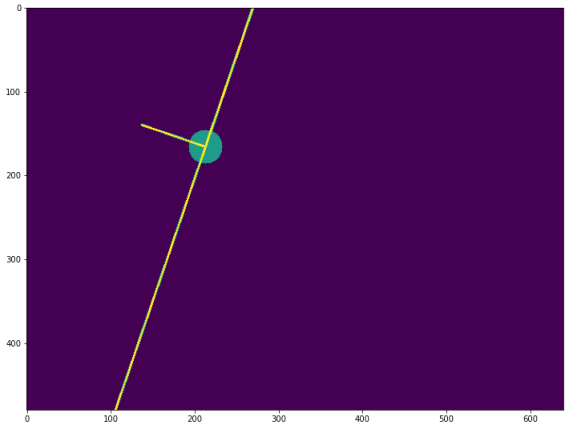

Figura 11. Líneas perpendiculares $Lp \perp \overline{PiPc}$. Fuente: elaboración propia.

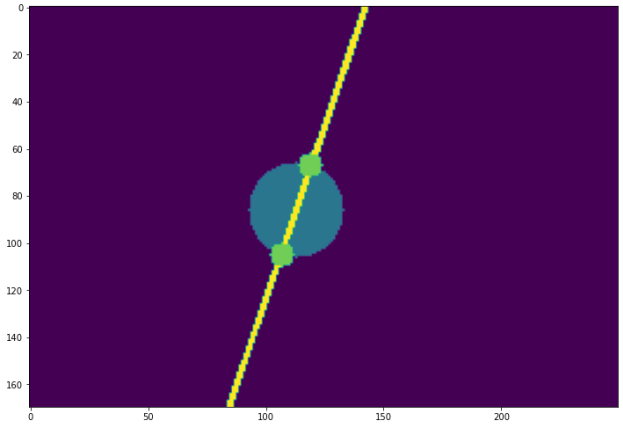

Figura 12. Puntos donde se da la intersección de la circunferencia formada por el radio de seguridad y la línea Lp. Fuente: elaboración propia.

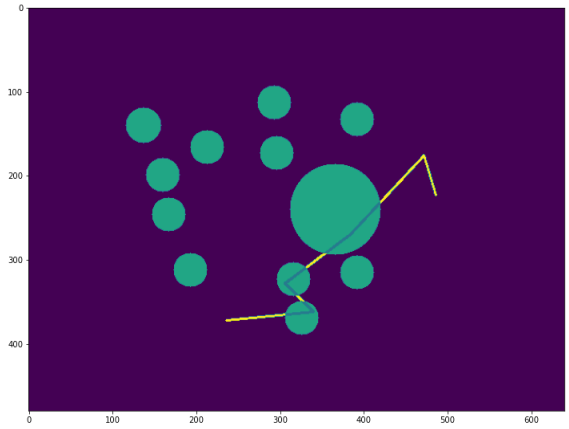

Figura 13. Detección de choque de la ruta contra obstáculos. Fuente: elaboración propia.

4. Montaje experimental

Considerando el área de interés, se propone un ambiente simulado en el software Robodk [18], basado en un sistema automatizado de una planta de producción. Este se enfoca en realizar un estudio en un escenario del mundo real, donde la información de referenciamiento espacial es captada por una cámara afín cuya matriz de calibración es la ecuación (1), con los parámetros de la Tabla 3. La información extraída a través de la cámara determinará el espacio de configuración que debe tener en cuenta la ubicación de los recipientes y herramientas comentados en la Tabla 1, el robot manipulador ABB IRB 120-3/0.6 [16], la mesa de trabajo y la banda transportadora.

$$K = \begin{bmatrix} c & cs & xH \\ 0 & c(1+m) & yH \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Tabla 3. Parámetros de la matriz de calibración de cámara 1

Parámetro 1	Valor
Constate de cámara (c)	1
Sheer (s)	0
Correción espacial eje X (xH)	320
Correción espacial eje Y (xH)	240
Distorsión lineal entre ejes (m)	-0,012431

El objetivo de este montaje es validar el algoritmo en un ejercicio de ordenamiento de objetos. Este debe establecer las rutas necesarias para recoger herramientas y recipientes de la mesa de trabajo y llevarlos a la cinta transportadora. Se evalúa el comportamiento del algoritmo cambiando la pieza a organizar.

Primero seleccionando aleatoriamente un objeto y luego dando un orden de recolección de objetos desde el más cercano al más lejano desde el punto de destino, como se muestra en la Figura 14. Finalmente, el algoritmo propuesto se evalúa junto con el algoritmo RRT para obtener una referencia de tiempo y eficiencia, Figura 15 y 16. Los experimentos fueron realizados en el lenguaje de programación Python y en un recurso computacional con un procesador AMD Ryzen 7 2700X Eight-Core 3.70 GHz y 16 GB de memoria RAM.

5. Resultados y discusiones

A través del proceso de validación del algoritmo se definió el límite de iteraciones necesarias para el aumento de la variable de radio de seguridad. También, se observó el comportamiento de este, en un ejercicio de ordenamiento con una selección de 9 objetos, en el que se observa una ventaja al realizarlo de una manera ordenada, siempre recogiendo el objeto más cercano al punto final. Además, se comprobó su complejidad computacional a través de la notación *big O* y se comparó en tiempo de ejecución con el algoritmo RRT. Por último, se validó un proceso planteado como plan B en caso tal de que el algoritmo no llegue a una solución en sus límites de iteraciones.

Inicialmente, el aumento de la variable de radio de seguridad se hace de acuerdo con las veces que el algoritmo intenta encontrar una solución sin choques presentes en la ruta y debe ser limitado en el número de iteraciones en las que se realiza este incremento. Este parámetro, definido en el algoritmo 1 como N, existe debido a que en ocasiones el algoritmo, puede no encontrar una solución y se define con el fin de que se ejecute la segunda variante del algoritmo que utiliza los segundos puntos extremos en vez de entrar en un bucle infinito.

Otra razón por la cual, se debe limitar este valor es que se evite que el algoritmo, encuentre soluciones que tengan conflicto con otros objetos y con el espacio de trabajo del robot, como se ve en la Figura 17. Se observó que, con un número mínimo de 10 iteraciones es suficiente para que se encuentre una solución en el 92% de las situaciones y se recomienda que, este parámetro no exceda un valor mayor a 10 iteraciones, ya que los radios de seguridad serían muy grandes y las soluciones encontradas excederían los límites físicos del robot manipulador. Por lo que se refiere al ejercicio de ordenamiento de objetos, se observó su comportamiento de en 2 casos de estudio ubicando sobre la mesa de trabajo 9 objetos y el robot manipulador.

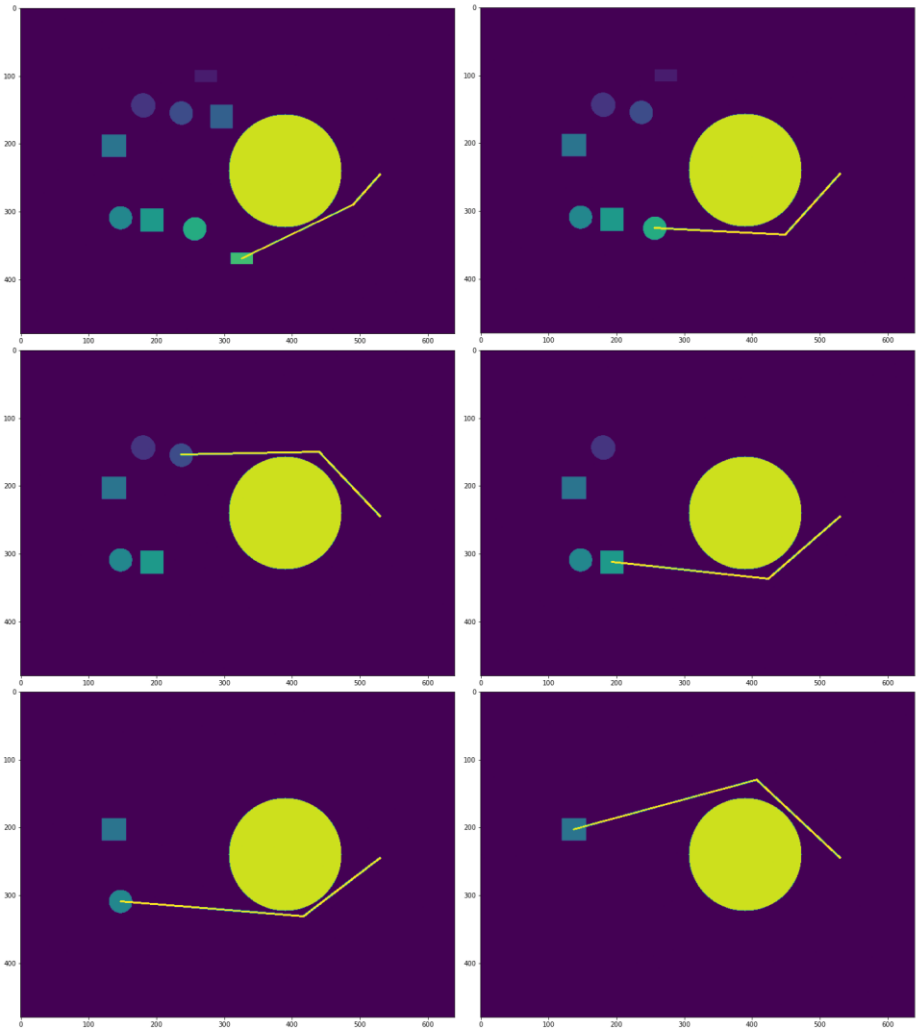

Figura 14. Secuencia de recolección de objetos, basada en el algoritmo EPGB. Fuente: elaboración propia.

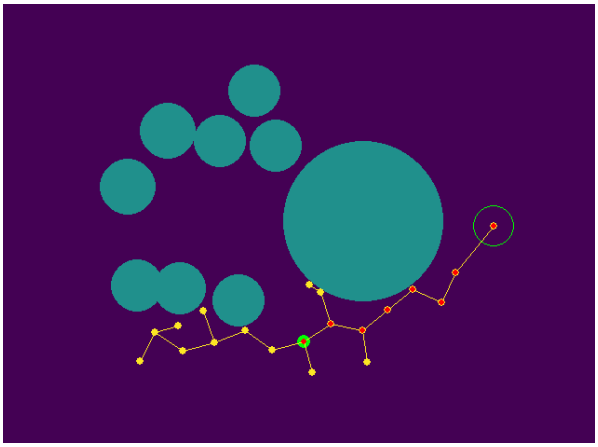

Figura 15. Representación del algoritmo RRT encontrando una ruta. Fuente: elaboración propia.

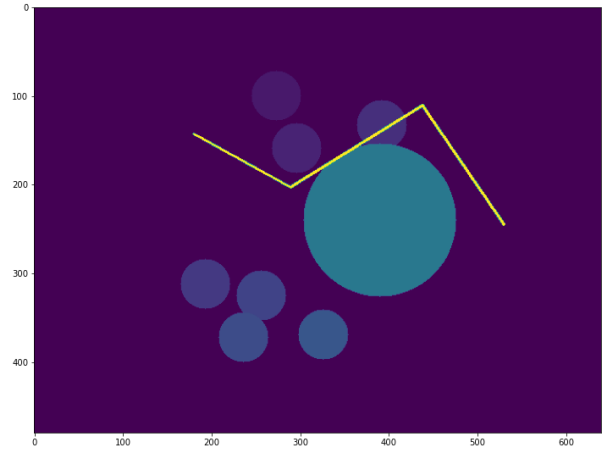

Figura 16. Fallo al encontrar una ruta en el algoritmo EPBG. Fuente: elaboración propia.

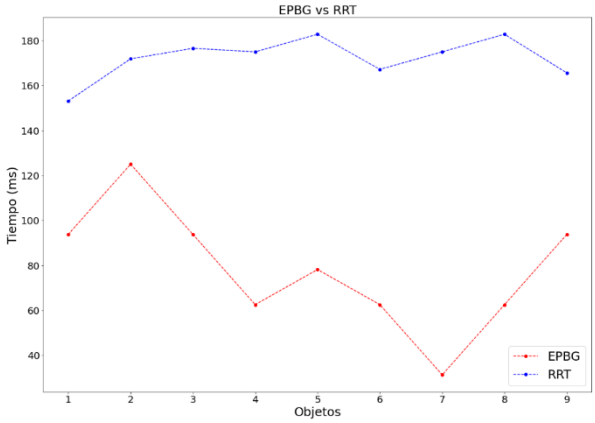

Figura 17. Tiempos de ejecución de EPBG vs RRT. Fuente: elaboración propia.

Por lo que se refiere al ejercicio de ordenamiento de objetos, se observó su comportamiento de en 2 casos de estudio ubicando sobre la mesa de trabajo 9 objetos y el robot manipulador. Para el primer caso, se deciden recoger los objetos de una manera ordenada y en el segundo se seleccionan al azar. Durante la implementación del algoritmo, se observó que cuanto más hay obstáculos entre el punto inicial y el punto final, más iteraciones son necesarias para encontrar una solución. Dado que, el método de planeador se ejecutará por cada obstáculo que se encuentre entre ellos. Por esto, hacer un recogimiento de los objetos de forma en la que, siempre se recoja el objeto más cercano al punto final, como se ve en la Figura 14, hace que las trayectorias sean más cortas y que los ciclos de iteraciones se mantengan al mínimo, entre 4 y 5 iteraciones. Además, el algoritmo encuentra de manera eficiente las soluciones, tomando para cada objeto un tiempo no mayor a los 100 ms, Tabla 4 y 5.

Tabla 4. Complejidad computacional para algoritmos RRT, PRM y algunas variantes

Algoritmo	Complejidad
PRM	O(n)
Sprm	$O(n^2)$
k-sPRM	O(n)
RRT	O(n)
PRM*	O(nlog(n))
k-PRM*	O(nlog(n))
RRG	O(nlog(n))
K-RRG	O(nlog(n))
RRT*	O(n)
K-RRT*	O(n)

Fuente: Sampling-based algorithms for optimal motion planning [19].

Tabla 5. Tiempos promedios tomados con 10 muestras por objeto usando el algoritmo RRT y algoritmo EPBG

Objeto	Tiempo(ms)	Tiempo(ms)
1	153,125	93,75
2	171,875	125
3	176,5625	93,75
4	175	62,5
5	182,8125	78,125
6	167,1875	62,5
7	175	31,25
8	182,8125	62,5
9	165,625	93,75

Fuente: elaboración propia.

En cuanto a la complejidad computacional del algoritmo, se han identificado dos ciclos anidados en su construcción.

El primer ciclo se encuentra en el método llamado "planeador" y se encarga de evadir cada obstáculo presente entre el punto inicial y final. Este ciclo tiene una complejidad computacional lineal, ya que su tiempo de ejecución depende del número de obstáculos presentes. Utilizando la notación *big O*, podemos decir que su complejidad es del orden O(N), donde N es el número de obstáculos.

El segundo ciclo es aquel que itera el aumento de los radios de seguridad. Sin embargo, es importante tener en cuenta que este ciclo tiene un límite físico impuesto por el robot manipulador, lo que significa que en el caso específico analizado no puede iterar más de 10 veces. En consecuencia, su complejidad computacional es constante y se puede representar como O(1) para este escenario particular.

Sin embargo, es relevante destacar que en otros casos donde no exista ese límite físico impuesto por el robot, es decir, si se utiliza el algoritmo para una situación diferente, es posible que el segundo ciclo se comporte de manera diferente y su complejidad computacional pueda ser lineal, representada como O(M). Por lo tanto, al analizar la complejidad de un algoritmo, es crucial considerar las limitaciones y condiciones específicas de la situación en la que se aplica.

Teniendo en cuenta estos análisis, la complejidad general del algoritmo se obtiene multiplicando la complejidad de ambos ciclos. Por lo tanto, la complejidad computacional del algoritmo es del orden O(N * 1), que se simplifica a O(N). Esto significa que el tiempo de ejecución del algoritmo aumenta linealmente con el número de

obstáculos presentes. En comparación con algoritmos como RRT, PRM y sus variantes, este algoritmo tiene una complejidad similar como se ve en la Tabla 4.

En términos de tiempo de procesamiento, se seleccionó un algoritmo con la misma complejidad computacional como lo es RRT. Realizando la comparación, tomando los tiempos que tarda cada algoritmo para encontrar una ruta desde un punto inicial a un punto final para cada objeto, evadiendo obstáculos y tomando primero los objetos más cercanos al punto final. Se tomó una muestra de 10 pruebas para cada objeto y se sacó un promedio con los valores de tiempo ejecutado, tanto para RRT como para EPBG, como se observa en la Tabla 5 y la Figura 18. Esto teniendo en cuenta que el algoritmo RRT crea sus rutas mediante un muestreo aleatorio [12]. Al final se evidenció que para todos los objetos el algoritmo EPBG presenta mejor rendimiento, presentando mejoras entre 38% y 48%.

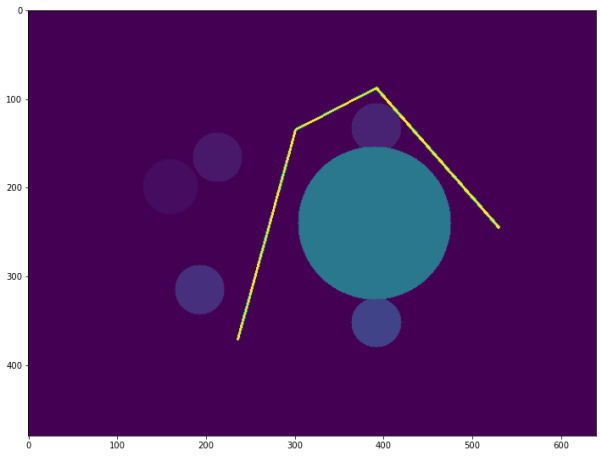

Figura 18. Solución encontrada utilizando el segundo punto de intersección entre la circunferencia del radio de seguridad y la línea de referencia.
Fuente: elaboración propia.

Con respecto a la posibilidad de que no se pueda encontrar una solución, se implementó una estrategia en el algoritmo para abordar este escenario. Después de completar el ciclo de iteraciones sin encontrar una solución, se decidió explorar una alternativa seleccionando el segundo punto más cercano al punto final. Este segundo punto se encuentra en la intersección entre la circunferencia formada por el radio de seguridad y la línea de referencia lp Figura 12. Sin embargo, se observó que las rutas generadas utilizando esta variante resultaban en caminos más largos, como se evidencia en la Figura 18. Esto implica que esta variante es menos eficiente tanto en la ejecución de las rutas debido a su mayor longitud, como para el propio algoritmo. Es importante destacar que esta variante del algoritmo puede presentar limitaciones. En ocasiones, el número de iteraciones necesarias para aumentar la variable de radio de seguridad y encontrar una solución puede hacer que se excedan los límites físicos del robot, lo que impide encontrar una ruta viable para su implementación.

Tomando en cuenta estas consideraciones, se establece que un valor de 10 iteraciones es más que suficiente para determinar la existencia o no de una solución aplicable utilizando esta variante.

6. Conclusiones

Basándose en el estudio de algoritmos de planeación de movimiento, se presenta el algoritmo de enrutado polilineal basado en geometría (EPBG). Mostrando una solución basada en el ordenamiento de objetos y buscando encontrar rutas eficientes para que puedan ser ejecutadas en un robot manipulador, teniendo en cuenta sus límites físicos.

Teniendo en cuenta las observaciones de los resultados obtenidos, es recomendable que se establezca en un valor de 10, en los límites de las iteraciones que aumentan del parámetro de radio de seguridad (EscalaRS), debido a que puede encontrar soluciones que no respeten el espacio de trabajo del robot en caso tal de que el valor de este parámetro sea más alto. Se planteó una segunda variante para que se ejecutara pasadas 10 iteraciones, si no encuentra una solución. Cambiando la selección al segundo punto de intersección de la línea de referencia con el radio de seguridad de los obstáculos, si bien puede encontrar soluciones a la trayectoria, se deja como un opcional y se debe ver como un último recurso, dado que las soluciones que encuentra por este método son generalmente caminos más largos y poco eficientes.

Observando los resultados obtenidos, se ve que el algoritmo de enrutado polilineal basado en geometría (EPBG), es bastante eficiente resolviendo la problemática planteada de recogimiento de objetos con un robot manipulador. Comparado con el algoritmo RRT, se pueden ver mejoras de entre 38% y 48%. Además de presentar un camino más suave y fácil de ejecutar, sin necesidad de tener que pasar por un post procesamiento para aliviar los cambios bruscos que se puedan presentar en la ruta entregada. Sin embargo, es de aclarar que el algoritmo propuesto no ha sido utilizado para otros ejercicios de planeación de movimiento u otras problemáticas, en las que el algoritmo RRT puede llegar a encontrar soluciones más eficientes, ya que es un algoritmo planteado para soluciones generales en distintos tipos de problemáticas.

A través de la fase experimental, se ha observado que la ordenación de la recogida de objetos, siguiendo un orden que va desde el objeto más cercano al más lejano con respecto a un punto de referencia final (Pf), mejora el rendimiento en la planificación de rutas tanto en términos de tiempo como de suavidad. Esto se debe a que, al organizar la secuencia de recogida de esta manera, se reduce la probabilidad de que ocurran conflictos entre obstáculos cuando se encuentran muy cerca, lo que implicaría un aumento en las operaciones realizadas por el algoritmo. Establecer un orden de recogida ayuda a mitigar esta situación y a hacer el proceso más eficiente.

Basados en el proceso de obtención para el espacio de configuración, con el uso de la herramienta js-anotator [17], se generó un dataset con las etiquetas de 2000 imágenes/máscaras [20]. El trabajo actual propone como trabajo futuro la aplicación de técnicas relacionadas con el aprendizaje reforzado, tomando como ventaja de la disponibilidad y factible referencia para el cálculo de coordenadas, forma, y área. También, probar más ejercicios de planeación de movimiento o problemáticas para comprobar su robustez.

Financiación

No aplica.

Contribuciones de los autores

P. A. Montaño-Herrera: Conceptualización, Investigación; Redacción: revisión y edición. J. P. Sosa-Esquivel: Investigación, Curación de datos; Análisis formal, borrador original. M. A. Jinete-Gómez: Investigación; Validación; Visualización; Escritura – borrador original.

Todos los autores han leído y aceptado la versión publicada del manuscrito.

Conflictos de interés

Los autores declaran no tener conflicto de intereses.

Declaración de la Junta de Revisión Institucional

No aplica.

Declaración de consentimiento informado

No aplica.

Referencias

[1] E. Barleta, G. Pérez, R. Sánchez, “La revolución industrial 4.0 y el advenimiento de una logística 4.0,” p. 15, 2020.

[2] L. Y. Becerra Sánchez, “Tecnologías de la información y las Comunicaciones en la era de la cuarta revolución industrial: Tendencias Tecnológicas y desafíos en la educación en Ingeniería,” *Entre Cienc. e Ing.*, vol. 14, no. 28, pp. 76–81, 2020, doi: https://doi.org/10.31908/19098367.2057

[3] S. M. Ross, *Simulation. Academic Press*, 2022.

[4] R. S. Kenett, G. Vicario, “Challenges and Opportunities in Simulations and Computer Experiments in Industrial Statistics: An Industry 4.0 Perspective,” *Adv. Theory Simulations*, vol. 4, no. 2, p. 2000254, 2021, doi: https://doi.org/10.1002/adts.202000254

[5] López de Lacalle, Posada, “Special Issue on New Industry 4.0 Advances in Industrial IoT and Visual Computing for Manufacturing Processes,” *Appl. Sci.*, vol. 9, no. 20, p. 4323, 2019, doi: https://doi.org/10.3390/app9204323

[6] A. El Wahabi, I. H. Baraka, S. Hamdoune, and K. El Mokhtari, “Detection and Control System for Automotive Products Applications by Artificial Vision Using Deep Learning,” *Advanced Intelligent Systems for Sustainable Development (AI2SD’2019)*, 2020, pp. 224–241. doi: https://doi.org/10.1007/978-3-030-36671-1_20

[7] A. N. Abbas, M. A. Irshad, H. H. Ammar, “Experimental Analysis of Trajectory Control Using Computer Vision and Artificial Intelligence for Autonomous Vehicles” *arXiv - CS - Artificial Intelligence*, 2021.

[8] H. Zhang, W. Lin, A. Chen, “Path Planning for the Mobile Robot: A Review,” *Symmetry (Basel).*, vol. 10, no. 10, p. 450, 2018, doi: https://doi.org/10.3390/sym10100450

[9] S. M. Alcántara Tacora, E. D. López Zapata, J. Peralta Toribio, R. R. Rodríguez Bustinza, “Planificación de movimiento mediante algoritmo de campos potenciales con optimización de parámetros aplicado a un manipulador antropomórfico de 6 GLD,” *TECNIA*, vol. 21, no. 2, 2021, doi: https://doi.org/10.21754/tecnia.v21i2.848

[10] J. Zhong, T. Wang, L. Cheng, "Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics," *Complex Intell. Syst.*, vol. 8, no. 3, pp. 1899–1912, 2022, doi: https://doi.org/10.1007/s40747-021-00366-1

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996, doi: https://doi.org/10.1109/70.508439

[12] S. M. LaValle, "Rapidly-exploring random trees: a new tool for path planning," *Annu. Res. Rep.*, 1998.

[13] R. K. Jenamani, R. Kumar, P. Mall, K. Kedia, "Robotic Motion Planning using Learned Critical Sources and Local Sampling," *2020 IEEE International Conference on Robotics and Automation*, doi: https://doi.org/10.48550/arXiv.2006.04194

[14] H. Zhang, Y. Wang, J. Zheng, J. Yu, "Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018, doi: https://doi.org/10.1109/ACCESS.2018.2871222

[15] X. Cao, X. Zou, C. Jia, M. Chen, Z. Zeng, "RRT-based path planning for an intelligent litchi-picking manipulator," *Comput. Electron. Agric.*, vol. 156, pp. 105–118, 2019, doi: https://doi.org/10.1016/j.compag.2018.10.031

[16] S. Chakraborty, P. S. Aithal, "ABB IRB 120-30.6 Build Procedure In RoboDK," *Int. J. Manag. Technol. Soc. Sci.*, vol. 6, no. 2, pp. 256–264, 2021, doi: https://doi.org/10.5281/zenodo.5782759

[17] P. Tangseng, Z. Wu, K. Yamaguchi, "Looking at Outfit to Parse Clothing," *Computer Vision and Pattern Recognition*, 2017, doi: https://doi.org/10.48550/arXiv.1703.01386

[18] A. Garbev, A. Atanassov, "Comparative Analysis of RoboDK and Robot Operating System for Solving Diagnostics Tasks in Off-Line Programming," *in 2020 International Conference Automatics and Informatics (ICAI)*, 2020, pp. 1–5. doi: https://doi.org/10.1109/ICAI50593.2020.9311332

[19] S. Karaman, E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011, doi: https://doi.org/10.1177/0278364911406761

[20] W. A. R. Ruiz, P. A. M. Herrera, J. P. S. Esquivel, "Tools in industry (ROBODK)," *Zenodo*, 2021. doi: https://doi.org/10.5281/zenodo.5768611