



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

Geometry-based polylinear routing for motion planning in the object sorting

Pedro Alejandro Montaña-Herrera¹, Juan Pablo Sosa-Esquivel², Marco Antonio Jinete-Gómez³

¹Semillero de investigación Open-Creator, Facultad de ingenierías, Universidad Piloto de Colombia, Colombia. Orcid: 0000-0003-1060-1523¹, 0000-0001-9725-9059³. Correo electrónico: pedro-montano@upc.edu.co, juan-sosa@upc.edu.co, marco-jinete@upc.edu.co

Recibido: fecha recibido. Aceptado: fecha aceptado. Versión final: fecha versión final.

Resumen:

Este trabajo propone el método de enrutado polilíneal basado en geometría como una solución a la planeación de movimiento en ejercicios de clasificación de objetos para procesos de manufactura. Este algoritmo se basa en propiedades geométricas que surgen entre la interacción de los objetos dentro del espacio de configuración. El método propuesto, evidencia en el montaje experimental, la generación de rutas suaves, con tiempos entre los 62,5 ms y los 125 ms en un ordenador con procesador AMD Ryzen 7 2700X Eight-Core 3.70 GHz y 16 GB de memoria RAM, comparado con el algoritmo RRT se ve una mayor eficiencia entre 38% y 48%, existiendo una reducción en procesos iterativos, que se ven reflejados en un menor tiempo de respuesta, por lo tanto, el método planteado es una solución viable para resolver escenarios de planeación de movimiento dentro del ejercicio de ordenamiento de objetos.

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

Abstract:

This work proposes the geometry-based polylinear routing method as a solution to motion planning in object classification exercises for manufacturing processes. This algorithm is based on geometric properties that arise from the interaction of the objects within the configuration space. The proposed method generated smooth routes with times ranging from 62.5 ms to 125 ms on a computer equipped with an AMD Ryzen 7 2700X Eight-Core 3.70 GHz processor and 16 GB of RAM, as demonstrated in the experimental setup. With the RRT algorithm, a greater efficiency is seen between 38% and 48%. It can be shown that there is a reduction in iterative processes, which is reflected in a shorter response time. Therefore, the proposed method is a viable solution to solve movement planning scenarios within the object sorting exercise.

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

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 artificial e inteligencia artificial [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 a día de hoy. 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 Source - 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 120-30.6 [16] se ubicara 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 el cómo se obtiene el espacio de configuración para el requerimiento de ordenamiento de objetos. La sección de algoritmo propuesto, donde se explica a profundidad los métodos principales que desarrollan el algoritmo. La sección montaje experimental, que muestra el montaje realizado para la evaluación del algoritmo. Resultados y discusiones, donde se compara como fue el rendimiento del algoritmo con respecto a RRT y se habla 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 con una única cámara ubicada sobre la mesa de trabajo y el robot, como se muestra en las figuras 1, 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.

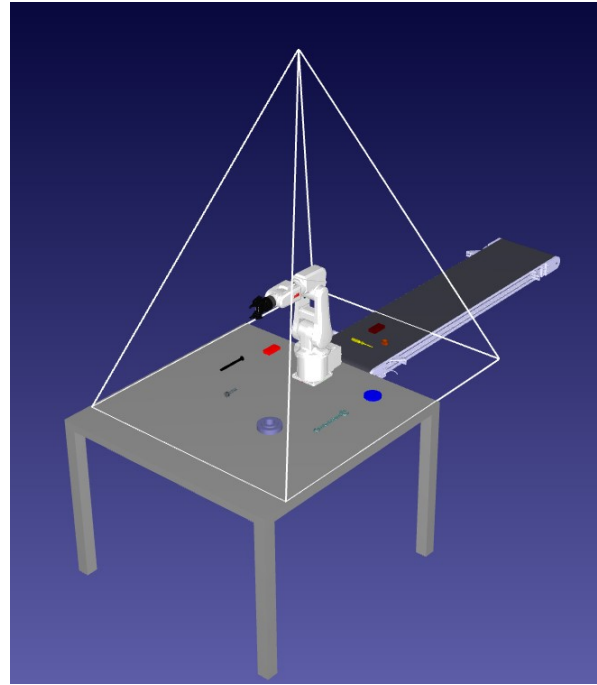


Figura 1. Imagen de representación de ubicación de la cámara respecto a la escena.

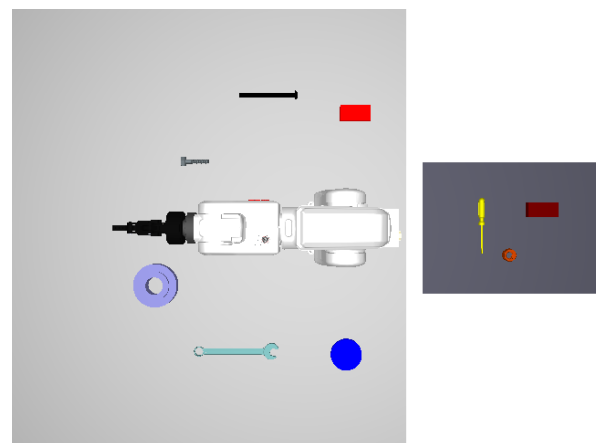


Figura 2. Vista superior tomada por la cámara afín.

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 resolu-

ción 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 máscaras 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.

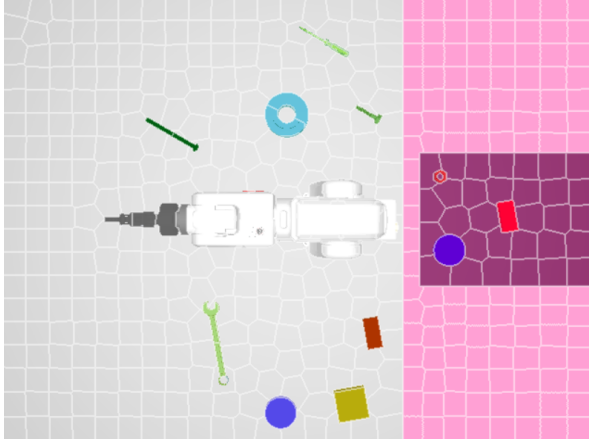


Figura 3. Herramienta de anotación *js segment annotator* con la cuadrícula de *SLIC super pixels*.

Tabla 1. Tabla de clases presentes en la línea de producción.

Tabla de clases	
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

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. Para esto se define un área cuadrada

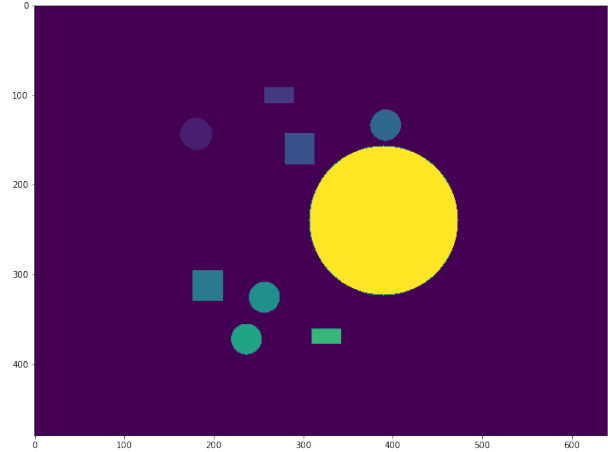


Figura 4. Mascara de salida del anotador.

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.

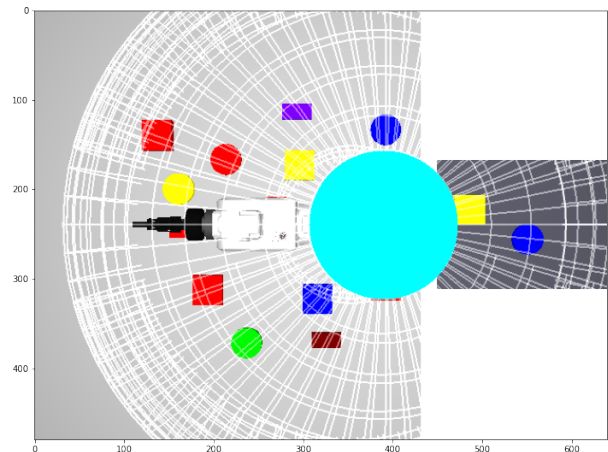


Figura 5. Espacio de trabajo del robot. El límite inferior representado por el color cian y los límites exteriores por la malla esférica alrededor.

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

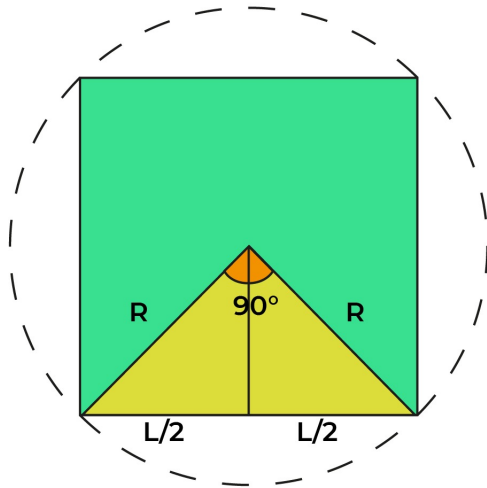


Figura 6. Representación de la relación de un cuadrado inscrito en una circunferencia

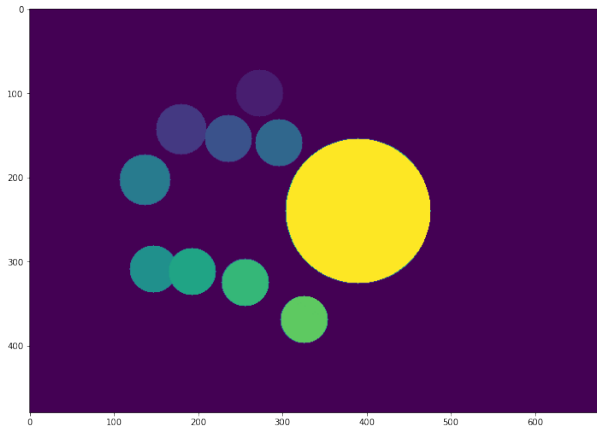


Figura 7. Mascará de representación del espacio de configuración, con los radios de seguridad definidos para cada objeto.

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 (P_i) y final (P_f) 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 (P_i), hasta que dejen de existir colisiones y se llegue a una trayectoria continua hasta el punto final (P_f).

Para comprender el algoritmo 1 (EPBG), se deben

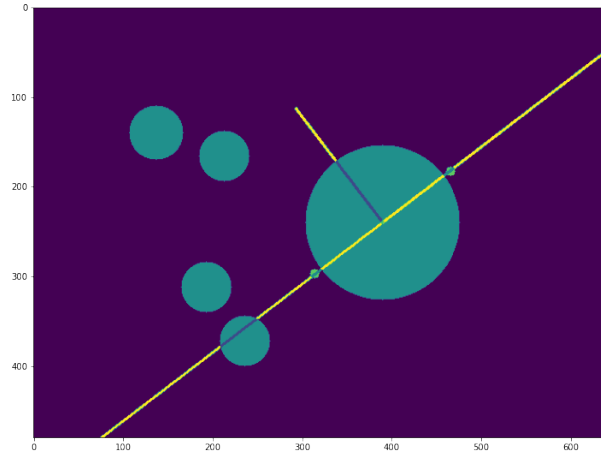


Figura 8. Representación de la líneas formadas a través de interacciones en el espacio de configuración.

definir todos los parámetros y variables que interactúan dentro de él. Primero, el algoritmo recibe como entrada el punto inicial (P_i), el objetivo final a alcanzar (P_f) 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. 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.

Algoritmo 1 Algoritmo EPBG

Entrada: $P_i = (x, y)$

Entrada: $P_f = (x, y)$

Salida: *Ruta*

EscalaRS \leftarrow 1

Choque \leftarrow verdadero

mientras *Choque* == verdadero **O** $N \leq 10$ **hacer**

Ruta \leftarrow Planeador($P_i, P_f, EscalaRS$)

Choque \leftarrow DetectarChoque(*Ruta*)

EscalaRS \leftarrow *EscalaRS* + 0,1

$N \leftarrow N + 1$

fin mientras

Una vez aclaradas las variables, entradas y sa-

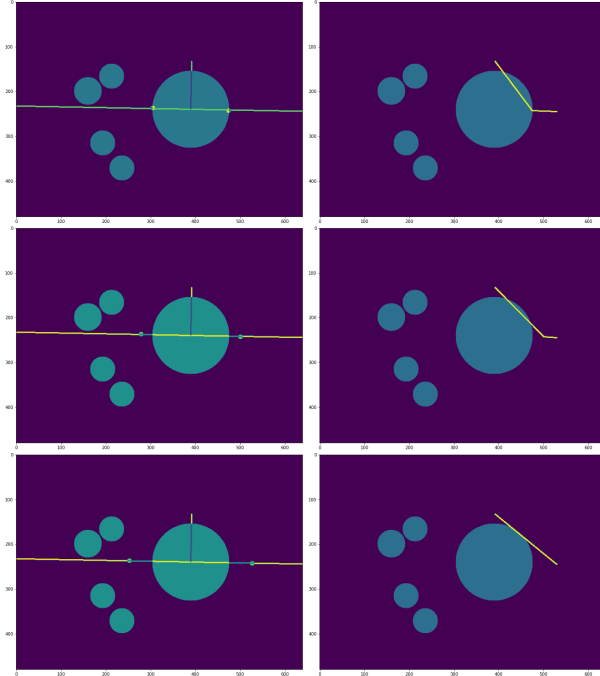


Figura 9. Representación efecto del aumento de la variable de escala de radio de seguridad $EscalaRS$.

lidas, se puede hablar de la función principal $Ruta \leftarrow Planeador(Pi, Pf, EscalaRS)$, que se muestra en el algoritmo 2. Dentro de esta función se encuentran los procesos fundamentales para la generación de una ruta.

Para empezar, el método $ObjChoque \leftarrow DetectarOC$ determina si existe un choque con algún objeto, generando 2 máscaras, una máscara con el trazado de una línea recta ($PiPf$), desde el punto inicial (Pi) al punto final (Pf), figura 10, y otra con la ubicación de los obstáculos y sus radios de seguridad (M_o). Tanto la línea como los obstáculos tienen intensidades distintas y al comparar las máscaras ($M_{ol} = M_o - M_l$), se crea una diferencia de intensidad en los puntos que comparten la línea de principio a fin ($PiPf$) y los obstáculos (M_o). 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 $M_p \leftarrow Perpendicular(Pi, ObjChoque)$, obteniendo una máscara (M_p) con la información de línea (Lp) perpendicular a una línea trazada entre el punto inicial (Pi) y el centroide (Pc). La línea

perpendicular ($Lp \perp \overline{PiPc}$) pasará por el centroide del obstáculo y se expandirá a lo largo de la imagen como se muestra en la figura 11.

Posteriormente, el método $Puntos \leftarrow PuntosExtremos(M_p, ObjChoque)$, resuelve los puntos ($Puntos \leftarrow Obs_{\varnothing} \cap Lp$) que interceptan la circunferencia formada por el radio de seguridad (Obs_{\varnothing}) y la línea perpendicular ($Lp \perp \overline{PiPc}$), como se ve en la figura 12. Una vez definidos, el último método del ciclo $Pi, Ruta \leftarrow NuevoPi(Puntos, Pf)$, 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 ($Ruta$).

Algoritmo 2 Método planeador

Entrada: $Pi = (x, y)$

Entrada: $Pf = (x, y)$

Entrada: $EscalaRS$

Salida: $Ruta$

mientras $Pi \langle \rangle Pf$ **hacer**

$ObjChoque \leftarrow DetectarOC$

$M_p \leftarrow Perpendicular(Pi, ObjChoque)$

$Puntos \leftarrow PuntosExtremos(M_p, ObjChoque)$

$Pi, Ruta \leftarrow NuevoPi(Puntos, Pf)$

fin mientras

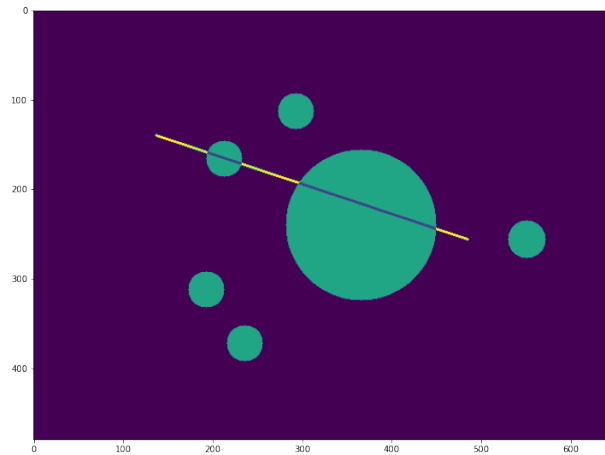


Figura 10. Detección de objeto de choque para una ruta trazada desde el punto inicial al punto final.

Por último, en el algoritmo 1 se encuentra el método $Choque \leftarrow DetectarChoque(Ruta)$. Este método hace una comprobación similar a la del método $ObjChoque \leftarrow DetectarOC$, en el algoritmo 2. Pero aquí, recibe la lista de puntos $Ruta$ y construye una

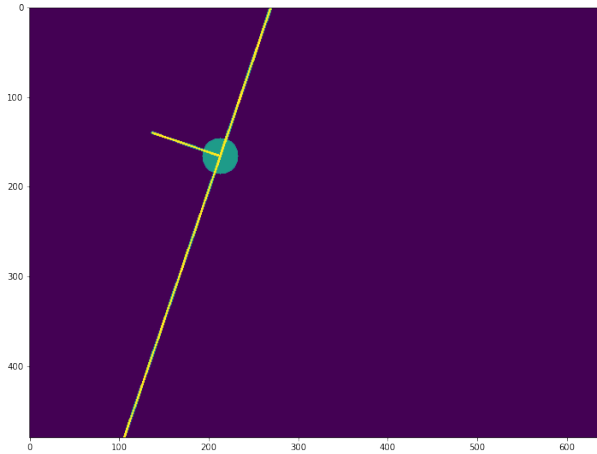


Figura 11. Líneas perpendiculares $L_p \perp \overline{PiPc}$.

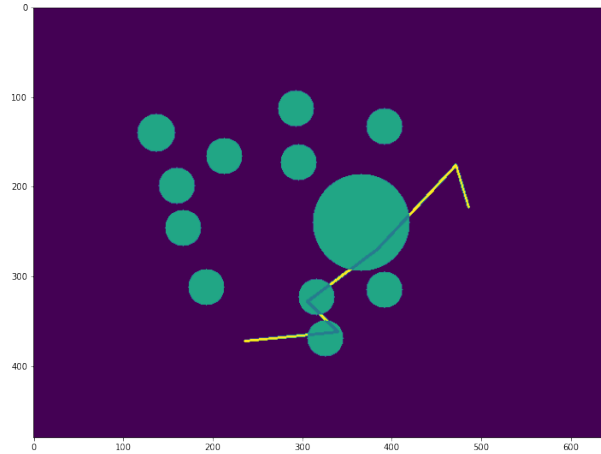


Figura 13. Detección de choque de la ruta contra obstáculos.

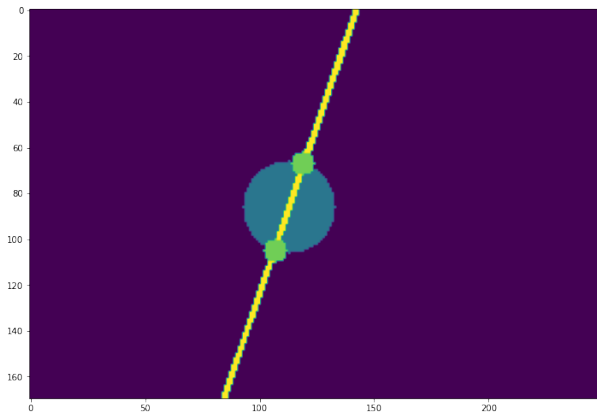


Figura 12. Puntos donde se da la intersección de la circunferencia formada por el radio de seguridad y la línea L_p .

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 $EscalaRS \leftarrow 1$, 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.

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 2. 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 120-30.6 [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 2. Tabla de parámetros de la matriz de calibración de cámara 1.

Tabla de parámetros	
Parámetro 1	Valor
Constante de cámara (c)	1
Sheer (s)	0
Corrección espacial eje X (xH)	320
Corrección espacial eje Y (yH)	240
Distorsión lineal entre ejes (m)	-0,012431

Fuente: elaboración propia

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 al-

goritmo RRT para obtener una referencia de tiempo y eficiencia, figuras 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.

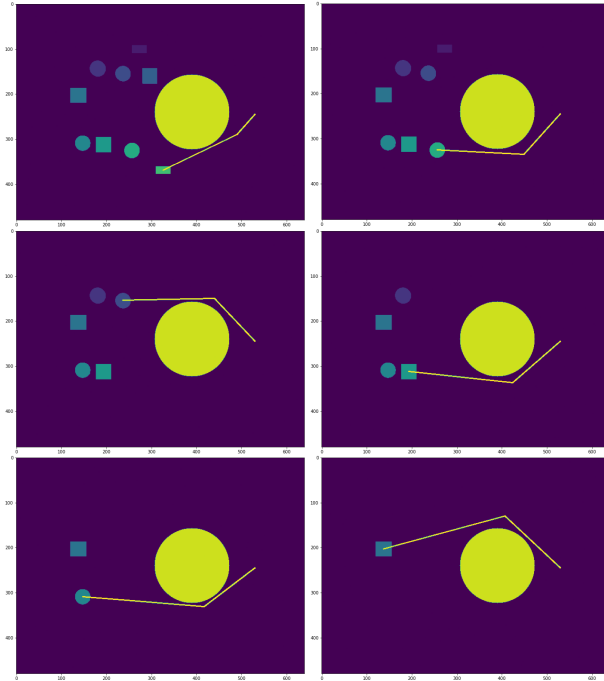


Figura 14. Secuencia de recolección de objetos, basada en el algoritmo EPBG.

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 a las veces que el algoritmo intenta encontrar una solución sin choques

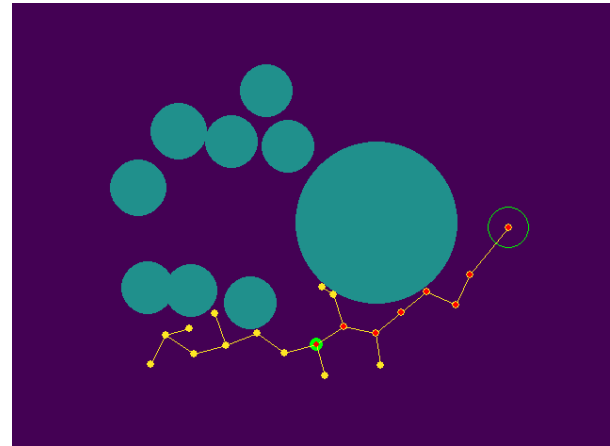


Figura 15. Representación del algoritmo RRT encontrando una ruta.

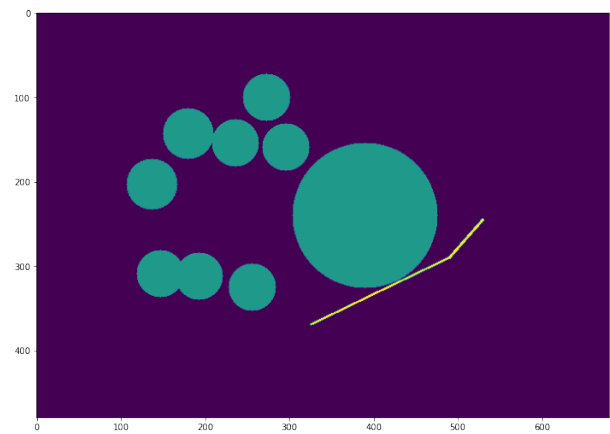


Figura 16. Representación del algoritmo EPBG encontrando una ruta.

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 en un muestreo de 1800 ordenamientos. 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.

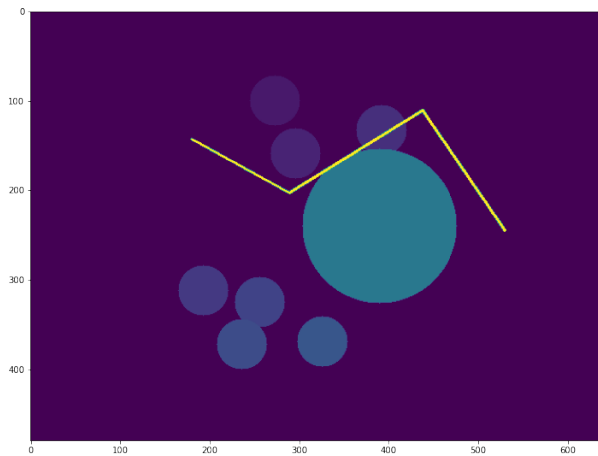


Figura 17. Fallo al encontrar una ruta en el algoritmo EPBG.

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 ejecutara 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.

En cuanto a la complejidad computacional del algoritmo, este posee 2 ciclos anidados en su construcción, uno de los ciclos, ubicado en el método llamado planeador, se encarga de evadir cada obstáculo presente entre el punto inicial y final, es decir que su complejidad computacional sigue un comportamiento lineal de acuerdo al número de obstáculos que se interpongan en la solución, Utilizando la notación big O sería del orden $O(n)$. El segundo ciclo es el que se encarga de iterar el aumento a los radios de seguridad, sin embargo, este tiene un límite físico dado por el robot manipulador,

impidiendo que este ciclo se comporte de manera lineal, ya que, no podrá iterar a más allá de 10 veces haciendo que, su complejidad computacional según la notación big O sea de $O(1)$. Esto da como resultado que el algoritmo obedezca de forma general una complejidad computacional de $O(n)$, estando a la par de algoritmos como RRT, PRM y algunas de sus variantes 3.

Tabla 3. Tabla de complejidad computacional para algoritmos RRT, PRM y algunas variantes.

Complejidad computacional	
Algoritmo	Complejidad
PRM	$O(n)$
sPRM	$O(n^2)$
k-sPRM	$O(n)$
RRT	$O(n)$
PRM*	$O(n \log(n))$
k-PRM*	$O(n \log(n))$
RRG	$O(n \log(n))$
k-RRG	$O(n \log(n))$
RRT*	$O(n)$
k-RRT*	$O(n)$

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

Tabla 4. Tabla de tiempos promedios tomados con 10 muestras por objeto usando el algoritmo RRT y algoritmo EPBG.

Tiempos promedio encontrando rutas		
Objeto	tiempo(ms) RRT	tiempo(ms) EPBG
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 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 4 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 %.

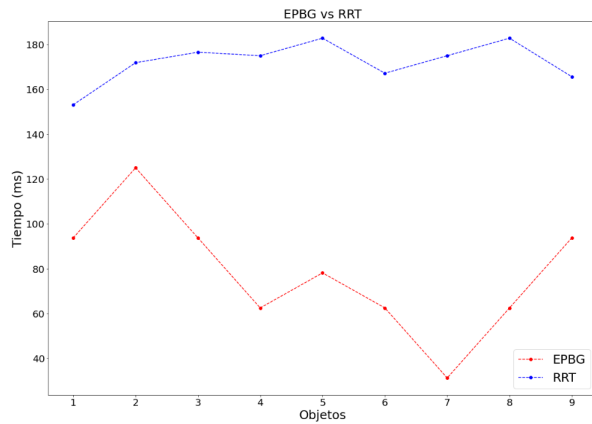


Figura 18. Tiempos de ejecución de EPBG vs RRT.

Con respecto a la posibilidad de que no se pueda encontrar una solución, siempre seleccionando el punto extremo más cercano al punto final, después de haber hecho el ciclo de iteraciones. Se realizó un caso de estudio, donde el algoritmo intentó una solución con el segundo punto, que está ubicado en la intersección entre la circunferencia formada por el radio de seguridad y la línea de referencia L_p , figura 12. Las rutas encontradas por este método dan como resultado caminos más largos como se puede apreciar en la figura 19. Esto representa que esta variante sea menos eficiente, tanto para la ejecución de sus rutas debido a la mayor longitud como para el propio algoritmo. Debido a que por este método, el número de iteraciones necesarias para aumentar la variable de radio de seguridad y que se encuentre una solución, en ocasiones hace que se excedan los límites físicos del robot, no encontrando así una ruta viable para implementar. Por esto se establece que con un valor de 10 iteraciones es más que suficiente para determinar si existe o no una solución aplicable por esta variante.

6 Conclusiones

Basándose en el estudio de algoritmos de planeación de movimiento, se presenta el algoritmo de enrutado

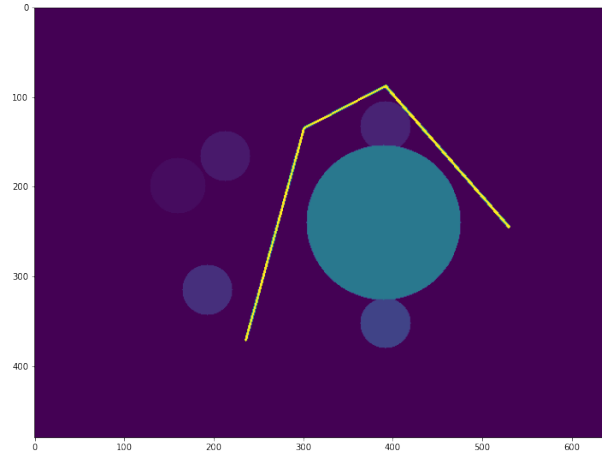


Figura 19. Solución encontrada utilizando la variante opcional del algoritmo EPBG.

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, si este valor es más alto. También, se planteó una segunda variante para que se ejecutara pasadas 10 iteraciones al no encontrar una solución. Cambiando la selección al segundo punto de intersección de la línea de referencia con el radio de seguridad de los obstáculos, que se ven en la figura 12. Si bien, esta variante puede encontrar soluciones a la trayectoria, se deja como un opcional y se debe ver como un último recurso, dado que las soluciones encontradas 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 observa como el organizar la recogida de los objetos siempre yendo del más cercano al más lejano del punto final de referencia (P_f), presenta un mejor rendimiento en la planificación de rutas tanto en el tiempo como en la suavidad de las mismas. Considerando que se pueden presentar conflictos entre obstáculos si se encuentran demasiado cerca haciendo que el algoritmo tenga que realizar más operaciones, situación que se hace menos frecuente estableciendo un orden de recogida haciéndolo más eficiente.

Basados en el proceso de obtención para el espacio de configuración, con el uso de la herramienta js-annotator [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.

Referencias

- [1] E. Barleta, G. Pérez, y R. Sánchez, “La revolución industrial 4.0 y el advenimiento de una logística 4.0”, *CEPAL*, 2020.
- [2] L. Y. Becerra, “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 Ciencia e Ingeniería*, vol. 14, pp. 76 – 81, 12 2020. DOI: <https://doi.org/10.31908/19098367.2057>.
- [3] S. M. Ross, *Simulation*. academic press, 2022.
- [4] R. S. Kenett y G. Vicario, “Challenges and opportunities in simulations and computer experiments in industrial statistics: An industry 4.0 perspective”, *Advanced Theory and Simulations*, vol. 4, no. 2, p. 2000254, 2021. DOI: <https://doi.org/10.1002/adts.202000254>
- [5] L. N. L. de Lacalle y J. Posada, “Special issue on new industry 4.0 advances in industrial iot and visual computing for manufacturing processes”, *Applied Sciences*, vol. 9, no. 20, 2019. DOI: <https://doi.org/10.3390/app9204323>
- [6] A. El Wahabi, I. H. Baraka, S. Hamdoune, y K. El Mokhtari, “Detection and control system for automotive products applications by artificial vision using deep learning”, in *Advanced Intelligent Systems for Sustainable Development (AI2SD’2019)*, M. Ezziyyani, Ed. Cham: Springer International Publishing, 2020, pp. 224–241. DOI: https://doi.org/10.1007/978-3-030-36671-1_20
- [7] A. N. Abbas, M. A. Irshad, y H. H. Ammar, “Experimental analysis of trajectory control using computer vision and artificial intelligence for autonomous vehicles”, *CoRR*, vol. abs/2106.07003, 2021. DOI: <https://doi.org/10.48550/arXiv.2106.07003>
- [8] H.-y. Zhang, W.-m. Lin, y A.-x. Chen, “Path planning for the mobile robot: A review”, *Symmetry*, vol. 10, no. 10, 2018. DOI: <https://doi.org/10.3390/sym10100450>
- [9] S. Alcántara Tacora, E. López Zapata, J. Peralta Toribio, y R. Rodríguez Bustinza, “Planificación de movimiento mediante algoritmo de campos potenciales con optimización de parámetros aplicado a un manipulador antropomorfo de 6 gld”, *TECNIA*, vol. 31, no. 2, pp. 39–47, jun. 2021. DOI: <https://doi.org/10.21754/tecnica.v21i2.848>
- [10] J. Zhong, T. Wang, y L. Cheng, “Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics”, *Complex & Intelligent Systems*, pp. 1–14, 2021. DOI: <https://doi.org/10.1007/s40747-021-00366-1>
- [11] L. Kavraki, P. Svestka, J.-C. Latombe, y M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”, *IEEE Transactions on Robotics and Automation*, 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”, *The annual research report*, 1998.
- [13] R. K. Jenamani, R. Kumar, P. Mall, y K. Keddia, “Robotic motion planning using learned critical sources and local sampling”,

- CoRR, vol. abs/2006.04194, 2020. DOI: <https://doi.org/10.48550/arXiv.2006.04194>
- [14] H. Zhang, Y. Wang, J. Zheng, y J. Yu, “Path planning of industrial robot based on improved rrt algorithm in complex environments”, *IEEE Access*, vol. 6, pp. 53 296–53 306, 2018. DOI: <https://doi.org/10.1109/ACCESS.2018.2871222>
- [15] X. Cao, X. Zou, C. Jia, M. Chen, y Z. Zeng, “Rrt-based path planning for an intelligent litchi-picking manipulator”, *Computers and Electronics in Agriculture*, vol. 156, pp. 105–118, 2019. DOI: <https://doi.org/10.1016/j.compag.2018.10.031>
- [16] S. Chakraborty y P. S. Aithal, “Abb irb 120-30.6 build procedure in robodk”, *International Journal of Management, Technology, and Social Sciences (IJMTS)*, vol. 6, no. 2, pp. 256–264, Dec. 2021. DOI: <https://doi.org/10.5281/zenodo.5782759>
- [17] P. Tangseng, Z. Wu, y K. Yamaguchi, “Looking at outfit to parse clothing”, *arXiv*, Mar 2017. DOI: <https://doi.org/10.48550/ARXIV.1703.01386>
- [18] A. Garbev y 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 y E. Frazzoli, “Sampling-based algorithms for optimal motion planning”, *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. DOI: <https://doi.org/10.1177/0278364911406761>
- [20] . J. p. s. Pedro Alejandro Herrera, William Alberto Ramirez. (2021, Dec.) Tools in industry(ROBODK). DOI: <https://doi.org/10.5281/zenodo.5768611>