Sistema didáctico para procesamiento de imagen y generación de trayectoria para un robot móvil.

Jorge Alberto García Muñoz
Departamento de Ingeniería Eléctrica y Electrónica
Tecnológico Nacional de México en Celaya
Celaya, Guanajuato, México
jorge.garcia@itcelaya.edu.mx

José Alfredo Padilla Medina
Departamento de Ingeniería Eléctrica y Electrónica
Tecnológico Nacional de México en Celaya
Celaya, Guanajuato, México
alfredo.padilla@itcelaya.edu.mx

Oscar Quiñones Busquets
Departamento de Ingeniería Eléctrica y Electrónica
Tecnológico Nacional de México en Celaya
Celaya, Guanajuato, México
M1903020@itcelaya.edu.mx

***Abstract*— Uno de los tópicos actuales de investigación tiene que ver con la navegación autónoma de vehículos autoguiados. Para tal fin, se han propuesto una serie de estrategias entre las que destacan métodos que pueden clasificarse en: mapa de carretera ("roadmap"), métodos de descomposición en celdas, métodos de campos de potencial, métodos probabilísticos, entre otros. Por otra parte, se requiere de metodologías para la detección de obstáculos basadas en diversos tipos de sensores, como los ultrasónicos, infrarrojos, sistemas de posicionamiento global, sensores lidar, visión artificial, etc.**

Muchos de estos métodos propuestos en la literatura difícilmente se tratan en la actualidad a nivel de licenciatura y menos aún se vinculan entre sí y se desarrollan en un curso. Se propone en este trabajo una implementación sencilla para la detección de obstáculos sobre la plataforma de programación Python empleando la biblioteca libre de funciones para visión artificial OpenCV, desarrollada por Intel, así como la programación de un algoritmo de generación de trayectorias basado en campos de potencial, pensando en el empleo de herramientas con licencia de código libre. Todo esto con el fin de demostrar la sencillez de la implementación

Keywords*— *Robótica móvil, planificación de trayectorias, campos de potencial artificial, reconocimiento de obstáculos.

I. INTRODUCCIÓN

Los robots móviles son ampliamente utilizados en muchos campos industriales. La investigación sobre la planificación de rutas es uno de los aspectos más importantes en la investigación de robots móviles. La planificación de la ruta consiste en encontrar una camino libre de colisión, a través del entorno del robot con obstáculos, partiendo desde una ubicación de inicio específica hasta un destino objetivo deseado o meta, mientras se satisfacen ciertos criterios de optimización.

Los robots móviles autónomos son capaces de moverse en cualquier entorno, pero no están fijos en una ubicación [1]. Se pueden usar en diversas aplicaciones como inspección en zonas de peligro, buscador de objetivos, exploración y patrullaje de seguridad además de otros campos como agricultura, medicina, minería, misión espacial, industrias, militar y educación. Los agentes del robot móvil se utilizan para interactuar con un entorno y el requisito básico en la planificación del movimiento del robot móvil que incluye la exploración del camino, la localización y la evasión de obstáculos. La planificación de la ruta en un robot móvil autónomo es más compleja cuando se opera en un entorno dinámico o no estructurado en comparación con un entorno estático. Algunos robots pueden ir a donde los humanos no podrían sobrevivir para recopilar la información o realizar tareas especializadas.

Se han propuesto muchas actividades de investigación en el campo de los robots móviles en condiciones difíciles durante las últimas décadas. Un robot móvil autónomo debe ser inteligente al obtener la información sobre el entorno, adaptarse al entorno cambiante y decidir sin ninguna interacción humana. La navegación es la capacidad de un robot móvil autónomo para planificar su ruta en tiempo real y navegar de forma segura desde una ubicación "A" a otra ubicación "B". Esto se logra utilizando ciertas técnicas. La planificación del movimiento del robot es el método básico para seleccionar si la implementación se puede realizar en un entorno estático o dinámico.

Los algoritmos de planificación de movimiento en el robot móvil utilizan tres enfoques básicos que son: algoritmos basado en cuadrícula, algoritmos basado en muestreo y algoritmos basado en potencial [2]. Los problemas de baja dimensión se pueden resolver con algoritmos basados en cuadrículas que se superponen a una cuadrícula en la parte superior del espacio de configuración, o algoritmos geométricos que calculan la forma y la conectividad del espacio de configuraciones libre de colisiones. Los algoritmos de campo potencial atraen al robot móvil hacia la posición de meta, pero pueden caer como presa de mínimos locales. Los algoritmos basados en muestreo evitan los mínimos locales y se consideran como el estado del arte en un espacio de alta dimensión. Los robots móviles son aplicables a varios campos como la agricultura, fábricas de materiales, transporte, almacén, edificios de oficinas, patrullas de seguridad interiores y exteriores, limpieza de sitios, aplicaciones submarinas y aplicaciones militares. Los trabajos de investigación actuales en robótica son la robótica humanoide y la interacción humano-robot.

La planificación de ruta o planificación de movimiento es el proceso de fraccionar la trayectoria deseada para trabajar en un movimientos separados que satisfagan la trayectoria y posiblemente, se optimicen varios aspectos del movimiento [3]. La planificación de la ruta en el robot móvil es muy compleja e incluso más complicada si el robot necesita encontrar una función objetivo como la ruta más corta, la minimización de energía, menos tiempo e implica resolver laberintos. El robot móvil también se encuentra con muchos problemas para evitar obstáculos, lo cual es extremadamente importante e implica lograr un objetivo en una ubicación estática y dinámica evitando o detectando obstáculos. La planificación del movimiento consta de muchos enfoques, pero generalmente involucra solo dos enfoques para encontrar la tarea. El primer enfoque se representa como planificación de ruta global y el segundo la planificación de rutas locales.

Una planificación de ruta global [3] es la descripción completa de una meta y un espacio estático de los obstáculos que están predefinidos o cuyos detalles se conocen de antemano. Un conocimiento previo de la ubicación le permite al robot encontrar el mejor camino para alcanzar su objetivo evitando la colisión con obstáculos. Aquí, los obstáculos son estáticos o predefinidos, la ruta es bien conocida por el robot móvil autónomo y la planificación de la ruta es aplicable tanto

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

en entornos interiores como exteriores. Por lo general, se obtiene una ruta de baja resolución y alto nivel desde la ubicación A hasta la ubicación B en el entorno evitando obstáculos y logrando una navegación efectiva. Un mapa completo, una cuadrícula y celdas se cargan previamente en el robot y el algoritmo de navegación determina la ruta óptima incluso antes de que el robot comience su movimiento. Aunque la descripción completa del entorno está disponible de antemano, la planificación del camino global es un problema muy complejo y se resuelve utilizando un algoritmo en un mapa de cuadrícula a gran escala [4]. El problema de la planificación de ruta global también se puede resolver utilizando métodos de búsqueda exactos o meta heurísticos [6] que exploran el espacio de búsqueda porque a menudo se pueden encontrar buenas soluciones con menos enfoque computacional, en lugar de cualquier otro método iterativo o tradicional. El método exacto tiene ventajas de integridad y el tiempo de ejecución aumenta con el tamaño del problema. La planificación de ruta global se aplica a mapas de cuadrícula a gran escala y también se combina con algoritmos como A*, D* y el Algoritmo de Dijkstra para encontrar una ruta más corta y de menor costo desde el origen hasta el destino en una topología al dividir un entorno en varias cuadrículas. La comparación entre el método de búsqueda exacta y meta heurística se menciona en [7].

Se prefiere la planificación de ruta local (LPP) [6] cuando la descripción ambiental es incompleta y cuando el entorno para escenarios en tiempo real es dinámico. Se genera una nueva ruta actualizando la planificación de ruta global original en respuesta a un cambio geográfico en el entorno. En una planificación de ruta local, la ruta u obstáculos no se conocen de antemano, por lo que el robot debe detectar el entorno antes de decidir moverse y generar una planificación de trayectoria hacia el destino. Si se encuentra el obstáculo, el robot móvil autónomo se desvía de su ruta actual y busca una nueva ruta hacia la ubicación objetivo en el entorno. El campo de potencial artificial es una de las técnicas de LPP más simples que opera en función del potencial de atracción y repulsión. En esta técnica, el robot móvil repele al identificar un obstáculo en el camino y atrae al identificar la posición objetivo deseada. Se implementa una estructura de planificación local de ruta [6] para el robot utilizando el movimiento Row-Wised y Column-Wised. En un movimiento de fila, el robot puede moverse fila por fila desde el punto inicial hasta el punto final, es decir, el robot se mueve en una línea horizontal donde el camino se encuentra solo una vez. Del mismo modo, en el movimiento de columna, el robot puede moverse columna por columna al destino desde la fuente (es decir, el objetivo se logra en un espacio en el formato de línea vertical). La técnica de codificación también se aplica para resolver los problemas de planificación de ruta que consta de cuatro variables como Ruta-Ubicación, Ruta-Dirección, Ruta-Bandera y Ruta-Interruptor.

La planificación de ruta implica dos enfoques generales, a saber, el método tradicional o convencional y el método de computación flexible. El método tradicional no aplica la inteligencia en la planificación de la ruta e incluye técnicas de búsqueda de gráficos, campo de potencial artificial, método de descomposición celular, método de campo vectorial y método de mapa de ruta. Los métodos de computación flexible introducen inteligencia en la planificación del camino y eso incluye algoritmo genético, optimización de colonias de hormigas, algoritmo de enjambre, redes neuronales y lógica difusa.

Los algoritmos Bio-Inspirados también se aplican para encontrar una ruta para el robot móvil donde el proceso de encontrar una solución se compara con cualquiera de los caracteres de los sistemas biológicos. La acción de los robots móviles para encontrar un camino o evitar el obstáculo puede derivarse de la inteligencia del comportamiento de un animal o pájaro. En el algoritmo Bug, se usa un sensor para detectar un obstáculo y minimizar el perímetro exterior del uso de un obstáculo sin la necesidad de un mapa y una respuesta a la salida basada en un contacto sensorial. La planificación de ruta se ha demostrado utilizando diferentes tipos de algoritmos de error como Bug1, Bug2 y Distbug [8].

En este trabajo se presenta solamente un algoritmo simple de planificación de trayectorias basado en campos de potencial artificial, que se describe en la sección II, implementado en lenguanje Python de manera que sea de fácil comprensión para alumnos a nivel licenciatura y promueva el interés en el estudio de la robótica. La sección III relata aspectos de la implementación del algoritmo así como la estructura móvil de prueba. Finalmente, sn la sección IV se presentan los resultados obtenidos.

II. CAMPOS DE POTENCIAL ARTIFICIAL

A. Funciones de potencial.

Una función potencial es una función de valor real diferenciable U: $\mathbb{R}^m \rightarrow \mathbb{R}$. El valor de una función potencial puede verse como energía y, por lo tanto, el gradiente del potencial es la fuerza. El gradiente es un vector

$$\nabla U(q) = DU(q)^T = \left[\frac{\partial U}{\partial q_1}(q),..., \frac{\partial U}{\partial q_m}(q)\right]^T \quad (1)$$

que apunta en la dirección que aumenta localmente al máximo U. El gradiente se emplea para definir un campo vectorial, que asigna un vector a cada punto. Cuando U es energía, el campo del vector gradiente tiene la propiedad de que el trabajo realizado a lo largo de cualquier camino cerrado es cero.

El enfoque de función potencial dirige a un robot como si fuera una partícula que se mueve en un campo vectorial de gradiente. Los gradientes se pueden ver intuitivamente como fuerzas que actúan sobre un robot de partículas con carga positiva que se siente atraído por el objetivo con carga negativa, figura 1. Los obstáculos también tienen una carga positiva que forma una fuerza repulsiva que aleja al robot de los obstáculos. La combinación de fuerzas repulsivas y atractivas dirige al robot desde la ubicación inicial a la ubicación objetivo, evitando obstáculos. Si ignoramos la dinámica del robot, los gradientes pueden tratarse como vectores de velocidad en lugar de vectores de fuerza.

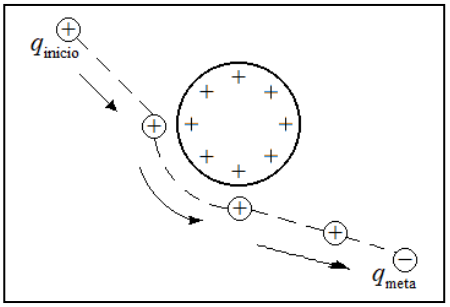

Fig. 1. La carga negativa atrae al robot y la carga positiva lo repele, lo que da como resultado una ruta, indicada por la línea punteada, alrededor del obstáculo y hacia la meta.

Las funciones potenciales pueden verse como un paisaje donde los robots se mueven por un camino "cuesta abajo" siguiendo el gradiente negado de la función potencial. Seguir ese camino se llama descenso en gradiente, es decir,

$$c(t) = -\nabla U(c(t)). \tag{2}$$

El robot termina el movimiento cuando alcanza un punto donde el gradiente se desvanece, es decir, ha alcanzado a q∗ donde $\nabla U\ (q^*) = 0$. Tal punto q^* se llama un punto crítico de U. El punto q^* es un punto máximo, mínimo o de silla de montar.

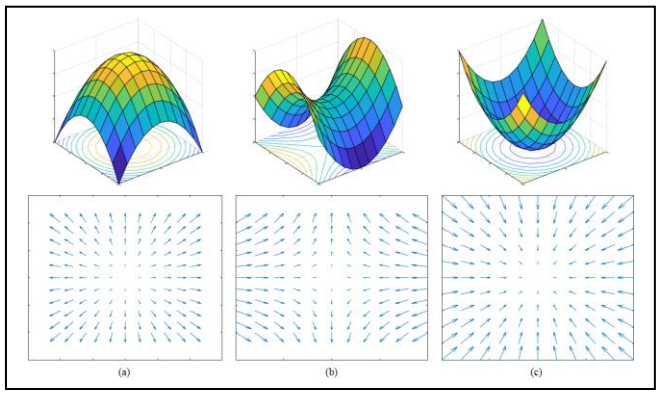

Fig. 2. Diferentes tipos de puntos críticos: (arriba) gráficos de funciones, (inferior) gradientes de funciones. (a) máximo, (b) punto silla, (c) mínimo.

La segunda derivada determina el tipo de punto crítico. Para funciones de valor real, esta segunda derivada es la matriz de Hesse.

Cuando el hessiano no es singular en q^*, el punto crítico en q^* no está degenerado, lo que implica que el punto crítico está aislado [9]. Cuando el hessiano es positivo-definido, el punto crítico es un mínimo local; cuando el hessiano es negativo-definido, entonces el punto crítico es un máximo local. En general, consideramos las funciones potenciales cuyos hessianos son no singulares, es decir, aquellos que solo tienen puntos críticos aislados. Esto también significa que la función potencial nunca es plana.

Para los métodos de descenso de gradiente, no es necesario calcular el hessiano porque el robot termina genéricamente su movimiento en un mínimo local, no en un máximo local o en un punto de silla de montar. Como suponemos que la función nunca es plana, el conjunto de máximos contiene solo puntos aislados, y la probabilidad de comenzar en uno es prácticamente cero. Sin embargo, incluso si el robot comienza al máximo, cualquier perturbación de la posición del robot lo libera, permitiendo que el campo vectorial del gradiente induzca movimiento sobre el robot. Llegar a un punto de silla también es poco probable, porque también son inestables. Los mínimos locales, por otro lado, son estables porque después de cualquier perturbación desde un mínimo, el descenso de gradiente devuelve el robot al mínimo. Con suerte, aquí es donde se encontraría la meta.

Hay muchas funciones potenciales además del potencial atractivo/repulsivo. Muchas de estas funciones potenciales son eficientes para calcular y pueden calcularse en línea [10]. Desafortunadamente, todos sufren un problema: la existencia de mínimos locales que no corresponden a la meta. Este problema significa que las funciones potenciales pueden llevar al robot a un punto que no es el objetivo; en otras palabras, muchas funciones potenciales no conducen a planificadores de ruta completos. Dos clases de enfoques abordan este problema: la primera clase aumenta el campo potencial con un planificador basado en búsquedas, y la segunda define una función potencial con un mínimo local, llamada función de navegación [11]. Aunque completo (o resolución completa), ambos métodos requieren un conocimiento completo del espacio de configuración antes del evento de planificación.

II. Potencial atractivo/repulsivo

La función potencial más simple en Q_{free} es el potencial atractivo/repulsivo. La idean detrás del potencial atractivo / repulsivo es directa: el objetivo atrae al robot mientras los obstáculos lo repelen. La suma de estos efectos atrae al robot hacia la meta mientras lo desvía de los obstáculos. La función potencial se puede definir como la suma de potenciales atractivos y repulsivos.

$$U\ (q) = U_{\text{atr}}\ (\text{q}) + U_{\text{rep}}\ (q) \tag{3}$$

A. *Potencial de atracción*

Hay varios criterios que el campo potencial U_{atr} debe satisfacer. Primero, U_{atr} debería estar aumentando monotónicamente con la distancia desde q_{meta}. La opción más simple es el potencial cónico, que mide una distancia escalada a la meta, es decir, $U(q) = \zeta d(q,q_{\text{meta}})$. Donde ζ es un parámetro utilizado para escalar el efecto del potencial atractivo. El gradiente atractivo es:

$$\nabla U(q) = \frac{\zeta}{d(q, q_{\text{meta}})}(q - q_{\text{meta}}) \tag{3}$$

El vector gradiente apunta lejos del objetivo con magnitud ζ en todos los puntos del espacio de configuración, excepto en la meta, donde no está definido. Comenzando desde cualquier punto que no sea la meta, siguiendo el gradiente negado, se traza un camino hacia la meta.

Al implementar numéricamente este método, el descenso del gradiente puede tener problemas de "parloteo" ya que hay una discontinuidad en el gradiente atractivo en el origen. Por esta razón, se prefiere una función potencial que sea continuamente diferenciable, de modo que la magnitud del gradiente atractivo disminuya a medida que el robot se acerca a q_{meta}. La función potencial más simple es aquella que crece cuadráticamente con la distancia a q_{meta}, por ejemplo,

$$U_{\text{atr}}(q) = \frac{1}{2}\zeta d^2(q, q_{\text{meta}})$$

con el gradiente

$$\begin{aligned}\nabla U_{\text{atr}}(q) &= \nabla\left(\frac{1}{2}\zeta d^2(q, q_{\text{meta}})\right), \\ &= \frac{1}{2}\zeta d^2(q, q_{\text{meta}}), \\ &= \zeta(q - q_{\text{meta}})\end{aligned}$$

que es un vector basado en q, apunta lejos de q_{meta} y tiene una magnitud proporcional a la distancia de q a q_{meta}. Cuanto más lejos esté q de q_{meta}, mayor será la magnitud del vector. En otras palabras, cuando el robot está lejos de la meta, se acerca rápidamente; Cuando el robot está cerca de la meta, el robot se acerca lentamente. Esta característica es útil para los robots móviles porque reduce el "exceso" del objetivo (resultante de la cuantificación por pasos).

En la figura 3 (a), el objetivo está en el centro y se dibujan los vectores de gradiente para varios puntos. La Figura 3 (b) contiene un diagrama de contorno para U_{atr}; cada círculo sólido corresponde a un conjunto de puntos q donde $U_{atr}(q)$ es constante. Finalmente, la figura 3 (c) traza la gráfica del potencial atractivo. Considerando que si bien el gradiente $\nabla U_{atr}(q)$ converge linealmente a cero cuando q se acerca a q_{meta} (que es una propiedad deseable), crece sin límites a medida que q se aleja de q_{meta}. Si q_{inicio} está lejos de q_{meta}, esto puede producir una velocidad deseada que es demasiado grande. Por esta razón, podemos elegir combinar los potenciales cuadráticos y cónicos para que el potencial cónico atrae al robot cuando está muy alejado de q_{meta} y el potencial cuadrático atrae al robot cuando está cerca de q_{meta}. Por supuesto, es necesario que el gradiente se defina en el límite entre las porciones cónica y cuadrática. Tal campo puede ser definido por

$$U_{atr}(q) = \begin{cases} \frac{1}{2}\zeta d^2(q, q_{meta}), & d(q, q_{meta}) \le d^*_{meta}, \\ d^*_{meta}\zeta d(q, q_{meta}) - \frac{1}{2}\zeta (d^*_{meta})^2, & d(q, q_{meta}) > d^*_{meta}. \end{cases} \quad (4)$$

$$\nabla U_{atr}(q) = \begin{cases} \zeta(q - q_{meta}), & d(q, q_{meta}) \le d^*_{meta}, \\ \frac{d^*_{meta}\zeta(q - q_{meta})}{d(q, q_{meta})}, & d(q, q_{meta}) > d^*_{meta}. \end{cases} \quad (5)$$

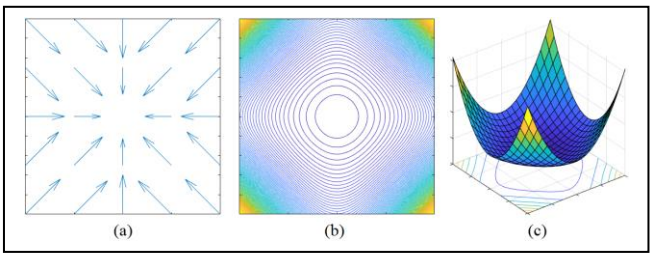

Fig. 3. (a) Campo vectorial de gradiente atractivo, (b) isocontornos potenciales atractivos, (c) gráfico del potencial atractivo.

donde d^*_{meta} es la distancia umbral del objetivo donde el planificador cambia entre potenciales cónicos y cuadráticos. El gradiente está bien definido en el límite de los dos campos ya que en el límite donde $d(q, q_{meta}) = d^*_{meta}$, el gradiente del potencial cuadrático es igual al gradiente del potencial cónico, $\nabla U_{atr}(q) = \zeta (q - q_{meta})$.

B. Potencial de repulsión

Un potencial repulsivo mantiene al robot alejado de un obstáculo. La fuerza de la fuerza repulsiva depende de la proximidad del robot al obstáculo. Cuanto más cerca esté el robot de un obstáculo, más fuerte debe ser la fuerza repulsiva. Por lo tanto, el potencial repulsivo generalmente se define en términos de distancia al obstáculo más cercano $D(q)$, es decir,

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)^2, & D(q) \le Q^* \\ 0, & D(q) > Q^* \end{cases} \quad (6)$$

cuyo gradiente es:

$$\nabla U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{Q^*} - \frac{1}{D(q)}\right)\frac{1}{D^2(q)}\nabla D(q), & D(q) \le Q^* \\ 0, & D(q) > Q^* \end{cases} \quad (7)$$

donde el factor $Q^* \in \mathbb{R}$ le permite al robot ignorar los obstáculos lo suficientemente lejos de él y η puede verse como una ganancia en el gradiente repulsivo. Estos escalares generalmente se determinan por ensayo y error.

Al implementar numéricamente esta solución, se puede formar una ruta que oscila alrededor de puntos que son equidistantes en dos direcciones de los obstáculos, es decir, puntos donde D no es suave. Para evitar estas oscilaciones, en lugar de definir la función de potencial repulsivo en términos de distancia al obstáculo más cercano, la función de potencial repulsivo se redefine en términos de distancias a obstáculos individuales donde $d_i(q)$ es la distancia al obstáculo $\mathcal{QO}_i$, es decir,

$$d_i(q) = \min_{c \in \mathcal{QO}_i} d(q, c) \quad (8)$$

Se puede mostrar para obstáculos convexos $\mathcal{QO}_i$, donde c es el punto más cercano a x que el gradiente de $d_i(q)$ es

$$\nabla d_i(q) = \frac{q - c}{d(q, c)} \quad (9)$$

El vector $\nabla d_i(q)$ describe la dirección que aumenta al máximo la distancia a $\mathcal{QO}_i$ desde q (figura 4).

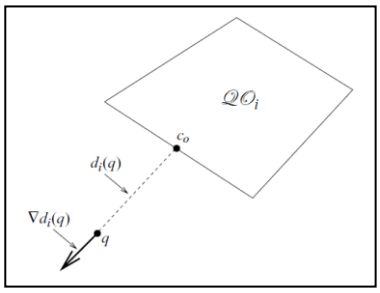

Fig. 4. La distancia entre x y $\mathcal{QO}_i$ es la distancia al punto más cercano en $\mathcal{QO}_i$. El gradiente es un vector unitario que apunta lejos del punto más cercano.

Ahora, cada obstáculo tiene su propia función potencial,

$$U_{rep_i}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d_i(q)} - \frac{1}{Q^*}\right)^2, & d_i(q) \le Q^* \\ 0, & d_i(q) > Q^* \end{cases}$$

donde Q^*_i define el tamaño del dominio de influencia para el obstáculo $\mathcal{QO}_i$. Entonces

$$U_{rep}(q) = \sum_{i=1}^{n} U_{rep_i}(q)$$

Suponiendo que solo hay obstáculos convexos o no convexos que se pueden descomponer en piezas convexas, las oscilaciones ya no ocurren porque el planificador ya no tiene cambios radicales en el punto más cercano.

C. Gradiente descendente

El descenso de gradiente es un enfoque bien conocido para los problemas de optimización. La idea es simple. Comenzando en la configuración inicial, dé un pequeño paso en la dirección opuesta al gradiente. Esto proporciona una nueva configuración y el proceso se repite hasta que el gradiente sea cero. Más formalmente, podemos definir un algoritmo de descenso de gradiente.

La figura 5 muestra un mapa tridimensional con la combinación de los potenciales atractivos y de repulsión para un escenario con dos obstáculos.

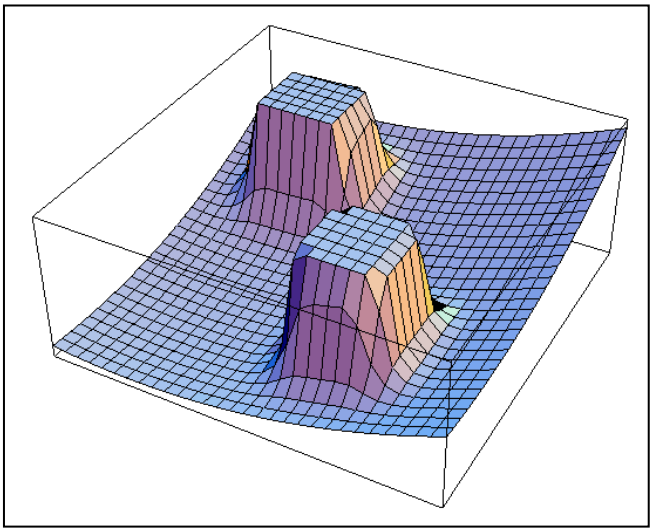

Fig. 5. Mapa tridimensional de la combinación de potenciales atractivo y repulsivo para un escenario con dos obstáculos.

Algoritmo. Gradiente descendente

Entrada: Un método para calcular el gradiente $\nabla U(q)$ en un punto q

Salida: Una secuencia de puntos $\{q(0), q(1), \ldots, q(i)\}$

1: $q(0) = q_{inicio}$
2: $i = 0$
3: mientras $\nabla U(q(i)) = 0$ hacer
4: $q(i + 1) = q(i) + \alpha(i) \nabla U(q(i))$
5: $i = i + 1$
6: finaliza

En el algoritmo, la notación $q(i)$ se usa para denotar el valor de q en la i-ésima iteración y la ruta final consiste en la secuencia de iteraciones $\{q(0), q(1), \ldots, q(i)\}$. El valor del escalar $\alpha(i)$ determina el tamaño del paso en la iteración i. Es importante que $\alpha(i)$ sea lo suficientemente pequeño como para que el robot no pueda "saltar" obstáculos, mientras que sea lo suficientemente grande como para que el algoritmo no requiera un tiempo de cálculo excesivo. En problemas de planificación de movimiento, la elección de $\alpha(i)$ a menudo se realiza de manera ad hoc o empírica, tal vez en función de la distancia al obstáculo más cercano o al objetivo. Una serie de métodos sistemáticos para elegir $\alpha(i)$ se puede encontrar en la literatura de optimización [12]. Finalmente, es poco probable que alguna vez se satisfaga exactamente la condición $\nabla U(q(i)) = 0$. Por esta razón, esta condición a menudo se reemplaza por la condición más indulgente. $\|\nabla U(q(i))\| < \varepsilon$, en el que se elige que ε sea lo suficientemente pequeño, en función de los requisitos de la tarea.

III. Implementación y resultados

Para la implementación del algoritmo se empleó el lenguaje de programación interpretado Python, cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, que soporta programación orientada a objetos, programación imperativa y, en menor medida, programación funcional. Existen varios ambientes de programación para Python, en este caso se utilizó Spyder, que es un entorno de desarrollo integrado (IDE) multiplataforma de código abierto para programación científica en lenguaje Python. Spyder se integra con una serie de paquetes destacados en la pila científica de Python, incluidos NumPy, SciPy, Matplotlib, pandas, IPython, SymPy y Cython, así como otro software de código abierto. Se publica bajo la licencia MIT [13].

Los materiales educativos son cada vez más importantes para que los futuros desarrolladores aprendan tecnologías básicas de navegación autónoma. Debido a que estas tecnologías autónomas necesitan diferentes conjuntos de habilidades tecnológicas, tales como: álgebra lineal, estadística, teoría de probabilidad, teoría de optimización y teoría de control, etc. Se necesita mucho tiempo para familiarizarse con estas áreas tecnológicas. Por lo tanto, se necesitan buenos recursos educativos para aprender tecnologías básicas de navegación autónoma. En [14], se describe un proyecto de software de código abierto (OSS) denominado PythonRobotics. Este proyecto proporciona una colección de códigos de algoritmos robóticos, especialmente centrados en navegación. El objetivo principal es proporcionar a los principiantes las herramientas necesarias para comprenderlo. Está escrito en Python bajo licencia MIT. Tiene muchas animaciones de simulación que muestran los comportamientos de cada algoritmo. Permite a los alumnos a comprender las ideas fundamentales.

El proyecto PythonRobotics se basa en tres filosofías principales: la primera es que el código debe ser fácil de leer. Si el código no es fácil de leer, sería difícil lograr el objetivo de permitir que los principiantes entiendan los algoritmos. Python tiene excelentes bibliotecas para operaciones matriciales, operaciones matemáticas y científicas, y visualización, lo que hace que el código sea más legible porque tales operaciones no necesitan ser re-implementadas. Tener los paquetes centrales de Python le permite al usuario concentrarse en los algoritmos, en lugar de las implementaciones. La segunda filosofía es que los algoritmos implementados tienen que ser prácticos y ampliamente utilizados tanto en la academia como en la industria. Aprender estos algoritmos será útil en muchas aplicaciones. Por ejemplo, filtros de Kalman y filtro de partículas para localización, mapeo de cuadrícula para generación de mapas, enfoques basados en programación dinámica, enfoques basados en muestreo para planificación de rutas, y enfoque basado en control óptimo para el seguimiento de rutas. La última filosofía son las dependencias mínimas. Tener pocas dependencias externas permite ejecutar muestras de código fácilmente y convertir los códigos de Python a otros lenguajes de programación, como C++ o Java para una aplicación más práctica. Cada código depende solamente de algunos módulos en Python3 como se muestra a continuación:

- *numpy*, para operaciones matriciales y vectoriales
- *scipy*, para computación matemática y científica
- *matplotlib*, para trazar y visualizar
- *pandas*, para importación y manipulación de datos

Para el reconocimiento de obstáculos mediante procesamiento de imágenes se utilizó la biblioteca para visión artificial desarrollada por Intel OpenCV, en su versión para Python. Su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X, Windows y Android. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estereoscópica y visión robótica. Entre las funciones implementadas en esta biblioteca se encuentran *threshold* que permite determinar un umbral para identificar a un obstáculo a partir de la intensidad de su color o escala de grises, así como la función *contours*, con la que se delimitan los conjuntos de pixeles en la imagen que conformarían un obstáculo, como se muestra en la figura 6. Si la cámara está adecuadamente calibrada, es posible obtener las coordenadas de los obstáculos en el mundo real utilizando funciones que incluye la misma biblioteca OpenCV.

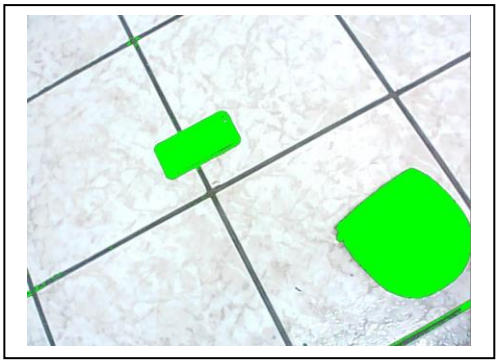

Fig. 6. Imagen capturada y procesada que muestra el mapa de obstáculos en dos dimensiones

Una vez capturada y procesada la imagen se aplicaría el algoritmode campos de potencial, para calcular una ruta libre de obstáculos que permita a robot alcanzar una posición determinada o meta a partir de una posición y configuración inicial, como se muestra en la figura 7.

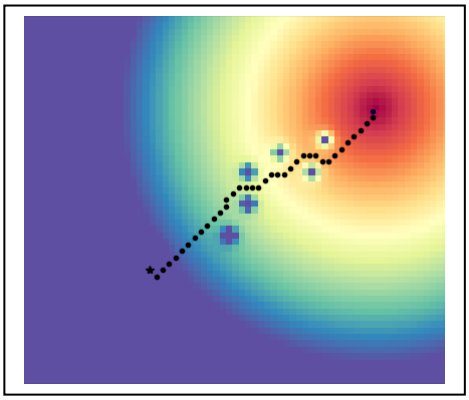

Fig. 7. Mapa bidimensional de campos de potencial con 6 obstáculos y trayectoria calculada.

IV. Conclusiones

En este proyecto se ha realizado una integración de algoritmos de reconocimiento de obstáculos por medio de visión utilizando la bilbioteca libre OpenCV para Python así como el algoritmo de planificación de trayectorias basado en campos artificiales de potencial. Los experimentos se realizaron en un ambiente estático y principalmente con un objetivo didáctico. Los resultados han sido satisfactorios desde el punto de vista del reconocimiento de obstáculos y generación del mapa sobre el que se define la trayectoria a seguir. Se recomienda para ambientes estáticos internos o externos con buena ilumninación. Es recomendable extender el trabajo hacia ambientes dinámicos mediante la actualización de mapas y trayectorias en línea así como extender el trabajo utilizando algoritmos de planificación de trayectorias distintos a los campos de potenciales. Es de gran ayuda contar con una biblioteca de algoritmos desarrollados de manera didáctica que permitan comprender de manera práctica el funcionamiento, características y desempeño de diferentes propuestas para localización, generación de mapas y trayectorias para aplicaciones en robótica, todo esto mediante herramientas de programación de uso libre.

Agradecimientos

El desarrollo de este proyecto ha sido posible gracias al apoyo del Tecnológico Nacional de México en el marco de la convocatoria “Apoyo a la Investigación Científica y Tecnológica en los Institutos Tecnológicos y Centros 2019” en su modalidad de líneas de investigación de acuerdo al convenio 5533.19-P.

Referencias

[1] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza, “Introduction to Auonomous Mobile Robots” second edition, 2004.

[2] P. Victerpaul, D. Saravanan, S. Janakiraman, J. Pradeep, “Path planning of autonomous mobile robots: A survey and comparison”, in Journal of Advanced Research in Dynamical and Control Systems, January 2017.

[3] Maram Alajlan k, Anis Koubaa, Imen Chaari , Hachemi Bennaceur k, Adel Ammark, “Global Path Planning for Mobile Robots in Large-Scale Grid Environments using Genetic Algorithms”, International Journal of AdvancedRobotic SystemsMarch-April 2017: 1–15.

[4] N.A.Vlassis N.M.Sgouros G. Efthivoulidis G. Papakonstantinou P.Tsanakas, “Global Path Planning for Autonomous Qualitative Navigation”, Proc. 8th IEEE Int. Conf. on ToolswithAI(ICTAI), Nov1996,Toulouse,France.

[5] Alejandra Cruz-Bernal, “Meta-Heuristic Optimization Techniques and Its Applications in Robotics”, http://dx.doi.org/10.5772/54460, January 2017

[6] Francesco Mondada, Edoardo Franzii, “Biologically Inspired Mobile Robot Control Algorithms”, 1997. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[7] Antonio Mucherino, Leo Liberti, Carlile Lavor, and Nelson Maculan. 2009. Comparisons between an exact and a metaheuristic algorithm for the molecular distance geometry problem. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO '09). ACM, New York, NY, USA, pp. 333-340. DOI: https://doi.org/10.1145/1569901.1569948, 2009

[8] Alpaslan Yufka, and Osman Parlaktuna, “Performance Comparison of Bug Algorithms for Mobile Robots”,5th International Advanced Technologies Symposium (IATS’09), May 13-15, 2009, Karabuk, Turkey.

[9] V. Guillemin and A. Pollack, editors. “Differential Topology”, Prentice-Hall, Inc., New Jersey, 1974.

[10] O. Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. International Journal of Robotics Research, 5:90–98, 1986.

[11] D. E. Koditschek and E. Rimon. “Robot navigation functions on manifolds with boundary”. Advances in Applied Mathematics, 11:412–442, 1990.

[12] D. Bertsekas. “Nonlinear Programming”. Athena Scientific, Belmont, MA, second edition, 1999.

[13] Licencia del entorno integrado de programación Spyder MIT: https://github.com/spyder-ide/spyder/blob/master/LICENSE.txt, consultada el 18 de noviembre de 2019.

[14] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, A. Paques. “PythonRobotics: a Python code collection of robotics algorithms” https://arxiv.org/abs/1808.10703. Rev. 2019.