# Práctica 5
# Cálculo de rutas utilizando A* y celdas de ocupación

M.I. Marco Negrete

Robots Móviles y Agentes Inteligentes

## Objetivos

- A partir del mapa creado en la práctica 4, calcular una ruta mediante el algoritmo A*.
- Suavizar la ruta utilizando descenso del gradiente.
- Publicar la ruta en un tópico y desplegarla en el visualizador `rviz`.

## 1. Marco Teórico

### 1.1. El algoritmo A*

La planeación de rutas consiste en la obtención de un movimiento continuo libre de colisiones que conecte una configuración inicial, con una final. Se asume que se dispone de una representación del ambiente con información sobre el espacio navegable y el ocupado por los obstáculos. En esta práctica se considera que el robot sólo se mueve sobre un plano y que se tiene una representación que consiste en un mapa de celdas de ocupación (obtenido en la práctica 4).

Una posible solución es aplicar un algoritmo de búsqueda en grafos. En el caso de las celdas de ocupación, cada celda representa un nodo en el grafo y se considera que está conectada únicamente con aquellas celdas vecinas que pertenezcan al espacio libre. Para determinar los nodos vecinos se puede utilizar conectividad cuatro u ocho. En esta práctica se utilizará la conectivad cuatro.

A* es un algoritmo de búsqueda que explora la ruta con el menor costo esperado. Para un nodo $n$, el costo esperado $f(n)$ se calcula como

$$f(n) = g(n) + h(n)$$

donde $g(n)$ es el costo de la ruta desde el nodo origen hasta el nodo $n$ y $h(n)$ es una heurística que determina *un* costo que se esperaría tener desde el mismo nodo $n$ hasta el nodo objetivo. Este costo esperado de hecho subestima el valor real, es decir, se debe cumplir que $h(n) \leq g(n) \quad \forall\, n \in \; Grafo$.

En la búsqueda por A* se manejan dos conjuntos principales: la *lista abierta* y la *lista cerrada*. La lista abierta contiene todos los nodos que han sido visitados pero no expandidos y la cerrada, aquellos que han sido visitados *y* expandidos (también llamados nodos conocidos). El algoritmo 1 muestra los pasos en pseudocódigo para implementar A*.

**Algoritmo 1:** Búsqueda con A*

**Datos**: Grafo, nodo inicial, nodo meta
**Resultado**: Ruta óptima expresada como una secuencia de nodos

```
Cerrado ← ∅
Abierto ← {nodo_inicial}
previo(nodo_inicial) ← ∅
mientras Abierto ≠ ∅ hacer
    nodo_actual ← nodo con el menor valor f del conjunto Abierto
    Abierto ← Abierto - {nodo_actual}
    Cerrado ← Cerrado ∪ {nodo_actual}
    si nodo_actual es nodo_meta entonces
        Anunciar éxito y salir de este ciclo
    fin
    para cada nodo_vecino de nodo_actual hacer
        si nodo_vecino ∈ Cerrado entonces
            Continuar con el siguiente nodo_vecino
        fin
        si nodo_vecino ∈ Abierto entonces
            costo_temporal ← g(nodo_actual) + d(nodo_actual, nodo_vecino)
            si costo_temporal < g(nodo_vecino) entonces
                g(nodo_vecino) ← costo_temporal
                f(nodo_vecino) ← costo_temporal + heurística(nodo_vecino, nodo_meta)
                previo(nodo_vecino) ← nodo_actual
            fin
        en otro caso
            g(nodo_vecino) ← g(nodo_actual) + d(nodo_actual, nodo_vecino)
            f(nodo_vecino) ← gnodo_vecino) + heurística(nodo_vecino, nodo_meta)
            previo(nodo_vecino) ← nodo_actual
            Abierto ← Abierto ∪ {nodo_vecino}
        fin
    fin
fin
si nodo_actual ≠ nodo_meta entonces
    Anunciar falla
en otro caso
    RutaOptima ← ∅
    mientras nodo_actual ≠ ∅ hacer
        //El nodo actual se inserta al principio de la ruta
        RutaÓptima ← {nodo_actual} ∪ RutaÓptima
        nodo_actual ← previo(nodo_actual)
    fin
    Regresar RutaÓptima
fin
```

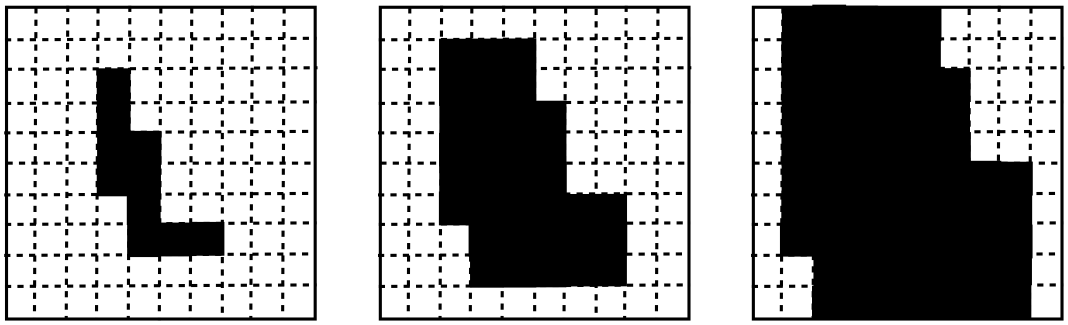

Figura 1: Crecimiento de obstáculos

## 1.2. Crecimiento de obstáculos

Como se mencionó anteriormente, se asume que se tiene un mapa con información sobre el espacio libre y ocupado, sin embargo, en una implementación en un robot real, las celdas que están junto al espacio ocupado no se pueden considerar como libres ya que si el robot se mueve hacia ellas seguramente chocará con algo. Una forma de solucionar esto sería hacer las celdas tan grandes de modo que puedan contener al robot, sin embargo, esta resolución sería muy baja y la representación del ambiente sería muy mala. Otra posible solución es *crecer* los obstáculos de modo que las celdas que están muy cerca del espacio ocupado se consideren también como ocupadas.

La figura 1 muestra un ejemplo en el que los obstáculos se han aumentado dos celdas. Este procedimiento se conoce como *dilatación* y es un tipo de *operador morfológico*. La explicación de estos conceptos está fuera del alcance de esta práctica.

Los obstáculos deben aumentarse cuando menos el radio del robot. Para el mapa obtenido en la práctica 4, dado que la resolución es de 5[cm] y los robots utilizados tienen casi 0.5[m] de diámetro, los obstáculos deben aumentarse cuando menos 5 celdas. El algoritmo 2 enumera los pasos para crecer obstáculos en un mapa.

**Algoritmo 2:** Crecimiento de obstáculos.

**Datos**: Mapa de celdas de ocupación $M = \{c_0 ... c_n\}$, número de celdas a aumentar $k$.
**Resultado**: Mapa con los obstáculos aumentados.
Repetir $k$ veces los siguientes pasos:
**para todo** $c \in M$ **hacer**
  **si** $c$ *está en el espacio ocupado* **entonces**
    $V(c) \leftarrow$ Conjunto de celdas vecinas de $c$ (conectividad ocho)
    **para todo** $v \in V(c)$ **hacer**
      Marcar $v$ como parte del espacio ocupado
    **fin**
  **fin**
**fin**
Regresar $M$

## 1.3. Suavizado de la ruta: función de costo

Dado que se está utilizando conectividad cuatro en el algoritmo de A*, la ruta calculada tendrá siempre vueltas con ángulos rectos, lo cual no es deseable por varias razones: la prime-

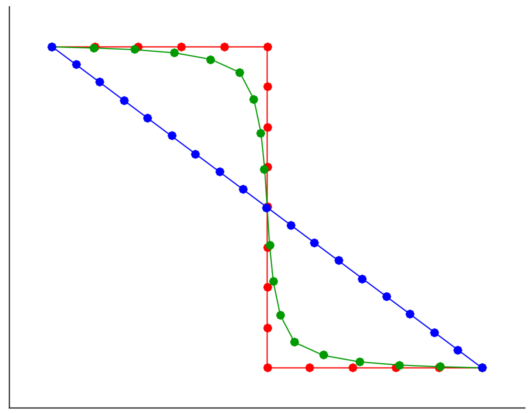

Figura 2: Ejemplos de suavizado

ra de ellas es que la función de posición no es diferenciable en las esquinas, además, este tipo de vueltas ocasionan cambios bruscos en las señales de control, lo que finalmente ocasiona daños a los motores de la base móvil. Por otro lado, las restricciones de movimiento en algunos tipos de bases (como es el caso de los automóviles) pueden impedir ejecutar vueltas en ángulo recto.

Por lo anterior, es conveniente suavizar la ruta calculada por A* de modo que la nueva ruta sea lo suficientemente parecida a la original pero al mismo tiempo, lo suficientemente suave para evitar curvas muy pronunciadas. La figura 2 muestra un ejemplo de suavizado con dos casos extremos: la ruta roja es una ruta igual a la original, la azul es una ruta sin vueltas pero que ha dejado de ser una ruta útil, y la verde, que es un *promedio ponderado* de las dos anteriores.

Para poder obtener una ruta como la verde de la figura 2, plantearemos una función de costo que tome en cuenta el parecido con la ruta original y el *nivel de suavizado* que se desee. Estos dos criterios están en conflicto: entre más suavizado, menor será el parecido con la ruta original y viceversa, es decir, la función de costo será grande con mucho suavizado y también debe ser muy grande si la ruta es muy parecida a la orignal. La idea es que dicha función de costo tenga un mínimo en un punto intermedio de modo que este punto corresponda a una ruta como la verde de la figura 2.

Las funciones cuadráticas son una buena opción como función de costo ya que tienen un mínimo global y además son diferenciables. Para el suavizado de la ruta se propone la función de costo

$$V = \frac{1}{2}\alpha \sum_{i=1}^{n} (p_i - q_i)^2 + \frac{1}{2}\beta \sum_{i=1}^{n-1} (p_i - p_{i+1})^2 \quad (1)$$

donde $Q = \{q_1 \dots q_n\}$ son todos los puntos $q = [x\, y]$ que forman la ruta original calculada con A* y $P = \{p_1 \dots p_n\}$ son los puntos de la ruta ya suavizada. La constante $\alpha \geq 0$ es un parámetro de diseño para indicar cuánto peso se le quiere dar al parecido de la ruta original con la suavizada y $\beta \geq 0$ indica el peso que se le da al suavizado en sí. Con $\alpha = 0$ se obtendría una línea recta, como la ruta azul de la figura 2, y con $\beta = 0$ se obtendría una ruta igual a la original.

## 1.4. Suavizado de la ruta: descenso del gradiente

La ruta suavizada se obtiene encontrando el argumento que minimiza la función 1, sin embargo, dada la cantidad tan grande de variables, es difícil encontrarlo analíticamente. Una forma más sencilla es mediante el médoto del descenso del gradiente, que consiste en mover el argumento pequeñas cantidades proporcionales al gradiente de la función $V$ y en sentido contrario a éste. Dado que el cambio entre cada iteración es proporcional a $\nabla V$, se puede asumir que cuando el cambio en el argumento es menor que una tolerancia, entonces se ha alcanzado el mínimo.

El algoritmo 3 contiene los pasos en pseudocódigo para implementar descenso del gradiente. La constante $\delta > 0$ debe ser lo suficientemente pequeña para evitar inestabilidad en el algoritmo, sin embargo, se debe considerar que entre más pequeña sea ésta, mayor será el costo computacional.

**Algoritmo 3:** Descenso del gradiente.

**Datos**: Función $V$ cuyo mínimo se desea encontrar, condición inicial $p(0)$, tolerancia $tol$.
**Resultado**: Argumento $p$ que minimiza $V$.
$i = 0$
$p_i \leftarrow p(0)$
**mientras** $\|\nabla V(p_i)\| > tol$ **hacer**
  $p_{i+1} \leftarrow p_i - \delta \nabla V(p_i)$
  $i \leftarrow i + 1$
**fin**
Regresar $p_i$

La ecuación 1 toma como argumentos las posiciones tanto de la ruta original como de la suavizada, sin embargo, dado que sólo varían los puntos de la nueva ruta, se puede considerar que el gradiente $\nabla V$ está dado por:

$$\Big[\underbrace{\alpha(p_1 - q_1) + \beta(p_1 - p_2)}_{\frac{\partial V}{\partial p_1}}, \dots, \underbrace{\alpha(p_i - q_i) + \beta(2p_i - p_{i-1} - p_{i+1})}_{\frac{\partial V}{\partial p_i}}, \dots, \underbrace{\alpha(p_n - q_n) + \beta(p_n - p_{n-1})}_{\frac{\partial V}{\partial p_n}}\Big] \tag{2}$$

Nótese que se está derivando con respecto a $p$ (puntos de la ruta suavizada), no con respecto a $q$ (puntos de la ruta original). Recuerde que cada punto de la ruta tiene coordenadas $[x\, y]$, por lo que el descenso de gradiente se tiene que aplicar a ambas coordenadas.

# 2. Tareas

**Nota:** Para esta práctica se asume que el alumno ejecutó correctamente la práctica 4.

## 2.1. Nodo que calcula y publica la ruta

## 2.2. Interfaz gráfica

# 3. Evaluación

- 
- 
- 