Robótica móvil - Un enfoque probabilístico (86.48)

Universidad de Buenos Aires
Facultad de Ingeniería
Año 2020 - 2do cuatrimestre

TRABAJO PRÁCTICO N.º 5
TEMA: ICP y planeamiento
FECHA: 18 de diciembre de 2020
GITHUB: `github.com/fnastasi/8648_Robotica_movil_TPs`

INTEGRANTE :
Nastasi, Franco Gabriel - #100002
fnastasi@fi.uba.ar

1. Mapeo con poses conocidas

Se aplicó el algoritmo ICP que mejora la odometría del robot con un conjunto de puntos de 2 dimensiones. Se aplicó el algoritmo para 2 conjuntos de puntos con correspondencia conocida y luego se ejecutó el algoritmo a partir de asignar a cada uno de los puntos de un conjunto, el punto más cercano del conjunto restante. Se programaron 3 formas de realizar la correspondencia y se ejecutó el algoritmo obteniéndose resultados satisfactorios solamente para uno de ellas en donde se verifica que esta forma de corresponder puntos de un conjunto a otro no siempre arroja resultados satisfactorios. Se generaron 2 conjuntos de puntos P3 y P4 a partir de modificar otro conjunto X y asignar una correspondencia aleatoria entre ambos conjuntos X y P3 y X y P4. A continuación se muestra el resultado de aplicar el algoritmo en donde se ve como se acercan los puntos de un conjunto al otro en cada iteración.
El conjunto P3 se creo a partir de sumarle a X un valor aleatorio gaussiano y luego rotar el conjunto obtenido 5 grados. A continuación se muestra la correspondencia original y la aleatoria y se muestran los 2 primeros pasos del algoritmo. En las iteraciones siguientes no se ven cambios significativos lo que significa que el algoritmo solo necesitó 2 paso para converger:

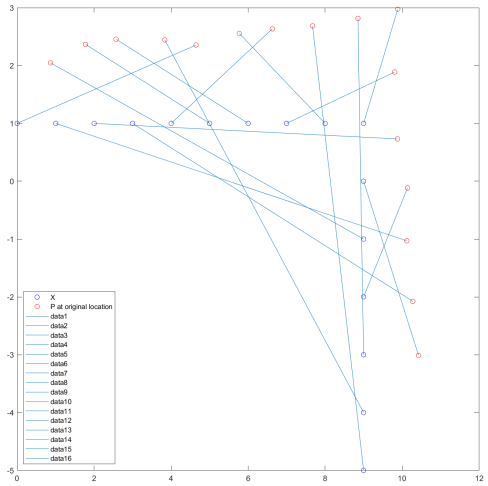

Figura 1.1: Correspondencia aleatoria del conjunto P3

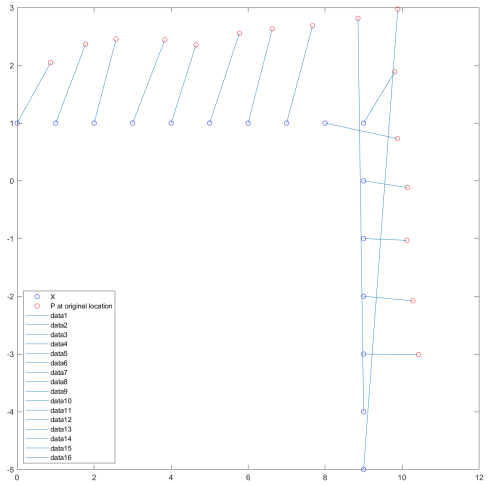

Figura 1.2: Correspondencia real del conjunto P3

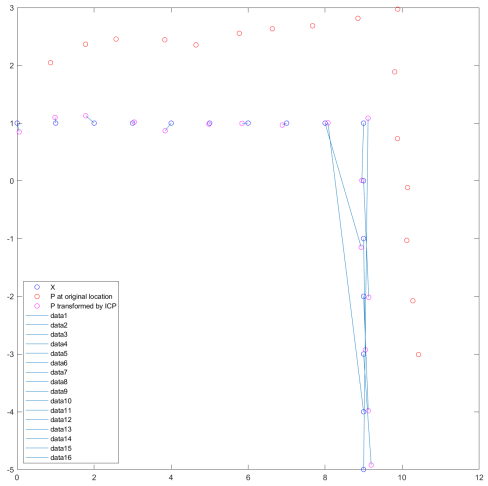

Figura 1.3: Primer paso del algoritmo ICP para el conjunto P3

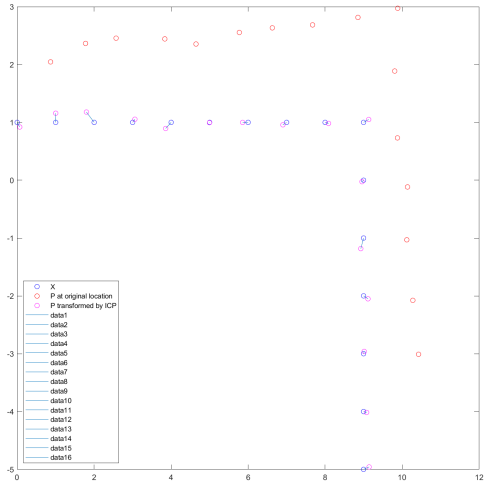

Figura 1.4: Segundo paso del algoritmo ICP para el conjunto P3

El conjunto P4 se creo a partir de sumarle a X un valor aleatorio gaussiano y luego rotar el conjunto obtenido 110 grados. A continuación se muestra la correspondencia original y la aleatoria. En este caso se necesitaron 3 iteraciones para lograr alcanzar la convergencia.

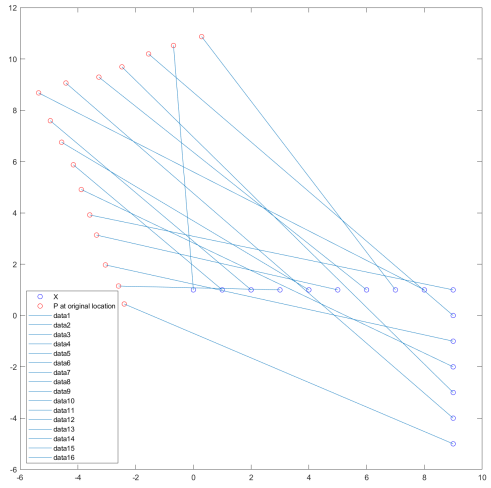

Figura 1.5: Correspondencia aleatoria del conjunto P4

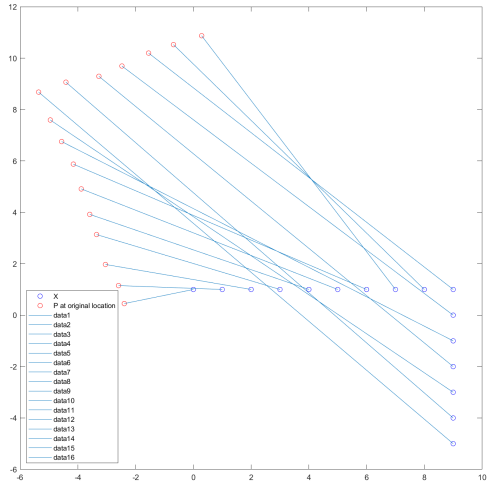

Figura 1.6: Correspondencia real del conjunto P4

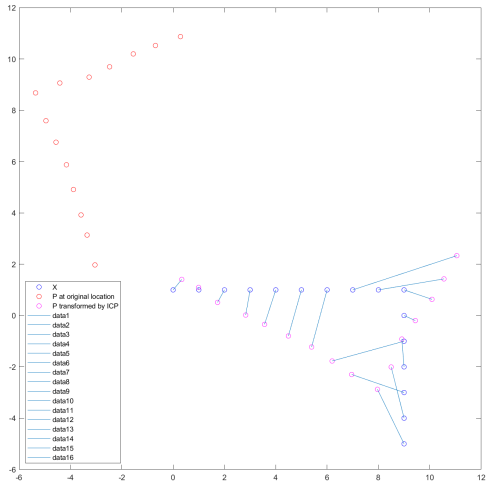

Figura 1.7: Primer paso del algoritmo ICP para el conjunto P4

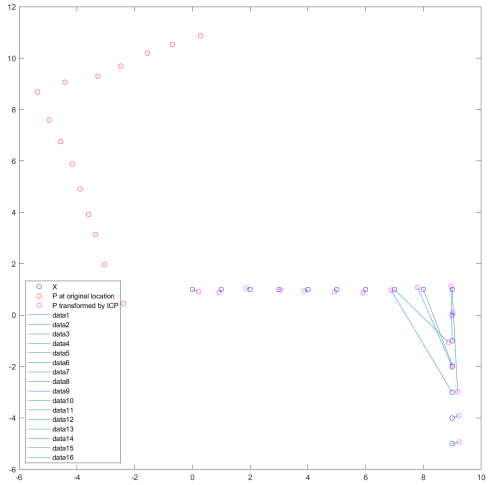

Figura 1.8: Segundo paso del algoritmo ICP para el conjunto P4

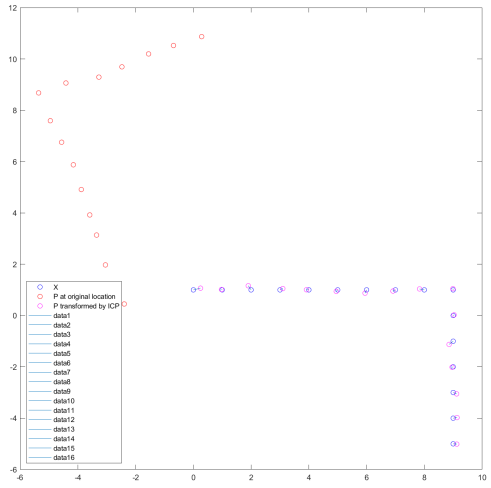

Figura 1.9: Tercer paso del algoritmo ICP para el conjunto P4

Es necesario mencionar que con el conjunto P4, en algunas ocasiones el algoritmo alcanzó mínimos locales que produjeron que no se obtuviesen resultados satisfactorios. A continuación se muestra un resultado en donde el algoritmo ICP converge a un estado en el que los puntos de p4 rotados y trasladados se encuentran alejados de los puntos de X.

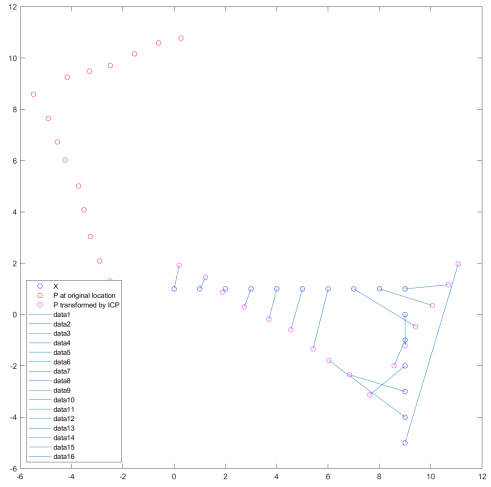

Figura 1.10: Convergencia del algoritmo ICP para el conjunto P4 en el que se alcanza un mínimo local

2. Planeamiento con el algoritmo de Dijkstra y A*

2.1. Algoritmo Dijkstra

Se generaron las funciones necesarias para aplicar el algoritmo de Dijkstra a un robot que se mueve por un entorno sobre el cual se conoce la grilla de ocupación del mismo. Como función costo se utilizó la siguiente:

$$g\left(\begin{bmatrix} y \\ x \end{bmatrix}, \begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix}\right) = k_{dist} \left\| \begin{bmatrix} y \\ x \end{bmatrix} - \begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} \right\| + k_{cost} \cdot cost\left(\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix}\right) \tag{2.1}$$

Es decir la función costo es una combinación lineal de la norma entre 2 nodos adyacentes y el valor de

ocupación del nodo a donde se está evaluando mover el robot. Está métrica permite que, dependiendo los valores de k_{dist} y k_{cost}, se le pueda dar más peso a la distancia entre los nodos o al valor de la grilla de ocupación. Esto produce que el planeamiento difiera significativamente para distintos valores. Para evitar que se tomaran puntos del grafo que corresponden a nodos donde el robot no se puede encontrar se tomó un umbral igual a 0.4. Este valor se tomó considerando que en la imagen de la grilla de ocupación, algunos nodos con probabilidad igual a 0.4 ya podían ser apreciados como obstáculos. A continuación se muestran algunos resultados para distintos valores de k_{dist} y k_{cost}. Primero se ejecutó el algoritmo con $k_{dist} = k_{cost} = 1$ dando como resultado la siguiente figura:

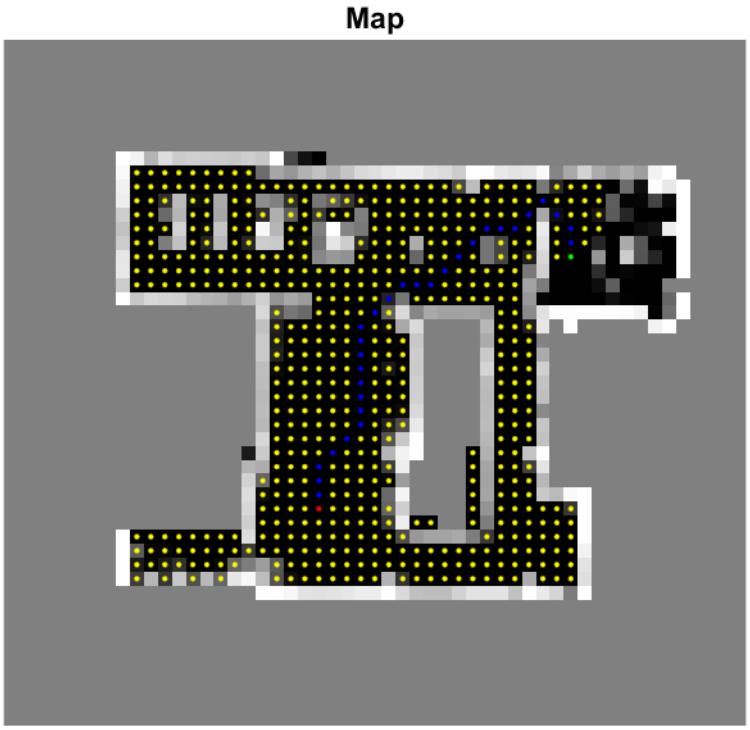

Figura 2.1: Resultado de algoritmo de Dijkstra para $k_{dist} = k_{cost} = 1$

Durante la ejecución del algoritmo se observó como los nodos correspondientes al menor costo en cada iteración se tomaban sin ninguna dirección privilegiada. Esto se debe a los valores elegidos de k_{dist} y k_{cost}. Si se repite la simulación ahora tomando $k_{dist} = 1$ y $k_{cost} = 5$ se obtiene el siguiente resultado en donde se ve que no hay una diferencia apreciable con el caso anterior, esto es, se consideraron aproximadamente los mismos nodos que durante la simulación tuvieron el menor costo y el planeamiento no se vio modificado.

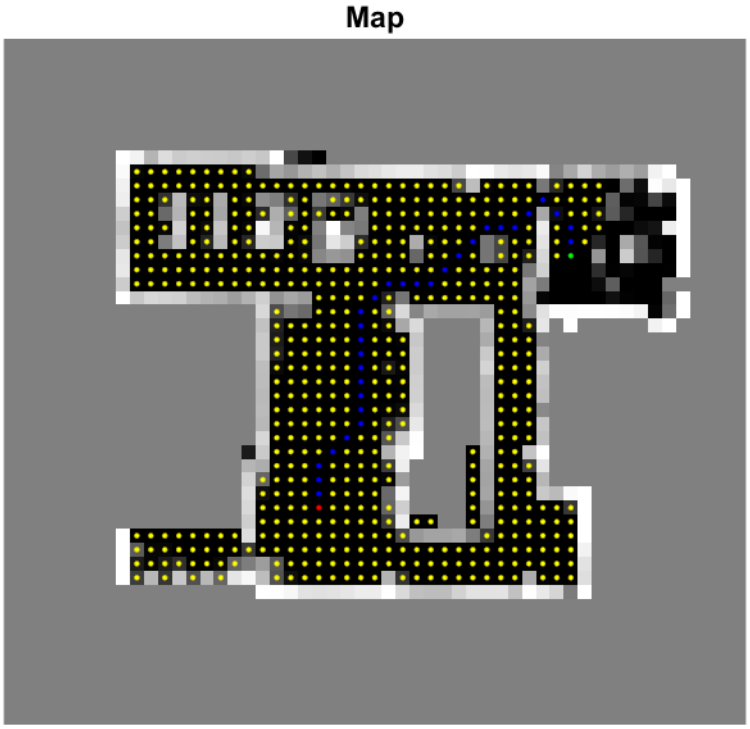

Figura 2.2: Resultado de algoritmo de Dijkstra para $k_{dist} = 1$ y $k_{cost} = 5$

En la siguiente simulación se tomó $k_{dist} = 5$ y $k_{cost} = 1$. Esto significa que el costo de moverse de un nodo a otro está afectado de forma significativa en caso de si se trata de los casilleros diagonales. Esto se debe a que los casilleros diagonales al actual poseen una distancia de $\sqrt{2}$ mientras que para los casilleros adyacentes, la distancia es igual a 1. La diferencia en costo de los nodos diagonales se ve aumentada por el factor $k_{dist}(\sqrt{2}-1)$ con respecto a los nodos adyacentes. Como se ve en la figura siguiente, esto produce que se dé preferencia a la transición entre nodos que corresponden a casilleros adyacentes.

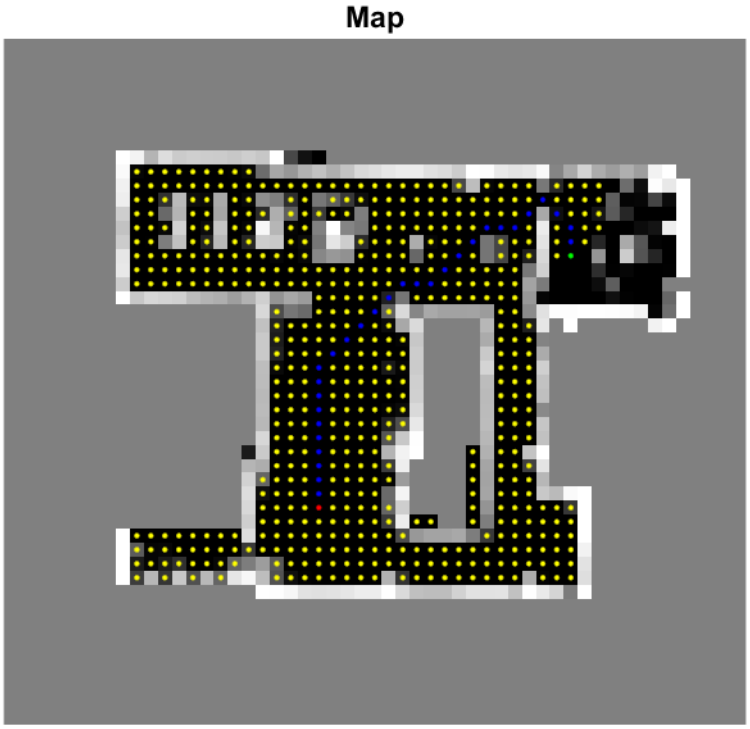

Figura 2.3: Resultado de algoritmo de Dijkstra para $k_{dist} = 5$ y $k_{cost} = 1$

Esta modificación en el planeamiento se puede acentuar si se aumenta significativamente el costo de los nodos correspondientes a los casilleros diagonales. Esto se logró aumentando el costo de este tipo de nodos por un factor de 10. El resultado es el siguiente, en donde se observa como en el planeamiento no se incluye ningún movimiento diagonal. Además se observó que los nodos que correspondían al costo mínimo, esto es, los puntos amarillos, se elegían de forma tal que siempre formaban una estructura similar a un rombo.

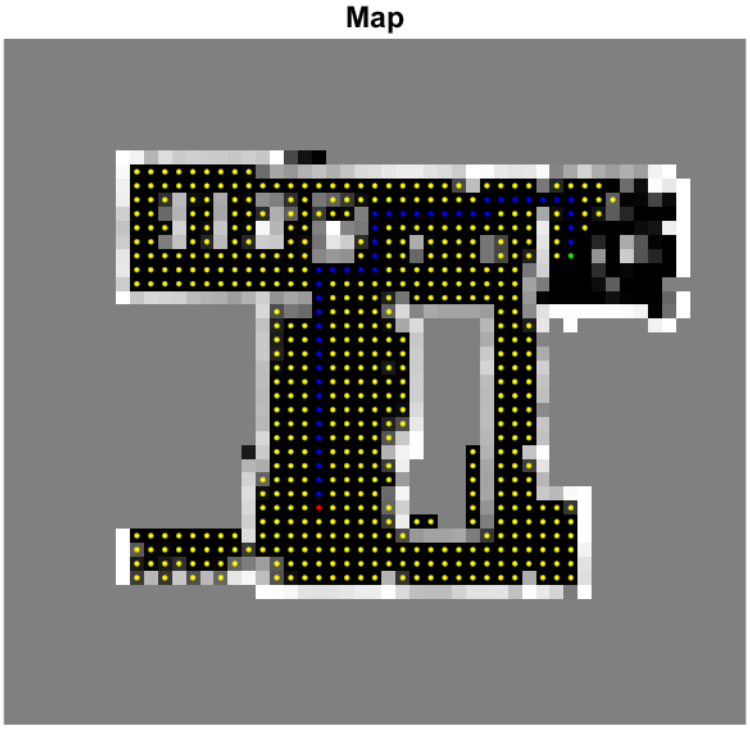

Figura 2.4: Resultado de algoritmo de Dijkstra en el caso en que se aumentó significativamente el costo de los nodos diagonales

Adicionalmente se observa como en ninguno de los casos estudiados, el algoritmo se posiciona sobre los bordes del mapa o lo que parecería ser un obstáculo. Esto se debe al umbral de 0.4 configurado al momento de calcular los costos de los nodos hijos.

2.2. Algoritmo A*

A continuación se muestra el resultado de modificar el algoritmo de Dikjstra agregándole información sobre la posición del nodo al que se quiere llegar. La heurística consta de simplemente agregar un costo proporcional a la distancia euclidiana del nodo destino al nodo actual, esto es.

$$g\left(\begin{bmatrix} y \\ x \end{bmatrix}, \begin{bmatrix} y_g \\ x_g \end{bmatrix}\right) = k_A \cdot \left\| \begin{bmatrix} y \\ x \end{bmatrix} - \begin{bmatrix} y_g \\ x_g \end{bmatrix} \right\| \tag{2.2}$$

Para lograr que el algoritmo sea óptimo se debería utilizar una heurística admisible, es decir que la heurística utilizada sea igual o menor al costo del camino óptimo. En este caso se está utilizando la distancia recta que es admisible en el espacio euclidiano.

A continuación se muestra el resultado de aplicar el algoritmo A* con distintos valores de k_A. En la siguiente figura se observa el resultado con $k_A = 1$. Se puede observar como los primeros nodos de costo mínimo que son explorados se encuentran en una disposición que une el nodo destino con el nodo de inicio.

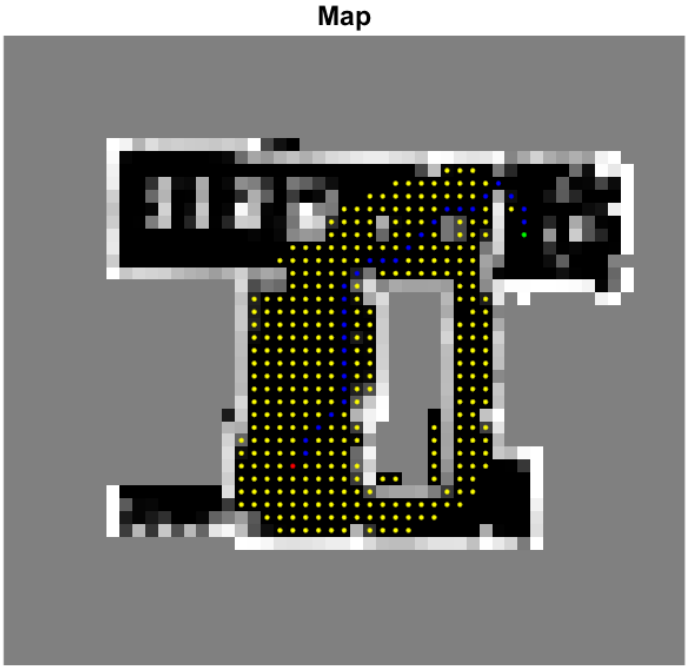

Figura 2.5: Algoritmo A* utilizando una constante de $k_A = 1$

Se observó que sucede cuando se ejecuta el algoritmo con $k_A = 2$, $k_A = 5$ y $k_A = 10$, El resultado se muestra en las siguientes figuras en donde se puede ver como el algoritmo es más agresivo a la hora de elegir los nodos de costo mínimo a explorar. Además para valores más elevados de k_A se observa como la trayectoria obtenida hace pasar al robot por al lado de la pared. Esto sucederá para cualquier entorno con cierta concavidad en donde los nodos más cercanos al nodo destino tendrán menor costo y por lo tanto serán preferidos por el algoritmo.

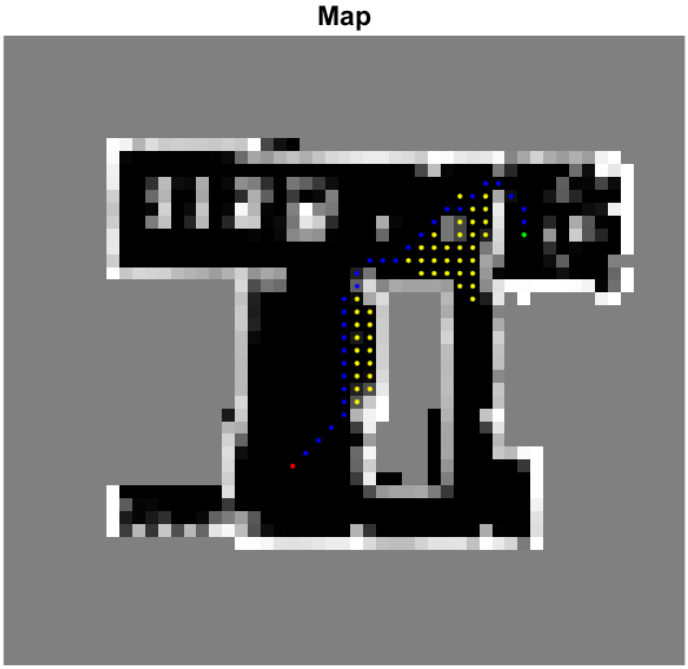

Figura 2.6: Algoritmo A* utilizando una constante de $k_A = 2$

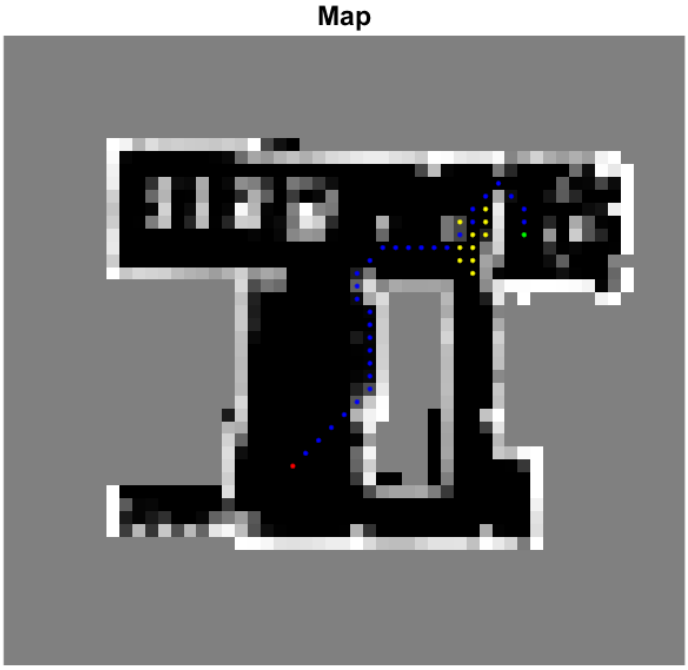

Figura 2.7: Algoritmo A* utilizando una constante de $k_A = 5$

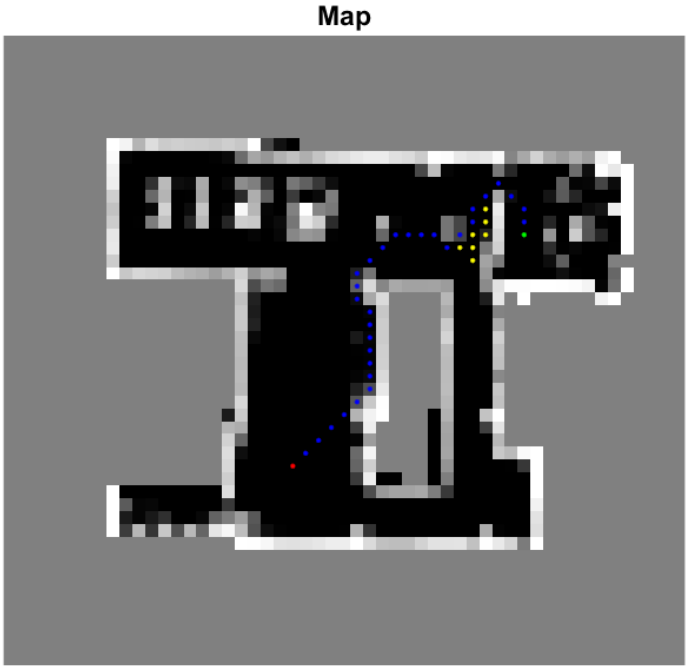

Figura 2.8: Algoritmo A* utilizando una constante de $k_A = 10$

Otra observación que se realiza sobre el algoritmo A* es la menor cantidad de nodos que se exploran hasta llegar al nodo destino cuando se consideran valores elevados de k_A. Esto obviamente produce una disminución en el procesamiento de la información y un valor de tiempo de convergencia hasta encontrar el nodo destino menor.

3. Conclusiones

Se logró ejecutar el algoritmo ICP a través de una correspondencia dada por los puntos más cercanos que permitió familiarizarse con el algoritmo y su función. También se logró ejecutar el algoritmo de Dijkstra y A* para lo cual se tuvieron que programar las funciones de costo y heurística. La programación de ambas funciones permitió entender el funcionamiento de ambos algoritmos y la manera en que se modifica el planeamiento del robot para llegar de un nodo origen a un nodo destino.