Universidad de Buenos Aires
Facultad De Ingeniería
Año 2021 - 1er Cuatrimestre

PRÁCTICA 5: ICP y Planeamiento
Robótica móvil- Un enfoque probabilistico(86.48).

1er. Cuatrimestre
Curso: MAS

INTEGRANTES:
Fuentes Acuña, Brian Alex - #101785
<bfuentes@fi.uba.ar>

Índice

1. Iterative Closest Point (ICP)

ICP es encontrar una transformación que alinea puntos de un mapa con otros, es decir machear un mapa con el siguiente de manera de con distintas mediciones de partes de mapas, armar un mapa completo.
Entonces dado dos conjuntos $X = \{x_1..X_{n_x}\}$ e $P = \{p_1...p_{N_p}\}$ el objetivo sería encontrar una roto-traslación que minimiza la suma de los errores cuadráticos.
Pero es posible si y solo si las correspondencias punto a punto son correctas, en caso contrario el algoritmo no funciona.
Teniendo esto en cuenta, se deben calcular los centros de masa de cada conjunto y restárselos así formando los nuevos conjuntos $X' = \{xi - \mu_x\}$ y $P' = \{pi - \mu_p\}$.
Para así poder encontrar una descomposición en valores singulares para W defino por

$$W = \sum_{i=1}^{Np} x_i' p_i'^T$$

donde su DVS es:

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

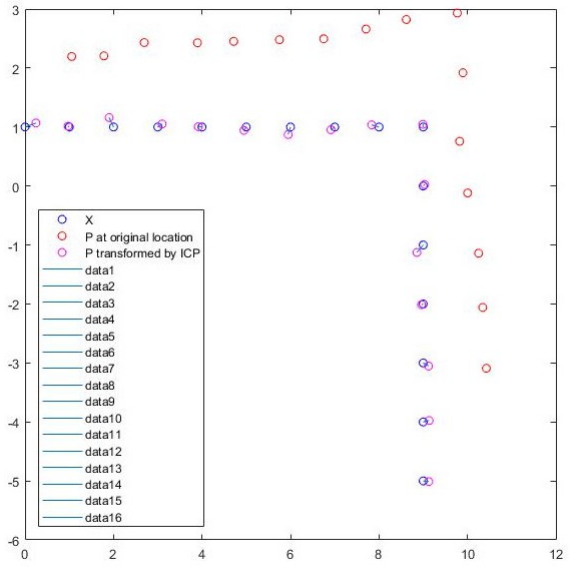
Y se sabe que si $rango(W) = 3$, la solución óptima que minimiza el error es única y está dada por la siguiente rotación y traslación:

$$\begin{aligned} R &= UV^T \\ t &= \mu_x - R\mu_p \end{aligned} \tag{1}$$

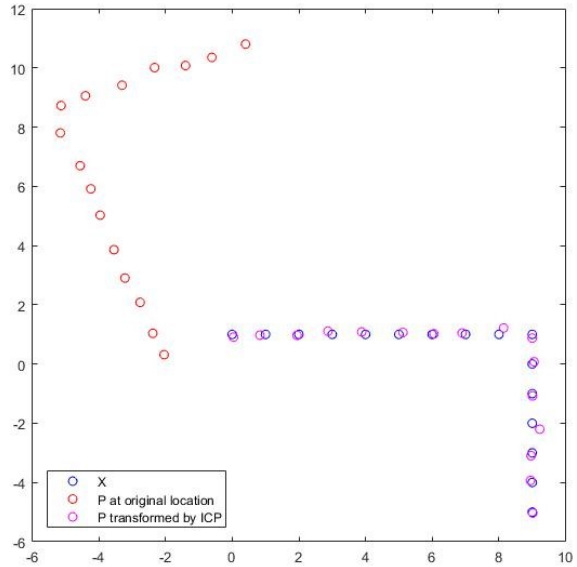
Entonces el problema de ICP estaría resuelto. solo que esta el detalle de las correspondencias punto a punto y es justamente por eso que el algoritmo es iterativo, ya que para conocer las correspondencias se debe buscar los puntos de forma que minimice el error y así calcular la roto-traslación relativa luego iterar minimizando el error.
Esto convergerá si las posiciones iniciales están lo suficientemente cerca.
Entonces el algoritmo formalmente seria:

- Determinar los puntos que se corresponden
- Calcular rotación R, translación t por SVD
- Aplicar R y t a los puntos de conjunto a ser registrado
- Calcular el error E(R,t)
- Si el error decrece y error ¿ umbral
- Repetir los pasos anteriores
- Sino
- Finalizar y devolver la alineación final

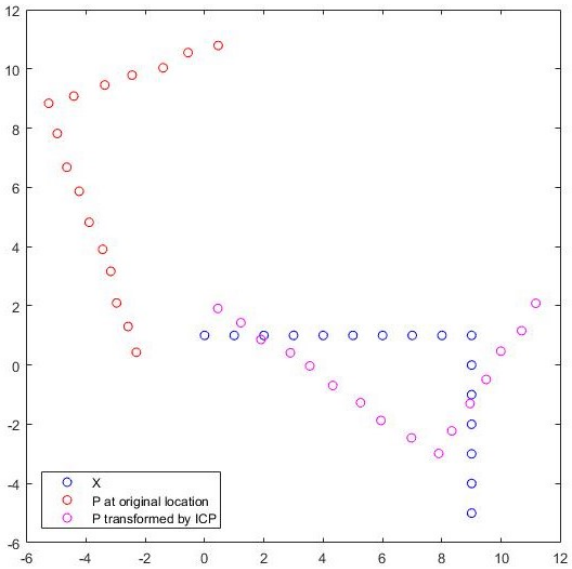
Luego se prueba el algoritmo primeramente con $P = P_1$ y $P = P_2$ con roto-traslaciones conocidas y macheados correctamente.



(a) P1: Conocido



(b) P2: Conocido

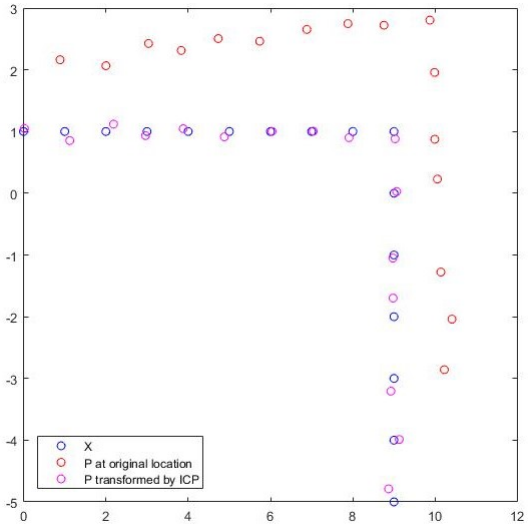


(c) P2: Conocido

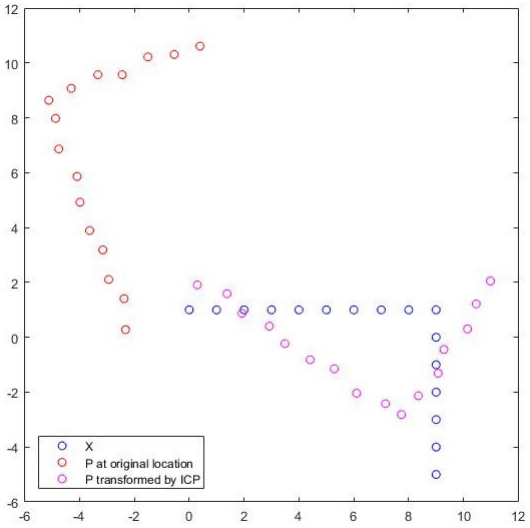
Figura 1: Matcheo con ICP para roto-traslaciones conocidas y desconocidas

Se puede observar que para la Figura-1a y Figura-1b efectivamente la solución converge y minimiza la varianza exitosamente, pero en la Figura-1c la solución que minimiza el error cuadrático no dio lo esperado puesto que para el caso de $P = P2$ las posiciones iniciales está muy alejas respecto a la original, esto hace muy difícil ICP converja a una buena solución y puede funcionar bien la mayoría

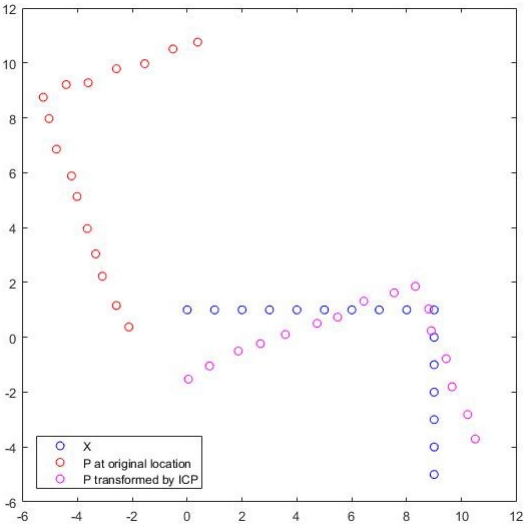
de las veces teniendo en cuenta que no puede haber posiciones iniciales muy alejadas de las originales. Luego se generar P3 y P4 alejados aleatoriamente respecto de P y se permutan los puntos, así se genera nuevos conjuntos de los que no se sabe la correspondencia con P ni la roto-traslación aplicada.



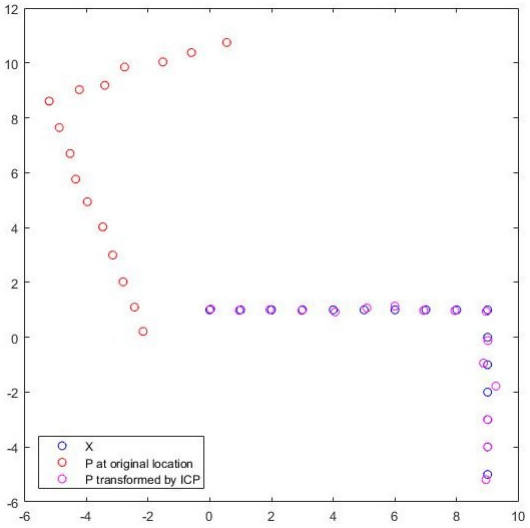
(a) Desconocido



(b) Desconocido



(c) Desconocido



(d) Desconocido

Figura 2: Matcheo con ICP para roto-traslaciones conocidas y desconocidas

Observar la figura-2a esta es similar a la anterior puesto que se generó teniendo en cuenta P1 pero aleatoriamente para así mostrar como con posiciones iniciales pequeñas respecto de P, ICP converge a una buena solución, pero para la figura-2b y c, la convergencia no es buena es decir ICP falla y esto no es porque no funciones ICP sino porque no se respetan las hipótesis iniciales (posiciones iniciales pequeñas) pero aun así ICP puede llegar a converger a una buena solución Figura-2d pero es poco probable.

2. Planeamiento de caminos

Los algoritmos de búsqueda en grafos como Dijkstra o A* pueden ser usados para planear caminos en grafos desde un lugar de inicio hasta un objetivo. Si las celdas de un mapa de grilla se representan como nodos conectados con sus celdas vecinas, estos algoritmos pueden aplicarse directamente para realizar planeamiento para robots. Para este ejercicio, consideramos las 8 celdas vecinas de una celda h_x, y_i, que se definen como las celdas adyacentes a h_x, y_i horizontalmente, verticalmente y en diagonal.

2.1. Algoritmo Dijkstra

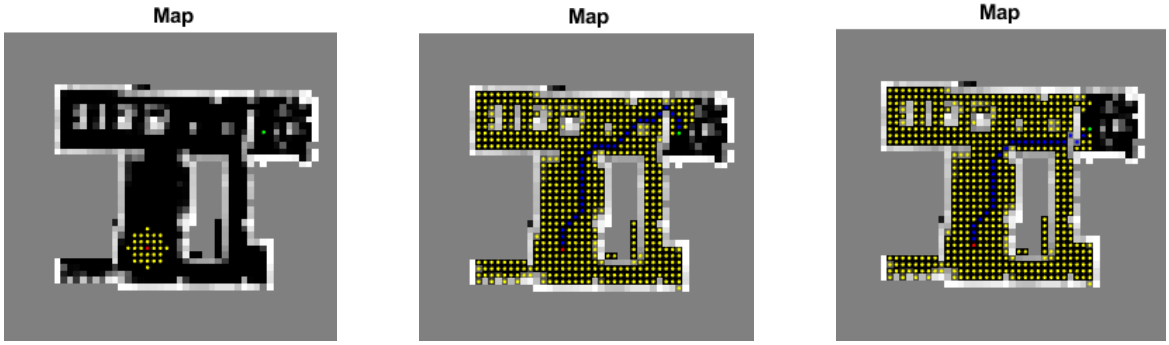
El algoritmo de Dijkstra se usa para calcular caminos de costo mínimo dentro de un grafo. Durante la búsqueda, siempre se busca el nodo del grafo con el menor costo desde el punto de inicio y se agregan los nodos vecinos al grafo de búsqueda.
La idea más general de algoritmo es:

1. Crear un conjunto de nodos no visitados con una distancia tentativa inicial. En particular, el nodo de origen se establece en cero mientras que los demás son infinitos.

2. Se considera todos los vecinos no visitados del nodo actual (inicialmente el nodo de origen) y se actualiza sus distancias tentativas (teniendo en cuenta la probabilidad del mapa).

3. Se marca el nodo actual como visitado y nunca se volverá a verificar. Si se llega a visitar el nodo de destino, el algoritmo se detiene. Caso contrario, se selecciona el nodo con la distancia tentativa más pequeña como el nuevo nodo actual y se vuelve al paso anterior.

En la Figura 3 se muestra cómo funciona el algoritmo Dijkstra implementado con un mapa de probabilidades 2-D en donde se ve como en la figura-3a se van calculando los vecinos y estos van incrementando a medida que se calculan sus costos, siempre quedándose con el menor, luego en la figura-3b termina conociendo gran parte del mapa hasta finalmente encontrar la meta. Luego tiendo en cuenta los costos el algoritmo es capaz de generar el camino de menor costo.



(a) Iniciando la busqueda, k= 10

(b) Meta encontrada, camino optimo generado, k= 10

(c) Meta encontrada, otro camino optimo generado, k= 1

Figura 3: Algortimo Dijkstra

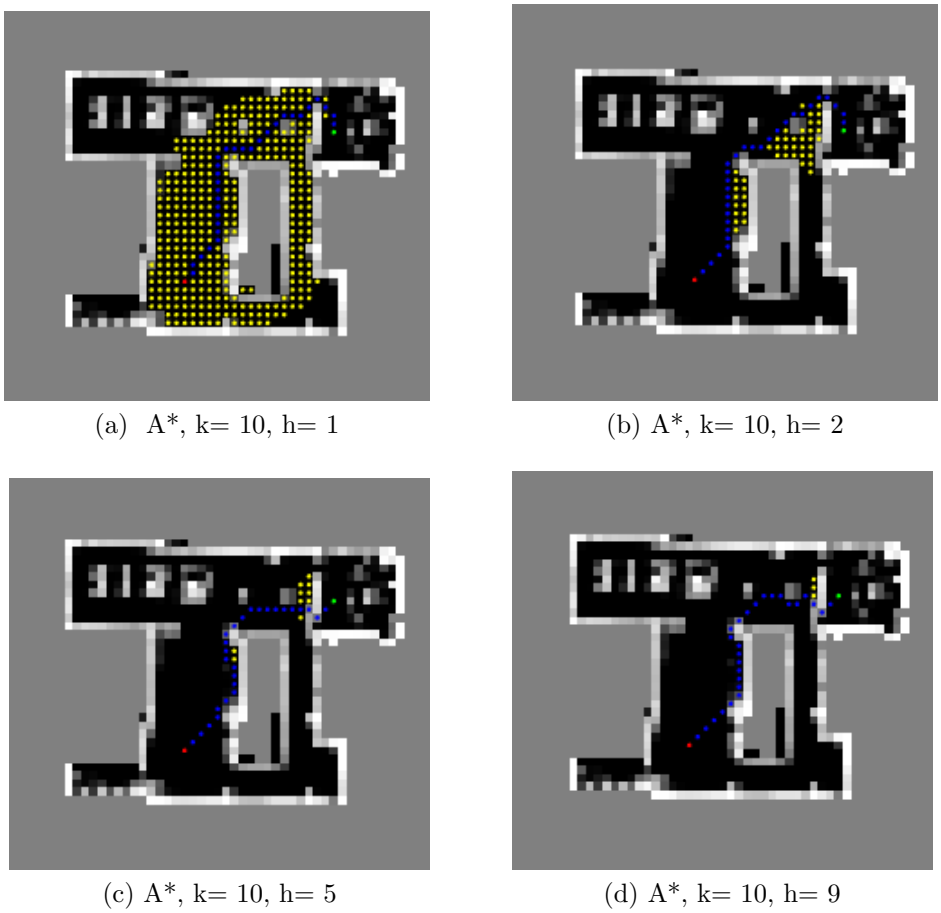
Este algoritmo es robusto y siempre encuentra la meta sin necesidad de conocerla, lo que a su vez hace que deba de buscarla en casi todo el mapa si es que este se encuentra muy alejado de la

posición de inicio.
Por supuesto es camino optimo depende de las probabilidades del mapa y esto se vio reflejado en el cálculo de los costos para los vecinos puesta este costo también es proporciona a dichas probabilidades de hecho $costo = norm(nodo - vecino) + (k = 10) * P(vecino)$ pero si K tiene un valor menor por ejemplo $K = 1$ el camino optimo puede ser totalmente distinto de hecho en la figura-3c se encuentra camino hacia la meta por lo que parece ser una puerta quizá medio abierta y por ello tiene mayor probabilidad de ocupación que la puerta abierta de arriba (caso-b).

2.1.1. Algortimo A*

El algoritmo A* es similar al Dijkstra solo que además utiliza una heurística para realizar una búsqueda informada que resulta ser más eficiente que el algoritmo de Dijkstra.
Pero esta heurística debe asegurar que A* sea optima, para ello la función heurística debe **ser admisible**, esto quiere decir que nunca debe sobrestimar el costo real para llegar a la meta.
Además, debe ser consistente esto quiere decir que el costo de un nodo hacia la meta no debe ser mayor que el de sus sucesores y los costos $[(nodo- > vecino) + (vecino- > meta) < (nodo- > vecino)]$.

En los siguientes grafico se implementa una heurística euclidiana que no es más que la norma de la diferencia de un nodo actual hacia la meta la cual participa en el costo según una constante h. con $K = 10$ para que tome la puerta con menor probabilidad de ocupación hacia la meta.



(a) A*, k= 10, h= 1

(b) A*, k= 10, h= 2

(c) A*, k= 10, h= 5

(d) A*, k= 10, h= 9

Figura 4: Algortimo A* variando la participacion de la heuristica en el costo para celdas vecinas

Para hacer un mejor análisis del algoritmo propuesto se variará la participación de la heurística en el costo para cada celda vecina, haciendo que a medida que aumenta h el algoritmo considere más la información de la ubicación de la meta en el cálculo de los costos, entonces de alguna forma se ponderan los pesos del mapa y la distancia hacia la meta.
En la figura-1a con $h = 1$ se puede notar la diferencia con el Dijkstra y es que al tener información de la meta la búsqueda del camino optimo se vuelve más rápida, haciendo que este sea más eficiente, numero de nodos visitados= 373.
Pero teniendo información de la meta se le podría dar más participación para el cálculo del costo, entonces se varia la constante h que pondera esta distancia euclidiana entre una celca y la meta.
Para $h = 2$ el tiempo de planeamiento se vio drásticamente reducido y esto es porque se deben calcular menos vecino y menos caminos alternativos, numero de nodos visitados= 75.
Para $h = 5$ la participación de la heurística es aún mayor y prácticamente es gobernada por ella pues la trayectoria es bastante directa y quizá lo más curioso es que aun con $k = 10$ se elige un camino distinto a los anteriores, justamente la que parece ser la "puerta de abajoïgnorando así el hecho de que "la puerta de arriba"tiene menor probabilidad de ocupación, numero de nodos visitados= 38.
Para $h = 10$ es similar al caso anterior solo que el camino es más directo, siendo la trayectoria muy cercana a los obstáculos, aunque al estar por cerca de las puerta pareciera que se dirige hacia la de arriba pero termina eligiendo la de menor distancia euclidiana (la de abajo) llegando aún más rápido a la meta, numero de nodos visitados= 31.

3. Conclusión

ICP es un algoritmo bastante interesante por su simpleza y los usos que se le puede dar aunque es necesario ejecutarlo sabiendo previamente las correspondencias y que la separación entre las mediciones sean pequeñas para así encontrar la rotación y traslación correctas, es decir con mínimo error.

Dijkstra es altamente confiable pero lento en comparación al de A*, por los nodos que debe recorrer, aunque su mayor ventaja es que no debe tener información de una heurística.
El algoritmo A* si bien se puede ver como una extensión del Dijkstra, es mucho más rápido, aunque es necesario información de la ubicación de la meta.
A medida que se aumenta h, aumenta la participación de la heurística y con ello puede ir mucho más rápido hacia la meta, pero esto no es del todo bueno ya que por ejemplo con el caso-d se puede ver como la trayectoria optima puede no ser la mejor dado que el robot debería pasar muy de cerca a los obstáculo que y estos tienen una probabilidad de ocupación distinta de cero, puede que realmente haya pequeño obstáculo en el camino optimo y realmente la distancia optima al principio de la trayectoria debería ser en realidad más directa.