Examen Parcial

Construcción de Robots Móviles, FI, UNAM, 2020-2

Nombre: Cruz Navarrete Victor Jesus

Instrucciones: Subir la solución en un solo archivo PDF con el nombre Examen.pdf, al repositorio en GitHub en la carpeta **Entregables**. **Fecha límite de entrega**: 04 de junio de 2020.

1.- Explique qué es la configuración, espacio de configuraciones y grados de libertad de un robot móvil.

Configuración: Descripción de todos los puntos en el espacio que ocupa el robot.

Espacio de configuraciones: Conjunto de todas las configuraciones posibles. Se describen mediante variedades, estas cumplen con las propiedades de un espacio euclidiano, pero no son un espacio euclidiano.

Grados de libertad: Número de parámetros independientes necesarios para para determinar una configuración. Si el robot se mueve en un plano, entonces tiene tres grados de libertad. Si el robot se mueve en el espacio 3D, el robot tiene seis grados de libertad.

2. Investigue dos métodos basados en grafos para planeación de rutas.

Método Dijkstra: El método de Dijkstra sirve para encontrar la ruta más corta desde un nodo origen a todos los demás nodos contenidos en el grafo ponderado G = [V, A]. Este algoritmo considera w(u, v) ≥ 0 por cada arista (u, v) ∈ A.

El algoritmo funciona de la siguiente manera: iniciando desde un vértice origen s, todos los nodos se etiquetan con su distancia al vértice origen d[v]. El algoritmo utiliza la técnica de relajación; inicialmente no se conocen las rutas, pero a medida que avanza el algoritmo y se encuentran las demás, las etiquetas pueden cambiar, reflejando mejores rutas. En un inicio, todas las etiquetas son temporales. Una vez que se descubre que una etiqueta representa la ruta más corta posible del origen a ese nodo, se vuelve permanente y no cambia más.

DIJKSTRA (G, w, s)
Entrada: requiere un vértice origen s y una matriz de distancias d donde $d[i, j]$ contiene la distancia desde el nodo i a j; $d[i, j] = \infty$ cuando no hay trayectoria directa desde i a j.
Salida: Regresa la distancia $d[v]$ desde s a cualquier otro vértice $v \in V$.

```
1:  for cada vértice v ∈ V do
2:      d[v] = ∞
3:  end for
4:  d[s] = 0
5:  S = 0
4:  while existan nodos que no han sido etiquetados permanentemente, V \ S ≠ 0 do
5:      u = argmin {d[v]}
8:      S = S ∪ u
9:      for cada arista (u, v) do RELAJACIÓN
10:         if d[v] > d[u] + w(u, v) then
11:             d[v] = d[u] + w(u, v)
12:         end if
13:     end for
14: end while
```

Los vértices etiquetados permanentemente se almacenan en S. La etiqueta permanente del nodo origen contiene d[s] = 0, ya que dist [s, s] = 0. Por cada vértice alcanzado v, una etiqueta temporal d[v] contiene la distancia desde el origen; durante el primer ciclo esta es simplemente la longitud de la arista desde s a v. Se asigna al vértice u la etiqueta temporal con la distancia más corta d[v]. Si no existe una ruta más corta para llegar al vértice v, su etiqueta se vuelve permanente. Para determinar cuál es la ruta más corta, el algoritmo utiliza relajación. Después de que el algoritmo de Dijkstra finaliza, todos los vértices contenidos en el grafo están etiquetados con la distancia más corta que se necesita recorrer a partir del origen s. Esas cantidades son usadas para obtener el recorrido desde el vértice final (t) al vértice origen.

Algunas características del método Dijkstra:

-Al igual que el método A*, nos da la distancia más corta y de menor costo

-Es un método óptimo y completo

-Usa cola con prioridad

-La función de costo es la cercanía

-Da la misma ruta que A*

-Es igual que el método Breadth-First-Search si la función de costo es 1

Método A*: es un algoritmo de búsqueda inteligente que busca el camino más corto desde un estado inicial a un estado meta a través de un espacio de problema, usando una heurística óptima. Este método ignora los pasos más cortos, en algunos casos rinde una solución subóptima.

A* es un algoritmo informado que basa su comportamiento en la evaluación de una función:

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

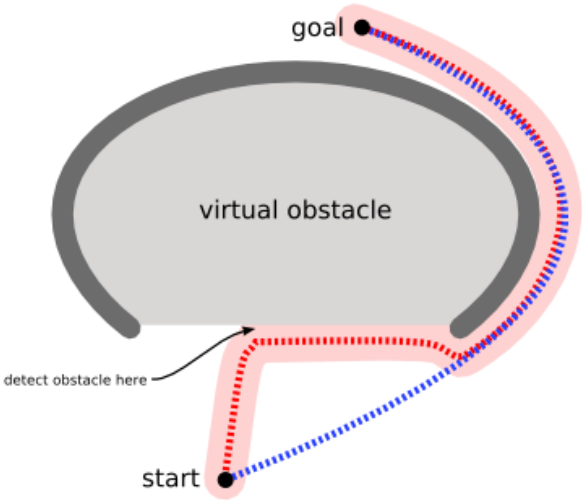
Donde:

g(n) es el costo de las movidas realizadas.

h(n) es la función heurística.

A* representa el costo estimado del mejor camino. Esta estimación debe ser realizada por defecto, siempre menor o igual a la real. En búsqueda de caminos, la función heurística suele ser el camino recto hacia la meta, ya que no importa como sea el mapa, es imposible que exista camino de costo menor.

El modo de realizar el cálculo de la distancia necesaria para llegar a la meta depende del tipo de movidas permitidas. Si solo podemos movernos vertical y horizontalmente podremos realizar el cálculo de la distancia de Manhattan, que consiste en sumar la cantidad de bloques en horizontal y vertical que restan para llegar a la meta. Si además se permiten movidas diagonales, debemos aplicar el principio de Pitágoras y el cálculo será la raíz cuadrada de la suma del cuadrado de los catetos.



Vemos que el algoritmo trata de llegar del punto de inicio (start) hasta el punto meta (goal). Como se usa el método A*, la ruta más corta a la meta se ve en color azul.

El movimiento de un punto a otro es simple, pero la búsqueda de rutas optimas es más complejo.

Algunas características del método A*:

-Nos da la distancia más corta de menor costo

-Es un método óptimo y completo (garantiza llegar a la meta)

-Expande menos nodos en comparación con otros métodos

-Si lo comparamos con el método de Dijkstra, A* se ejecuta más rápido

-Nos da la misma ruta que el método Dijkstra

-Toma el nodo de menor costo, comúnmente llamado nodo “F”

3. Investigue dos métodos basados en muestreo para planeación de rutas.

Método RRT: Consiste en construir un árbol de exploración que cubriera uniformemente todo el espacio libre de colisión. El algoritmo tiene como misión seleccionar un punto de forma aleatoria y extender el árbol de configuraciones. Esta función tiene el cometido de expandir el árbol conforme el punto aleatorio lo marca, si es que hay camino libre.

El algoritmo RRT se basa en la construcción de un árbol de configuraciones que crece buscando a partir de un punto de origen a partir de un punto de origen.

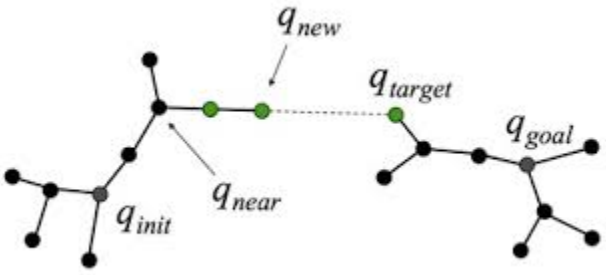
El algoritmo comienza inicializando la tabla asociada al árbol con la configuración de origen. Luego, se entra en un bucle, limitado por un valor Kmax, cuya función es finalizar el algoritmo una vez que se haya realizado un número prefijado de iteraciones. Este valor se utilizará posteriormente para parar el algoritmo en caso de que no se alcance la configuración final.

Dentro del bucle del algoritmo RRT hay dos instrucciones. Con la primera se obtiene un punto al azar dentro de un espacio libre de colisión; la segunda hace crecer al árbol en dirección a la configuración aleatoria anteriormente obtenida.

El crecimiento del árbol se consigue por la función extiende. La estructura de dicha función comienza con el cálculo del punto cercano. Esto se realiza mediante la función VectorMásProximo que aplica la métrica R definida anteriormente a todos los vértices del árbol, obteniendo los puntos más cercanos al punto aleatorio. Seguidamente, la función nueva configuración calcula un nuevo punto a agregar, mediante un salto de tamaño ε partiendo del punto cercano al punto aleatorio. Para la obtención de un punto nuevo se debe considerar que si hay una colisión en dicho desplazamiento, devolviendo verdadero o falso según sea un movimiento posible o colisione con algún obstáculo.

```
RRT (qini)
{
  Arbol[0] = qini;
  Para k = 1 hasta Kmax
  {
    qrand = ConfiguraciónAleatoria();
    Extiende(Arbol, qrand);
  }
  Devuelve Arbol;
}
```

```
Función Extiende(Arbol, qrand)
{
  qnear= VecinoMásPróximo(qrand, Arbol);
  Si ( NuevaConfiguración(qrand, qnear, qnew) )
  {
    AñadeVértice(Arbol, qnew);
    Si ( qnew=qrand )
    {
      Devuelve alcanzado;
    }
    Si no
    {
      Devuelve avanzado;
    }
  }
  Si no
  {
    Devuelve rechazado;
  }
```



Método bayesiano dinámico: El robot integra incrementos de posición (x,y) y de orientación (θ), medidos por contadores en sus partes móviles del robot. Estas mediciones permiten estimar pequeños incrementos de posición.

El modelo de movimiento se puede formular a partir de las posiciones del robot por odometría. Supongamos la posición anterior del robot en el tiempo t-1

$$\left(\hat{x}_{t-1}, \hat{y}_{t-1}, \hat{\theta}_{t-1}\right)$$

y la posición siguiente del robot en el tiempo t

$$\left(\hat{x}_t, \hat{y}_t, \hat{\theta}_t\right)$$

Tenemos el desplazamiento lineal y angular, por odometría:

$$\Delta_{t-1}\hat{d} = \sqrt{(\hat{x}_t - \hat{x}_{t-1})^2 + (\hat{y}_t - \hat{y}_{t-1})^2} \qquad \Delta_{t-1}\hat{\theta} = \hat{\theta}_t - \hat{\theta}_{t-1}$$

Se puede definir una variable aleatoria Δt-1d que define el desplazamiento real del robot dado Δt-1 medido mediante la odometría. Su función de densidad se define como una distribución normal de media Δt-1 y desviación típica σd .

$$p(\Delta_{t-1}d \mid \Delta_{t-1}\hat{d}) = \frac{1}{\sqrt{2\pi}\sigma_d} \exp\left(-\frac{(\Delta_{t-1}d - \Delta_{t-1}\hat{d})^2}{2\sigma_x^2}\right)$$

Una vez definida la variable aleatoria Δt-1d, es posible formular las posiciones del robot en el instante t, xt e yt , y como variables aleatorias calculadas a partir de Δt-1d y las posiciones del instante anterior.

$$\begin{aligned} x_t &= x_{t-1} + d\cos(\theta_{t-1}) \\ y_t &= y_{t-1} + d\sin(\theta_{t-1}). \end{aligned}$$

4. Explique en qué consiste el proceso de SLAM (Simultaneous Localization and Mapping).

El proceso del SLAM consiste en un cierto número de pasos: extracción de características, asociación de datos, estimación del estado y actualización de las características. La finalidad del proceso es usar el entorno para actualizar la posición del robot. La odometría del robot no es 100% confiable, por lo que es necesario el uso de sensores.

Esto se consigue extrayendo características del entorno y re-observándolas mientras el robot se mueve. Un filtro extendido de Kalman es responsable de actualizar la estimación de la posición del robot basándose en estas características, a las que son bien conocidas como landmarks.

El filtro mantiene un estimado de la incertidumbre en la posición del robot y también de los landmarks que el robot ha visto en su entorno.

Los puntos de referencia son características del entorno que pueden ser fácilmente distinguidas y reobservadas. Son utilizadas por el robot para saber en dónde se encuentra.

Para que un punto se considere un buen landmark, debe cumplir con lo siguiente:

- Debe ser posible observarla desde diferentes posiciones y ángulos.
- Debe ser lo suficientemente única como para poder ser distinguida de otras características.
- Si decidimos un punto que sea landmark, esta debe permanecer en un estado estacionario.
- Aquellos puntos en donde se asigne un landmark, deberán ser suficientes, para que el robot no circule durante largos periodos de tiempo y además se pueda perder.

5. Explique en que consiste la localización mediante filtros de partículas, sus ventajas sobre el Filtro de Kalman y los paquetes de ROS que lo implementan.

Consiste en la sustitución el movimiento probabilístico apropiado y los modelos perceptivos en el algoritmo de filtros de partículas, este representa la creencia bel(xt) por un conjunto de partículas.

$$Mxt = \{x_t^1, x_t^2, \cdots, x_t^M\}.$$

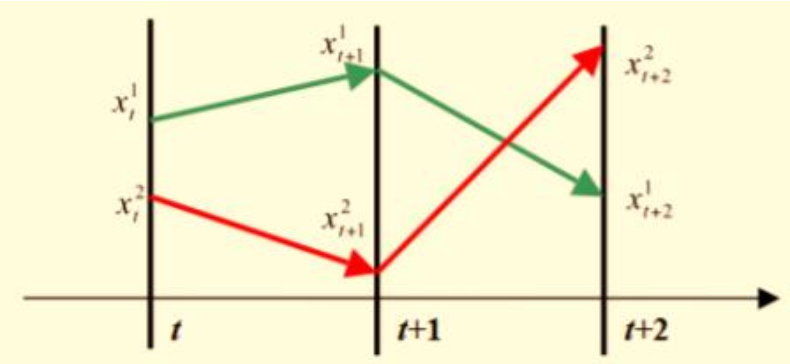
El modelo de medición se aplica en línea para determinar el peso de la partícula. La creencia inicial bel(x0) se obtiene al generar M partículas al azar de la distribución previa p(x0), ya asignado el factor de importancia uniforme M − 1 a cada partícula.

Es sencillo aplicar el algoritmo para el escenario de localización basado en puntos de referencia.

Una estrategia común para establecer M es mantener el muestreo hasta que lleguen el siguiente para ut y zt.

A medida que se adquiere la posición, las partículas en otros lugares probablemente no desaparezcan prácticamente. En algún momento, las partículas solo "sobreviven" cerca de una posición única, y el algoritmo no puede recuperarse si esta postura resulta ser incorrecta.

La idea subyacente es crear unas hipotéticas trayectorias del estado (dejando propagar “las partículas”) y a cada una de ellas atribuir unos pesos según mejor expliquen las observaciones. La estimación final del estado se hará con una media de las trayectorias ponderada por los pesos.



Las ventajas que presentan sobre el filtro Kalman es que no pide restricciones en las funciones deterministas y tampoco pide ninguna específica distribución para los ruidos. Se supone que hay que conocer las formas analíticas de las funciones determinísticas y de las propiedades estadísticas.

Los paquetes de ROS que implementan los filtros de partículas son:

• Coreslam

• Slam_gmapping

• Stageros

6. Investigue que son los campos potenciales y explique los pasos generales para implementarlos.

El método de campo de potencial es conocido como un método local, ya que solo calcula el movimiento para una configuración inicial-final dada. Comienzan en la inicial e intentan mover el robot en pequeños pasos hacia el objetivo.

Este método esencialmente trata el ambiente de trabajo como un campo de fuerzas, dado un valor diferente a cada objeto sobre este. El objetivo o punto de llegada se considera como un imán de polaridad contraria a la del robot, es decir que tiene una fuerza atractiva mientras que los obstáculos se comportan como imanes de igual polaridad emitiendo fuerzas repulsivas. Este método considera la velocidad del móvil en la cercanía del obstáculo.

Para desarrollar el método de los campos potenciales es necesario realizar los siguientes pasos:

a. Considerar que cada obstáculo genera una fuerza repulsiva normal a la superficie sobre la que se realiza el planteamiento y que la posición objetivo es una fuerza de atracción.

b. Calcular el vector de fuerzas resultante.

c. Calcular una nueva posición para el robot como resultado de aplicar una fuerza aceleradora.

d. Regresar al paso a.

7. Explique que es una transformación homogénea y para que se utiliza en robots móviles.

La transformada homogénea es un método matemático que consiste en la expresión de un punto p y cuyo vector P es , expresándolo en base a un nuevo sistema de referencia trasladado y rotado, con respecto al original (o fijo).

$$\bar{p} = \bar{o}_n + \bar{p}_1$$

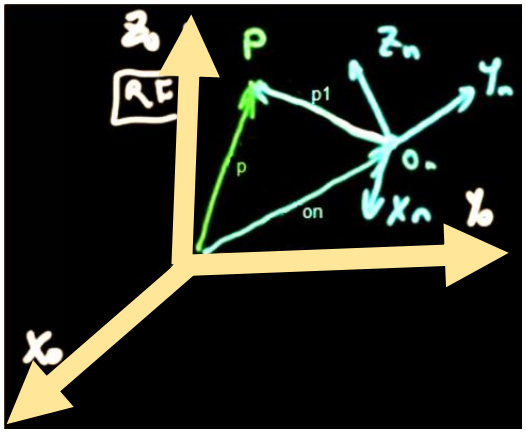
$$\bar{p} = \bar{o}_n + R_n \cdot \bar{p}_1$$

O_n: traslación

$R_n \cdot \bar{p}_1$: cambio de orientación

R_n:matriz ortogonal y con vectores columnas perpendiculares y unitarios

Deberá existir un operador único y compacto, que se encargue de las operaciones anteriores. Matriz de transformación homogénea.



$$H_n^o = \begin{bmatrix} \text{Cambio de orientación (3x3)} & \text{Traslación (3x1)} \\ (0,0,0) & 1 \end{bmatrix}$$

$$H_n^o = \begin{bmatrix} R_n & \bar{p}_1 \\ 0 \;\; 0 \;\; 0 & 1 \end{bmatrix}$$

$$H_n^o = \begin{bmatrix} R_n & \bar{p}_1 \\ \bar{o}^T & 1 \end{bmatrix}$$

Esta matriz u operador tiene referencia al origen fijo. Nótese que: $H_n^{-1} \neq H_n^T$

Este operador se aplica en la robótica en el modelado de manipuladores, modelado de robots articulados y en el control dinámico (fuerzas) del robot.

8. Investigue qué es un robot con restricciones no holonómicas de movimiento.

Que el robot puede moverse instantáneamente adelante o atrás pero no lateralmente por el deslizamiento de las ruedas.

Se puede interpretar matemáticamente como $G(p, \dot{p}, t) = 0$

Restricción holonómica no depende de $\dot{p}$.

$$\dot{x} = R \cdot \dot{\theta}$$

$$\int \mathrm{d}t \rightarrow x - R \cdot \theta = cte$$

La restricción no holonómica depende de $\dot{p}$ y no es integrable.

$$-\dot{x} \sin \phi + \dot{y} \cos \phi = \dot{\theta} \cdot R$$

$$\dot{x} \cos \phi + \dot{y} \sin \phi = 0$$