

# Tarea 03

## Representación del ambiente\*

Robots Móviles (TSM I, TSM II, TSCR), FI, UNAM, 2025-2

Nombre: \_\_\_\_\_

### 1. Actividades

1. Abra el archivo `catkin_ws/src/navigation/map_augmenter/scripts/map_inflater.py` y pegue el siguiente código en la línea 31 para implementar el algoritmo de inflado de mapas:

```
31 for i in range(len(static_map)):
32     for j in range(len(static_map[0])):
33         if static_map[i,j] == 100:
34             for k1 in range(-inflation_cells, inflation_cells):
35                 for k2 in range(-inflation_cells, inflation_cells):
36                     inflated[i+k1, j+k2] = 100
37
```

Agregue su nombre completo en la línea 18.

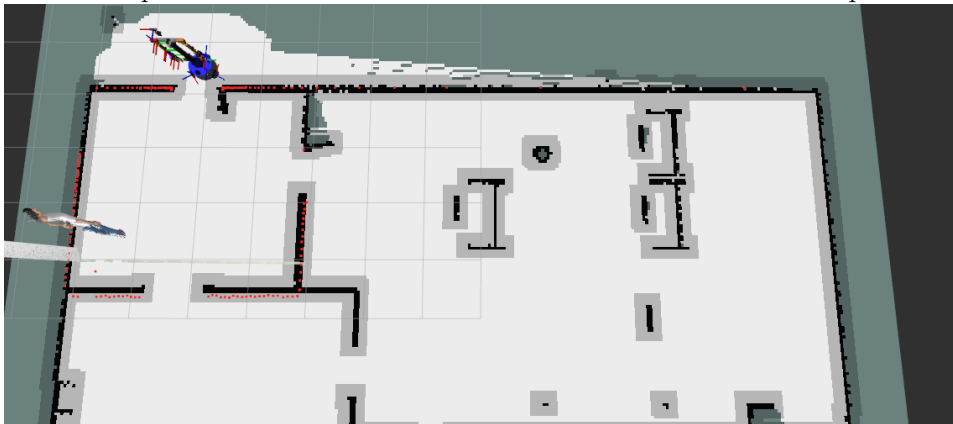
2. Abra una terminal y corra la simulación con el comando (revise las instrucciones en el README del repositorio):

```
1 roslaunch fiat_lux movement_planning.launch
2
```

3. En otra terminal, corra el inflado de mapas con el comando:

```
1 rosrun map_augmenter map_inflater.py _inflation_radius:=0.2
2
```

Observe lo que sucede en el visualizador RViz. Se debería visualizar el mapa inflado como se muestra en la figura:



4. Detenga el programa (Ctrl + C) y ejecútelo nuevamente con radios de inflado diferentes, por ejemplo:

---

\*Material elaborado con apoyo del proyecto PAPIME PE112525

```

1   rosrun map_augmenter map_inflater.py _inflation_radius:=0.3
2

```

Observe qué sucede. ¿Qué pasaría si el radio de inflado es muy grande o muy pequeño?

- Abra el archivo `catkin_ws/src/navigation/map_augmenter/scripts/split_and_merge.py` y pegue el siguiente código en la línea 45 para implementar la parte de ajuste de líneas del algoritmo *split and merge*:

```

45 if len(points) < min_points:
46     return lines
47 rho, theta, xm, ym, length = adjust_line(points)
48 idx, dist = find_farthest_point(points, rho, theta)
49 if dist < threshold:
50     return [[rho, theta, xm, ym, length]]
51 lines1 = split(points[0:idx], threshold, min_points)
52 lines2 = split(points[idx+1:len(points)], threshold, min_points)
53 lines = lines1 + lines2

```

- En la línea 64 del mismo archivo, pegue el siguiente código para implementar la mezcla de líneas parecidas del algoritmo *split and merge*:

```

64 if len(lines) < 2:
65     return lines
66 for i in range(1, len(lines)):
67     rho1, theta1, xm1, ym1, length1 = lines[i]
68     rho2, theta2, xm2, ym2, length2 = lines[i-1]
69     e_rho = abs((rho1 - rho2)/min(rho1, rho2))
70     e_theta = abs(theta1 - theta2)
71     if e_rho < rho_tol and e_theta < theta_tol:
72         new_lines.append([(rho1+rho2)/2, (theta1+theta2)/2, (xm1+xm2)/2, (ym1+ym2)/2,
73                             length1+length2])
74     else:
75         new_lines.append([rho1, theta1, xm1, ym1, length1])
76         new_lines.append([rho2, theta2, xm2, ym2, length2])
77 return new_lines

```

- En otra terminal, corra el algoritmo *split and merge* con el comando:

```

1   rosrun map_augmenter split_and_merge.py _dist:=0.01 _points:=3 _rho:=0.05 _theta
2   :=0.05

```

- Detenga la detección de líneas y ejecútela nuevamente con diferentes parámetros de sintonización. Con la GUI, mueva el robot a diferentes puntos del mapa para verificar la correcta sintonización de dichos parámetros. Se deberían observar segmentos de línea coincidentes con las paredes, como se muestra en la figura:



- En otra terminal, corra el algoritmo *Brushfire* para obtener un Diagrama de Voronoi Generalizado a partir del mapa inflado (el inflado de mapas debe estar corriendo):

```
1  rosrun map_augmenter gvd.py
2
```

Debería observar un GVD como el de la figura:



10. Detenga el GVD y abra el archivo `catkin_ws/src/navigation/map_augmenter/scripts/gvd.py` y en las líneas 53 y 56 cambie la distancia Euclídeana por distancia de Manhattan (revise los comentarios arriba de cada línea).
11. Ejecute nuevamente la obtención del GVD y observe lo que sucede en el visualizador RViz.

## 2. Entregables

- Código modificado en la rama correspondiente
- Documento impreso con los siguientes puntos:
  - Captura de pantalla del inflado de mapas con una vista diferente a la mostrada en este documento.
  - Capturas de pantalla de la extracción de líneas resultante con varios conjuntos de parámetros. Incluir una discusión sobre los parámetros que mejor hayan funcionado.
  - Captura de pantalla del GVD funcionando con una vista diferente a la mostrada en este documento.

No es necesario pegar el código modificado en el documento escrito. Para eso está el repositorio en línea.

## 3. Evaluación

Para la evaluación se utilizará la siguiente lista de cotejo:

- **[0 puntos]** Los cambios al código se subieron a la rama correspondiente con las credenciales de cada alumno. Este punto es requisito para evaluar los demás.
- **[1 punto]** El programa de inflado de mapas funciona correctamente.
- **[1 punto]** Captura de pantalla del inflado de mapas con una vista diferente a la mostrada en este documento.
- **[1 punto]** El programa de extracción de líneas funciona correctamente.
- **[1 punto]** Al menos una captura de pantalla de la extracción de líneas funcionando.
- **[1 punto]** Se incluye más de una captura de pantalla de la extracción de líneas, con diferentes parámetros de sintonización.

- **[1 punto]** Se incluye una discusión sobre cómo elegir los parámetros del algoritmo *split and merge*.
- **[1 punto]** El programa para la obtención del GVD funciona correctamente.
- **[1 punto]** Captura de pantalla del algoritmo *brushfire* funcionando (se muestra el GVD) utilizando distancia euclídeana.
- **[1 punto]** Captura de pantalla del algoritmo *brushfire* funcionando (se muestra el GVD) utilizando distancia de Manhattan.
- **[1 punto]** Se incluye una discusión sobre el uso de las distancias Euclídeana y de Manhattan para la obtención del GVD.