Práctica 1 Sistemas Expertos

Jordi Blasco Lozano

Razonamiento y representación del conocimiento

Grado en Inteligencia Artificial

Indice:

1. Objetivo de la practica

El objetivo fue crear un sistema experto para controlar un robot móvil en un plano horizontal, haciendo que recorra un segmento de manera rápida y precisa, utilizando funciones para regular su velocidad lineal y angular. Se implementaron dos versiones: una con reglas normales y otra con lógica difusa para mejorar la precisión y la capacidad de respuesta del robot.

El robot debía moverse desde un punto aleatorio del mapa hasta un punto final. Si el robot se encontraba en un segmento lineal, debía seguir la línea lo más cerca posible. Si estaba en un segmento de tipo triángulo, debía rodear el área triangular para sumar más puntos.

2. Consideraciones

Para realizar la práctica se tuvieron en cuenta varias consideraciones comunes para ambas versiones, además de consideraciones específicas para cada una:

- **Alineación inicial**: Al inicio de cada objetivo lineal, el robot debía dirigirse al punto más cercano a la recta para alinearse rápidamente con el segmento.
- **Velocidad constante**: El robot mantendría siempre la velocidad lineal máxima, y el programa se encargaría de ajustar solo la velocidad angular. En los segmentos triangulares, se redujo la velocidad máxima en 0.2 unidades en la versión fuzzy y en 0.1 unidades en la versión normal, para darle tiempo suficiente al robot para girar y llegar al punto final del triángulo.
- **Permitir giros mas bruscos**: El robot podrá girar de manera mas brusca en los triangulos en el modelo fuzzy para que llegue al punto final del segmento a altas velocidades.

3. Lógica Común

Para que el robot se alineara y siguiera los segmentos lineales, se implementó la siguiente lógica:

- **Cálculo del Target Point**: Se creó una función que calcula un punto en la recta, basado en un parámetro k entre 0 y 1. Esta función devuelve un punto proporcionalmente ubicado entre el punto de inicio y el punto final del segmento. Esto facilita la integración de la lógica para los puntos objetivo relativos que usa el código.

- **Seguimiento del Target Point**: En cada iteración, el robot realiza una búsqueda del target point deseado. Si el robot está lejos de la línea, el target point será el punto más cercano a la línea hasta que se acerque. Una vez cerca de la línea, el valor de k se incrementa gradualmente, guiando al robot a lo largo de la línea. Cuando el robot está alineado, sigue un punto objetivo que se mueve dentro del segmento para que siga la línea justo por el medio. Si el target point se fijara directamente en el punto final una vez alineado, el robot no sumaría suficientes puntos debido a desviaciones acumuladas, lo cual descubrí gracias a una función de "logs" que imprimía la distancia del robot a la línea en cada iteración. Aunque el logs en tiempo real afectaba el rendimiento, dejé la función para pruebas (aunque no es recomendable usarla ya que disminuye el rendimiento).

- **Anticipación del Giro**: En ambas versiones (simple y fuzzy) implementé un FACT_ANTICIPACION_GIRO, un factor que ajusta el valor de k para que el robot anticipe los giros. Esto asegura que el giro no ocurra demasiado tarde, evitando un deslizamiento causado por la desaceleración angular. En la versión normal, este factor cambia dependiendo del segmento para maximizar la puntuación. Si se desea usar un único factor constante, se puede desactivar la variable MAXIMIZACION_DE_ESTE_EJERCICIO.

- **Segmentos Triangulares**: Para los segmentos de tipo triángulo, usé una lógica diferente. Utilicé las mismas funciones de seguimiento de línea, pero agregué un target intermedio entre el inicio y el fin. En la versión normal, este target es el punto medio del triángulo, mientras que en la versión fuzzy se desplaza una unidad en la dirección perpendicular al segmento. Esto se hizo porque noté que la versión fuzzy podía mejorar este aspecto, y dado que la versión normal daba menos puntos con este cambio, decidí mantenerla sin modificaciones.

- **Tolerancias con el target**: Para los segmentos lineales, la tolerancia del target final siempre se mantuvo en 0.5, tal como indicaba el enunciado de la práctica. Sin embargo, para los segmentos triangulares, modifiqué la tolerancia del target intermedio a 3 unidades para permitir una mayor flexibilidad, ya que no era el principal objetivo del recorrido. Para la tolerancia final del triángulo, también utilicé la establecida por la práctica.

---

4. Cálculo de la Aceleración Angular

Esta parte es donde más difieren ambas versiones del sistema experto.

Cálculo del Error Angular

El error angular es la diferencia entre el ángulo al que el robot debería dirigirse y el ángulo actual del robot. Se calcula utilizando las coordenadas del objetivo relativo (target_point) y la posición del robot (poseRobot). Primero se determina el ángulo hacia el punto objetivo (angulo_a_target) usando la función atan2 para los componentes delta_x y delta_y entre el objetivo y la posición del robot. Luego, se resta la orientación actual del robot (theta), la cual está en radianes. La fórmula es:

error_angular = angulo_a_target – theta

El valor resultante se normaliza para que esté en el rango [-π, π] con la siguiente expresión:

error_angular = (error_angular + π) % (2 * π) - π

Este error angular indica cuán desalineado está el robot respecto a la dirección ideal hacia el objetivo. Un error positivo indica que el robot debe girar hacia la derecha, mientras que un error negativo indica que debe girar hacia la izquierda. Esta señal se usa para calcular la velocidad angular necesaria para corregir la trayectoria del robot.

El calculo de este error se usa tanto en la version simple como en la fuzzy.

Calculo W en el modelo experto simple

En el modelo lógico simple, W (la velocidad angular) se calcula utilizando una ganancia proporcional (k_w) y el error angular, que es la diferencia entre el ángulo deseado y la orientación actual del robot. La fórmula utilizada es la siguiente:

velocidad_angular_deseada = WMAX * tanh(k_w * error_angular)

El valor de k_w determina la sensibilidad del robot al error angular. En este caso, se utilizó un valor de k_w = 2.0, lo que significa que el robot reacciona de forma más agresiva a grandes errores y suavemente a errores pequeños. Este enfoque busca garantizar una corrección rápida de la orientación sin excesivos cambios de direccion.

Para limitar la aceleración, se compara la velocidad_angular_deseada con la velocidad_angular_previa. El cambio máximo permitido en la velocidad angular está limitado por la constante WACC (aceleración angular máxima). De este modo, la variación de velocidad angular (delta_w) se calcula como:

delta_w = min(max(velocidad_angular_deseada - velocidad_angular_previa, -WACC), WACC)

Finalmente, la nueva velocidad angular se obtiene sumando delta_w a la velocidad angular previa, asegurando así que el cambio en la velocidad sea suave y manejable para el robot:

velocidad_angular = velocidad_angular_previa + delta_w

Este método garantiza que el robot gire de manera controlada y que no haya cambios bruscos que podrían causar inestabilidad.

Calculo W en el modelo experto fuzzy

En el modelo fuzzy, W (la velocidad angular) se calcula utilizando un sistema de inferencia difusa, que tiene en cuenta el error angular para determinar la acción de control más adecuada. El objetivo es que el robot corrija su orientación de forma suave y efectiva mediante el ajuste de la velocidad angular.

Variables Difusas

Para el modelo fuzzy, se definen dos variables principales:

Entrada (error_angular): La desviación del robot respecto al ángulo deseado. Esta entrada tiene varios términos de pertenencia: "NegativoGrande", "NegativoPequeño", "Cero", "PositivoPequeño" y "PositivoGrande". Estos términos permiten representar la desviación angular en diferentes niveles, de modo que el sistema pueda diferenciar entre grandes errores y errores menores.

Salida (velocidad_angular): La acción de control que el robot debe realizar. Los términos de la velocidad angular incluyen "FuerteIzquierda", "Izquierda", "Recto", "Derecha" y "FuerteDerecha", que representan la dirección y magnitud del ajuste que debe realizar el robot.

Reglas Difusas

Las reglas difusas relacionan las entradas (error angular) con la salida (velocidad angular). Estas reglas son de la forma "Si el error angular es X, entonces la velocidad angular es Y". Por ejemplo:

Si el error angular es "Cero", entonces la velocidad angular es "Recto".

Si el error angular es "PositivoGrande", entonces la velocidad angular es "FuerteDerecha".

Estas reglas ayudan a convertir el error angular en una acción concreta que permita al robot corregir su rumbo.

Defusificación: El resultado final del proceso de inferencia es un conjunto difuso. Para convertirlo en un valor concreto de velocidad angular (W), se usa el método del centroide (cog - center of gravity). Esto significa que el valor final se obtiene calculando el centro de masa del área bajo la curva resultante, lo que proporciona un valor ponderado que tiene en cuenta todas las reglas activadas y sus grados de pertenencia.

Limitación de Aceleración y Ajuste Final

Después de calcular el valor difuso de la velocidad angular (W_fuzzy), se aplica una limitación de aceleración similar al modelo lógico simple. Se asegura que el cambio en la velocidad angular (delta_w) no exceda la aceleración angular máxima (WACC). Luego, la nueva velocidad angular se calcula como:

velocidad_angular = velocidad_angular_previa + delta_w

Este proceso garantiza que el robot realice los giros de manera controlada y sin cambios bruscos que podrían causar inestabilidad en su movimiento.

5. Resultados y conclusiones

Tras probar muchas veces ambos códigos, he llegado a las siguientes conclusiones:

El modelo difuso es mucho más inestable. Dependiendo de la ejecución, el modelo difuso puede hacer entre 37 y 45 puntos. No estoy seguro de a qué se debe esto, pero influye mucho la manera en la cual el sistema conecta desde el triángulo 2 al segmento 3. Sin embargo, el sistema no difuso es mucho más estable, ya que siempre obtiene una puntuación cercana a los 43 puntos.

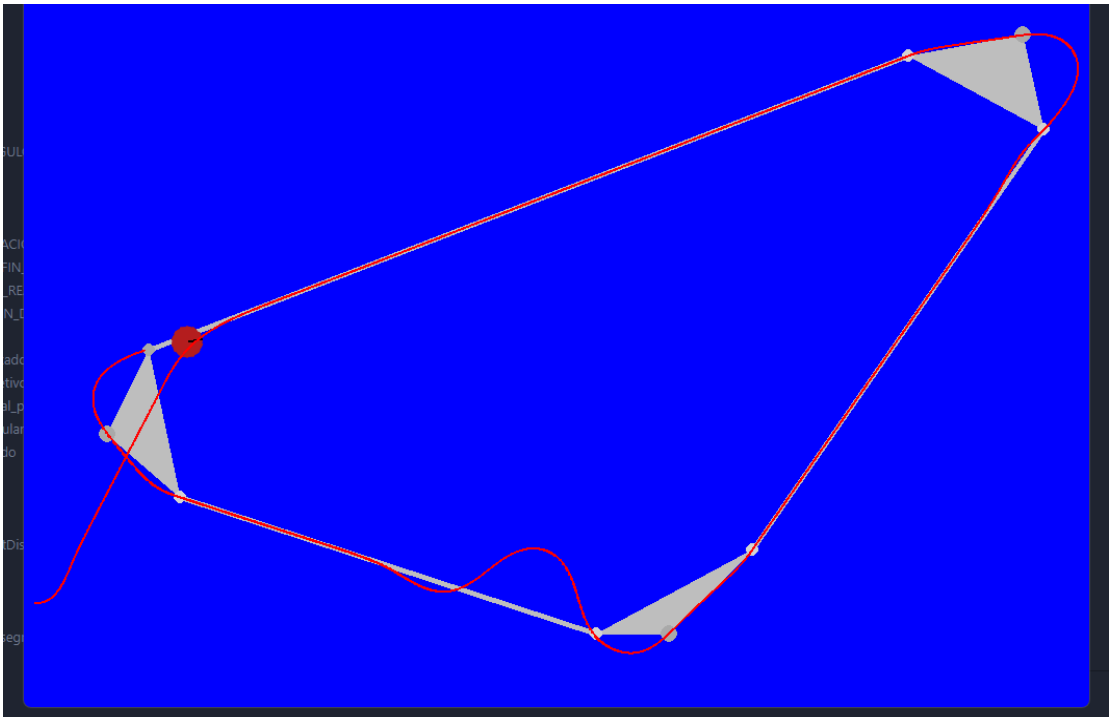
El sistema difuso es mucho mejor a la hora de conectar las trayectorias de salida de los triángulos con la entrada a los segmentos lineales, pero el sistema no difuso realiza un seguimiento impecable de los segmentos lineales, consiguiendo sumar muchos más puntos en estos tramos. Es decir, para un problema base en el cual el robot tuviera que seguir varias líneas en muchas posiciones distintas, el sistema fuzzy obtendría ventaja. Sin embargo, si solo hubiera que seguir una línea recta, el modelo simple superaría al modelo fuzzy en puntos. Esto se debe a que el sistema difuso calcula la velocidad angular en base a variables ya definidas, mientras que el modelo simple calcula exactamente cuál debe ser la velocidad angular para reducir el error en cada iteración.

Ejemlos:

Captura 1: modelo no fuzzy

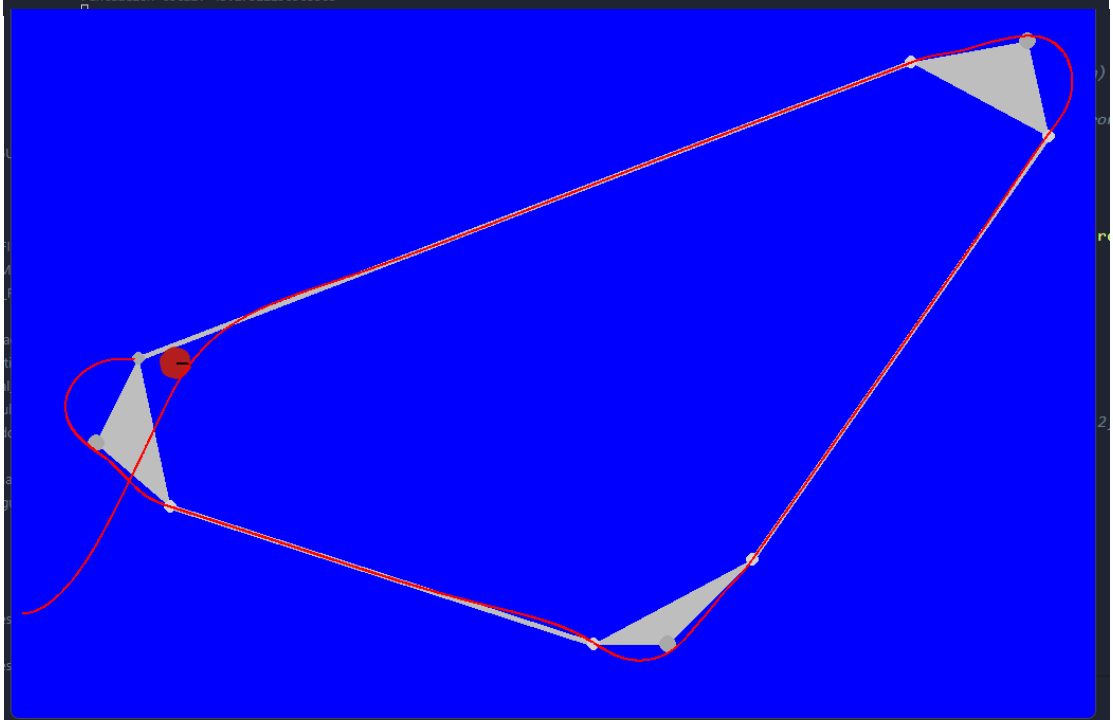
Captura 2: modelo fuzzy

1)



```
PS C:\Users\jordi\Documents\IA\IA\2doAnyo\representacionConocimiento\practica1\v1.5_robot> python .\P1Launcher.py
pygame 2.6.1 (SDL 2.28.4, Python 3.9.13)
Hello from the pygame community. https://www.pygame.org/contribute.html
Puntuación del objetivo: 2.656082622217861. Puntuación de distancia: 133892.09434170544 en 35.940643072128296 segundos
Puntuación del objetivo: 5.2829007168566005. Puntuación de distancia: 495 en 8.679799795150757 segundos
Puntuación del objetivo: 14.870667468437222. Puntuación de distancia: 76694.98024850749 en 16.277417421340942 segundos
Puntuación del objetivo: 7.870535844754229. Puntuación de distancia: 477 en 6.784971714019775 segundos
Puntuación del objetivo: 5.000966804098728. Puntuación de distancia: 23852.424058512162 en 15.832922220230103 segundos
Puntuación del objetivo: 7.494068453290994. Puntuación de distancia: 499 en 7.160024166107178 segundos
Puntuación total: 43.17522190965563
```

2)



```
PS C:\Users\jordi\Documents\IA\IA\2doAnyo\representacionConocimiento\practica1\v1.5_robot> python .\P1Launcher.py
pygame 2.6.1 (SDL 2.28.4, Python 3.9.13)
Hello from the pygame community. https://www.pygame.org/contribute.html
Puntuación del objetivo: 1.5438254783967391. Puntuación de distancia: 77235.9481552705 en 35.84742069244385 segundos
Puntuación del objetivo: 5.467661685536073. Puntuación de distancia: 499 en 8.553212881088257 segundos
Puntuación del objetivo: 11.565201941414083. Puntuación de distancia: 59647.66864513379 en 16.277467489242554 segundos
Puntuación del objetivo: 7.899905773870291. Puntuación de distancia: 499 en 6.947660684585571 segundos
Puntuación del objetivo: 12.947636226691 65. Puntuación de distancia: 44122.75466991733 en 14.048413276672363 segundos
Puntuación del objetivo: 5.60430714471418. Puntuación de distancia: 500 en 8.445480108261108 segundos
Puntuación total: 45.02853825062301
```