

# Práctica 1

# Sistemas

# Expertos

Jordi Blasco Lozano

Razonamiento y representación del conocimiento

Grado en Inteligencia Artificial

---

## Indice:

<b>Indice:</b>	<b>2</b>
<b>1. Objetivo de la practica</b>	<b>3</b>
<b>2. Consideraciones</b>	<b>3</b>
<b>3. Lógica Común</b>	<b>4</b>
<b>4. Cálculo de la Aceleración Angular</b>	<b>6</b>
<b>Cálculo del Error Angular</b>	<b>6</b>
<b>Calculo W en el modelo experto simple</b>	<b>7</b>
<b>Calculo W en el modelo experto fuzzy</b>	<b>8</b>
<b>5. Resultados y conclusiones</b>	<b>9</b>

---

## 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 línea para alinearse rápidamente con el segmento.
- **Velocidad constante:** El robot siempre mantendría 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 0.1 unidades en la versión normal para darle tiempo suficiente al robot para girar y llegar al punto final del triángulo. En la versión fuzzy, se realizaron ajustes adicionales para maximizar la puntuación.

- **Conexión precisa entre segmentos:** Para conseguir una conexión precisa entre los segmentos triangulares y los lineales, se implementaron tres técnicas:
    1. **Anticipación del Giro personalizada** (aplicada a ambas versiones, explicada más adelante).
    2. **Cambios manuales y precisos** en las entradas y salidas del sistema fuzzy para ajustar la velocidad angular y asegurar que el robot girara más rápido o mantuviera la dirección recta con distintos niveles de error angular.
    3. **Uso de un target intermedio personalizado** para cada triángulo, facilitando una transición recta hacia cada segmento lineal. (explicada más adelante)
- 

### 3. Lógica Común

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

- **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:** Implemente un `FACT_ANTICIPACION_GIRO`, un factor que ajusta el valor de `k` para que el robot anticipe los giros, aumentando la distancia del target point. Esto asegura que el giro no ocurra demasiado tarde, evitando un deslizamiento causado por la desaceleración angular. El valor de 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_ESTO_EJERCICIO`.
- **Segmentos Triangulares:** Para los segmentos de tipo triángulo, usé una lógica diferente. Utilicé las mismas funciones de seguimiento de línea, pero agregué un punto objetivo fijo intermedio entre el inicio y el fin. En la versión normal, este punto objetivo es el punto medio del triángulo, mientras que en la versión fuzzy se desplaza '`x`' unidades en la dirección perpendicular al segmento inicio-fin y '`y`' unidades en la dirección paralela al segmento inicio-fin hacia el punto inicial. Esto se hizo porque noté que la versión fuzzy mejoraba considerablemente este aspecto, pasando de una puntuación máxima de 47 a 67 (al encontrar el punto exacto donde el robot debía comenzar a girar para conectar de la mejor manera con la recta siguiente). Dado que la versión normal obtenía 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 fijo 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:

$$\text{error\_angular} = \text{angulo\_a\_target} - \text{theta}$$

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

$$\text{error\_angular} = (\text{error\_angular} + \pi) \% (2 * \pi) - \pi$$

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:

$$\text{velocidad\_angular\_deseada} = W_{MAX} * \tanh(k_w * \text{error\_angular})$$

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 dirección.

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:

$$\Delta w = \min(\max(\text{velocidad\_angular\_deseada} - \text{velocidad\_angular\_previa}, -WACC), WACC)$$

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:

$$\text{velocidad\_angular} = \text{velocidad\_angular\_previa} + \Delta w$$

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 ( $W_{\text{fuzzy}}$ ), 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 ( $W_{\text{ACC}}$ ). Luego, la nueva velocidad angular se calcula como:

$$\text{velocidad\_angular} = \text{velocidad\_angular\_previa} + \Delta w$$

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 ambos códigos en numerosas ocasiones, he llegado a las siguientes conclusiones:

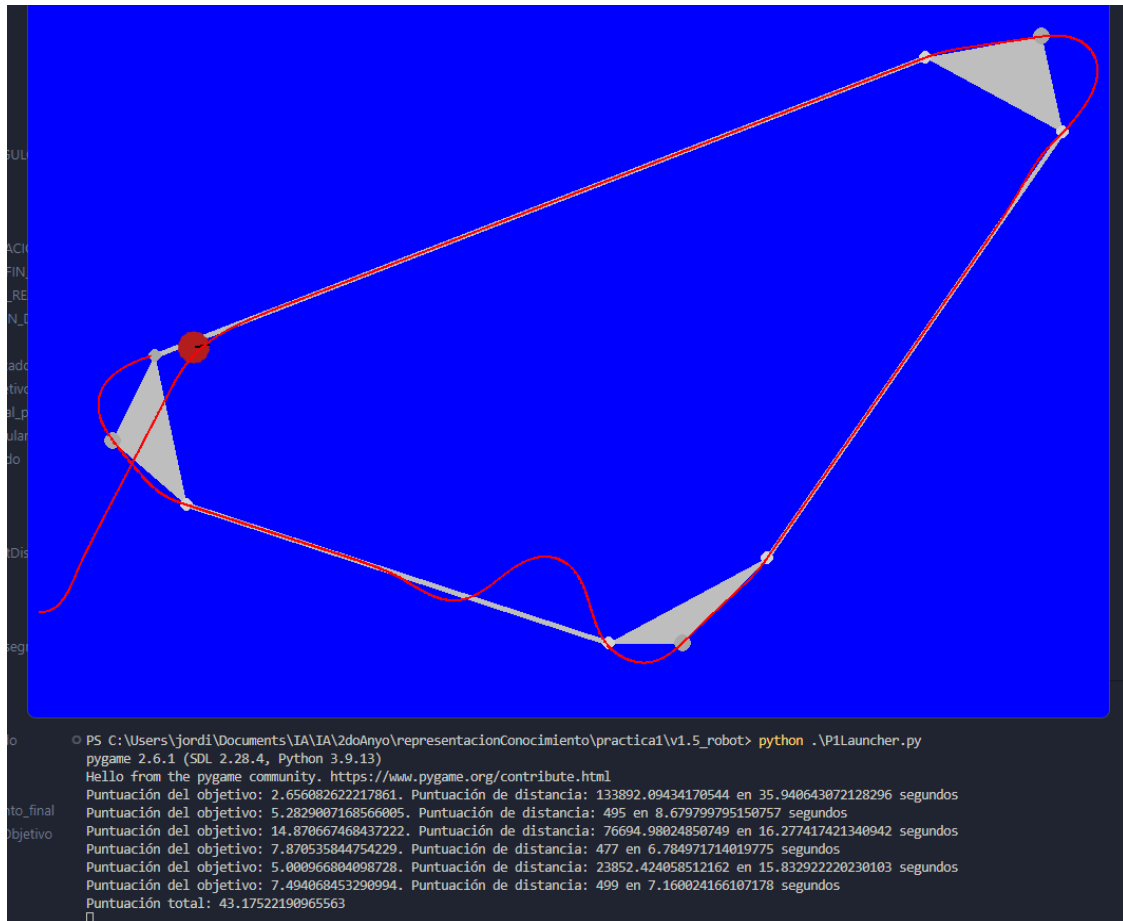
El modelo difuso es mucho más fácil de mejorar. Con algunos ajustes simples, se puede lograr que el robot siga el segmento de manera mucho más recta. Además, el modelo difuso se adapta mucho mejor a los cambios en tiempo real. Por otro lado, el modelo convencional, a menos que se tengan todos los segmentos previamente calculados, tiene dificultades para superar al sistema difuso. Para que esto ocurra, sería necesario tener la capacidad de calcular la trayectoria perfecta de cada tipo de segmento o de todos ellos simultáneamente.

El sistema difuso también resulta más efectivo al conectar las trayectorias de salida de los triángulos con la entrada a los segmentos lineales. No obstante, el sistema convencional realiza un seguimiento impecable de los segmentos lineales. En resumen, para un problema donde el robot deba seguir múltiples líneas en diferentes posiciones, el sistema difuso tiene una clara ventaja. Sin embargo, si solo se tratara de seguir una línea recta, el modelo simple podría superar al modelo difuso en términos de puntuación, debido a su mayor precisión en los cálculos. Esto dependería de cuán precisas sean las variables difusas definidas. Esto se debe a que el sistema difuso calcula la velocidad angular en función de variables ya establecidas, mientras que el modelo simple calcula exactamente la velocidad angular necesaria para reducir el error en cada iteración. **Ejemplos:**

Captura 1: modelo no fuzzy (43)

Captura 2: modelo fuzzy (67)

1)



2)

