UNIVERSIDADE DA CORUÑA

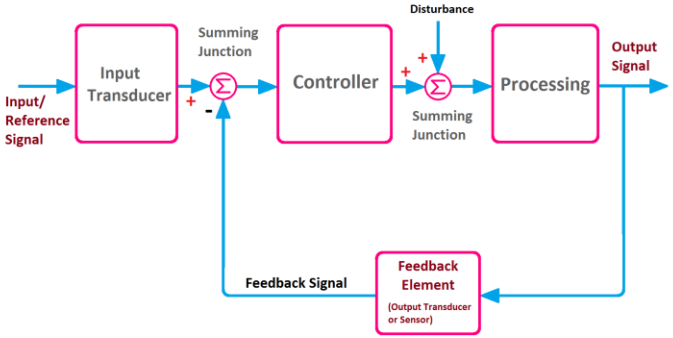
Robobo

Ejercicios con python

Aplicando el control autónomo: Control Proporcional y PID

Introducción

- Los sistemas de control son fundamentales en la robótica inteligente.
- Permiten que los robots reaccionen a su entorno y se adapten a cambios.
- El control retroalimentado ayuda a mejorar precisión, estabilidad y autonomía.
- El control PID es ampliamente utilizado en robótica para ajustar movimientos y navegación.



Closed Loop Control System Block Diagram

Control Proporcional

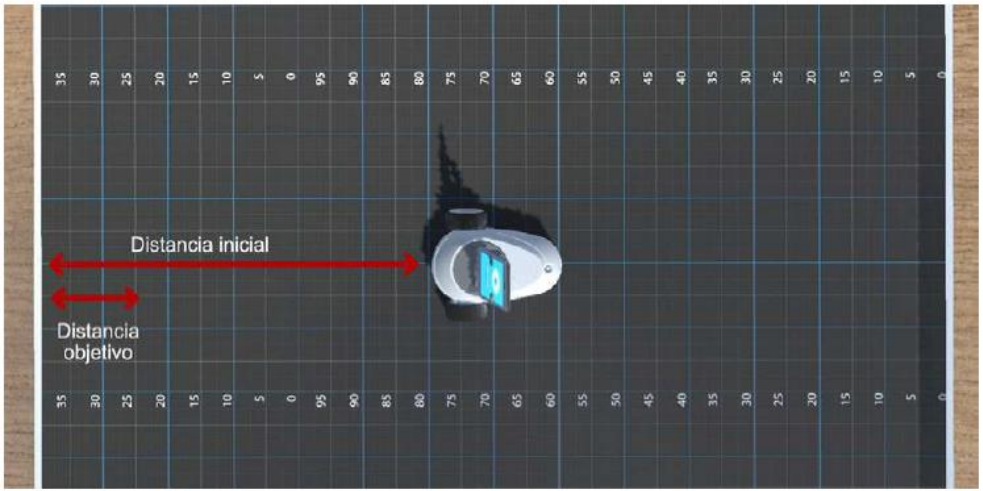
- Es el más básico de los controles retroalimentados.
- La salida del sistema es proporcional al error.
- Se utiliza una constante de proporcionalidad (Kp).
- Fórmula:

$$u(t) = K_p \cdot e(t)$$

donde K_p es la ganancia proporcional y $e(t)$ es el error.

Ejemplo de control Proporcional

- Un robot avanza en línea recta hacia una pared y debe reducir la velocidad proporcionalmente a la distancia, llegando a detenerse sin chocar.



Ejemplo de control Proporcional

- El **error del sistema** será la diferencia entre la distancia objetivo y la distancia inicial (actual).
- La **respuesta** del sistema será la velocidad que se aplicará a las ruedas del robot.
- Código básico en Python:

```
goal = 130
error = goal - rob.readIRSensor(IR.FrontC)
speed = error * Kp
```

- Se prueba con diferentes valores de Kp para optimizar el comportamiento.

Ejemplo de control Proporcional

- Partiendo del ejemplo donde el robot se acerca a velocidad constate a un obstáculo:

```
from robobopy.Robobo import Robobo
from robobopy.utils.IR import IR

rob = Robobo("localhost")
rob.connect()

spped = 20
goal = 100

rob.moveWheels(speed, speed)
while rob.readIRSensor(IR.FrontC) < goal:
    rob.wait(0.1)

rob.stopMotors()
rob.disconnect()
```

```
from robobopy.Robobo import Robobo
from robobopy.utils.IR import IR

rob = Robobo("localhost")
rob.connect()

kp = 1
goal = 100

while rob.readIRSensor(IR.FrontC) < goal:
    error = goal - rob.readIRSensor(IR.FrontC)
    speed = error * kp
    rob.moveWheels(speed, speed)
    rob.wait(0.1)

rob.stopMotors()
rob.disconnect()
```

Limitaciones del control Proporcional

- Si **Kp** es muy alto, el robot se mueve demasiado rápido.
- Si **Kp** es muy bajo, la respuesta es lenta.
- No tiene en cuenta la historia del error ni cambios bruscos.

Control PID

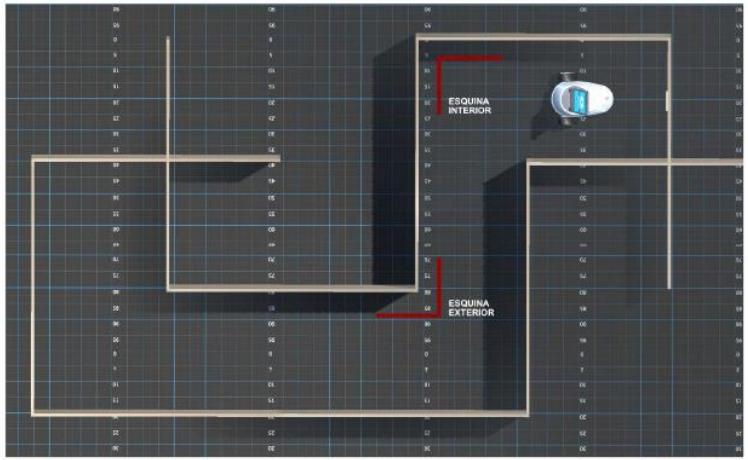
- Mejora la estabilidad y precisión del control proporcional.
- Se compone de:
  - **Proporcional (P):** Responde al error actual.
  - **Integral (I):** Considera la acumulación del error.
  - **Derivativo (D):** Considera la tasa de cambio del error.
- Fórmula:

$$u(t) = K_p \cdot e(t) + K_i \int e(t)d(t) + K_d \frac{de(t)}{dt}$$

donde K_p es la ganancia proporcional, K_i es la ganancia integral, K_d es la ganancia derivativa, y $e(t)$ es el error.

Ejemplo de control PID

- Un robot sigue una pared manteniendo una distancia constante.
- En primer lugar el robot avanza hasta localizar una pared (utilizando el código del ejemplo anterior).
- A continuación, la sigue indefinidamente dejándola a su derecha mientras realiza correcciones en la velocidad de sus ruedas para acercarse o alejarse de ella en función de la distancia.



Ejemplo de control PID

- El **error del sistema** será la diferencia entre la distancia a la que el robot se encuentra de la pared y la distancia objetivo.

```
error = rob.readIRSensor(IR.FrontRR) – wall_distance
```

- La **respuesta** del sistema será la velocidad que se aplicará a las ruedas del robot para corregir su trayectoria:
  - error > 0 → el robot se encuentra más cerca de la pared de lo que debería, por lo que debe girar levemente a la izquierda para alejarse.
  - error < 0 → el robot se encuentra más alejado de la pared de lo que debería, por lo que debe girar ligeramente a la derecha para acercarse.

```
right_speed = limit_speed(speed + correction, speed)
left_speed = limit_speed(speed - correction, speed)
```

  - Se establecen límites para evitar giros bruscos.

Ejemplo de control PID

- Código básico en Python:

```
# Inicialización de constantes y los errores integral y derivado
kp = 1.5
ki = 0.2
kd = 1
integral = 0
previous_error = 0
```

```
# Errores P, I y D a calcular durante el recorrido del robot
error = rob.readIRSensor(IR.FrontRR) - wall_distance
integral = integral + error
derivative = error - previous_error
correction = round(error * kp + integral * ki + derivative * kd)
```

Ejemplo de control Proporcional

- Bloque principal del ejemplo:

```
rob = Robobo("localhost")
rob.connect()
speed = 20
turn_speed = 5
front_distance = 80
goal = 50
kp = 1.5
ki = 0.2
kd = 1
integral = 0
previous_error = 0

go_to_wall(rob, front_distance)
```

```
while True:
    if rob.readIRSensor(IR.FrontC) >= front_distance:
        turn_degrees(rob, -90, turn_speed)
        integral = 0
        previous_error = 0
    else:
        error = rob.readIRSensor(IR.FrontRR) - goal
        integral = integral + error
        derivative = error - previous_error
        correction = round(error * kp + integral * ki +
derivative * kd)
        previous_error = error
        correction = limit_speed(correction, speed)
        right_speed = limit_speed(speed + correction, speed)
        left_speed = limit_speed(speed - correction, speed)
        rob.moveWheels(right_speed, left_speed)

    rob.wait(0.1)
```

Ejemplo de control Proporcional

- Funciones auxiliares:

```
def turn_degrees(robobo, degrees, speed):
    orientation = robobo.readOrientationSensor()
    # Se parte de la posición actual y se suman los grados a girar
    goal_angle = orientation.yaw + degrees
    # giro a la derecha
    if degrees > 0:
        turn_right(robobo, speed, goal_angle)
    # giro a la izquierda
    else:
        turn_left(robobo, speed, goal_angle)
    robobo.stopMotors()


def limit_speed(speed, max_speed):
    if speed < -2:
        if speed > -5:
            speed = -5
        elif speed < -max_speed:
            speed = -max_speed
    elif speed > 2:
        if speed < 5:
            speed = 5
        elif speed > max_speed:
            speed = max_speed
    return speed
```

Ejemplo de control Proporcional

- Funciones auxiliares:

```
def turn_right(robobo, speed, goal_angle):
    robobo.moveWheels(-speed, speed)
    # Si pasa de 180° --> continua en -180, -179, ...
    if goal_angle > 180:
        goal_angle = goal_angle - 360
        while 0 <= round(robobo.readOrientationSensor().yaw) <= 180:
            robobo.wait(0.001)
    while round(robobo.readOrientationSensor().yaw) < round(goal_angle):
        robobo.wait(0.001)


def turn_left(robobo, speed, goal_angle):
    robobo.moveWheels(speed, -speed)
    # Si pasa de -180° --> continua en 180, 179, ...
    if goal_angle < -180:
        goal_angle = goal_angle + 360
        while -180 <= round(robobo.readOrientationSensor().yaw) <= 0:
            robobo.wait(0.001)
    while round(robobo.readOrientationSensor().yaw) > round(goal_angle):
        robobo.wait(0.001)
```

Ejemplo de control Proporcional

- La función *go_to_wall* utiliza el control desarrollado en el ejemplo anterior para mover el robot en línea reacta hasta llegar a una pared.
- Dentro del bucle se contemplan dos posibles situaciones:
  - Que el robot encuentre una pared de frente: esto sucede inicialmente o cuando el robot llega a las esquinas interiores del recorrido.
    - En este caso el robot realiza un giro de 90º a la izquierda utilizando el giróscopo. De esta forma el robot podrá comenzar a seguir la pared (de nuevo).
  - Que el robot no se encuentre una pared de frente: esto sucede cuando el robot se encuentre siguiendo la pared a una determianda distancia y no encuentre obstáculos enfrente.
  - OJO: En este ejemplo no se está contemplando el caso de las esquinas exteriores.
- Como la velocidad aplicada a las ruedas debe ser un número entero, se utiliza la función ***round***.
- En lugar de aplicar el valor de ***correction*** directamente, se usa la función ***limit_speed*** para mantener la velocidad dentro de unos límites aceptables.

Ajuste de Parámetros en PID

- El valor preestablecido para las diferentes constantes de proporcionalidad se determina en base a pruebas.
- Se recomienda variar estos valores para comprobar el efecto que producen en el control.

- **Kp:** Influye en la velocidad de respuesta.
- **Ki:** Asegura que el error acumulado se corrija con el tiempo.
- **Kd:** Reduce oscilaciones bruscas.

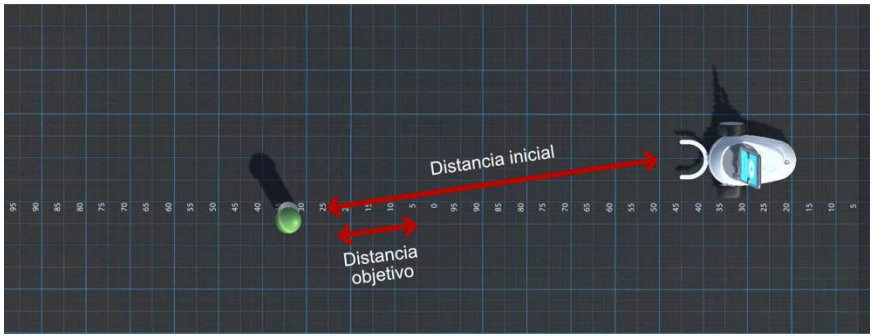
Conclusión

- El control proporcional es simple pero insuficiente en casos complejos.
- El control PID mejora la precisión y estabilidad.
- Es fundamental ajustar los parámetros Kp, Ki y Kd mediante pruebas.
- Se pueden implementar controladores intermedios: P, PI, PD.

Ejercicio 1

OBJETIVO DEL EJERCICIO

Este ejercicio consiste en desarrollar un control proporcional para que el robot se acerque a la pelota de color. Sin embargo, en lugar de hacerlo a velocidad constante, se trata de que se acerque de forma gradual, variando su velocidad en función de la distancia a la pelota de color hasta que se detiene cuando alcanza la distancia de parada establecida.



Ejercicio 1

CONFIGURACIÓN ROBOBOSIM

- Mundo
  - BALL TRACK o CYLINDER.
- Opciones
  - Activate Random Behaviour in World → Activada

Ejercicio 2

OBJETIVO DEL EJERCICIO

Este ejercicio consiste en desarrollar un control para el robot para que:

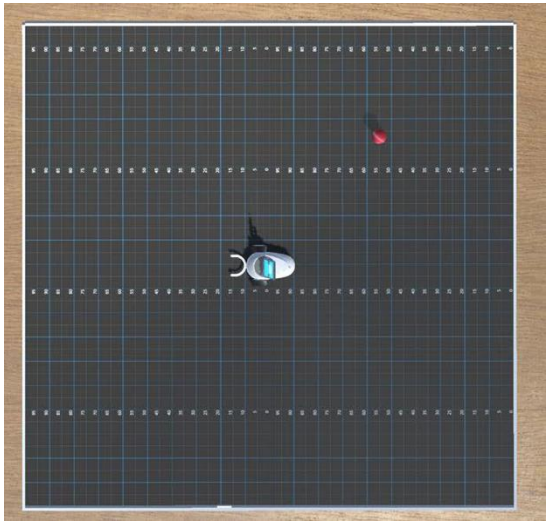
- Recoja la pelota roja con el pusher pero a medida que se acerca a recogerla, vaya girando el pusher en función de la desviación de la coordenada X obtenida al detectar un blob de color.

El control a desarrollar puede ser PD, PI o PID, elige el que consideres que ofrece mejores resultados en este caso y justifica tu elección.

Ejercicio 2

OBJETIVO DEL EJERCICIO

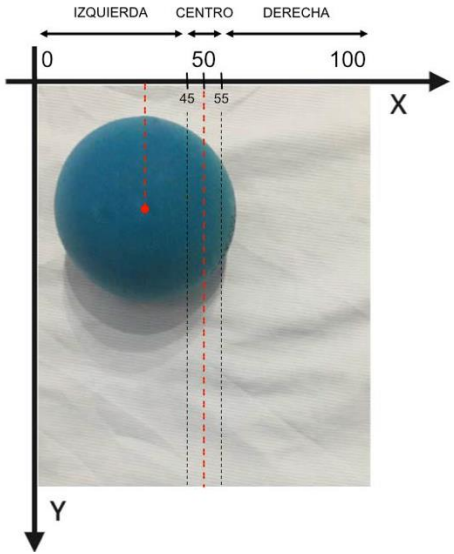
- Antes de utilizar el control para coger la pelota con el pusher, el robot ha de buscar la pelota, ya que inicialmente puede que no la vea.
- A continuación, el robot girará hacia la pelota hasta situarse delante de ella y, cuando se encuentre de frente, se acercará a la pelota de color utilizando el control desarrollado en el ejercicio anterior.
- Finalmente, se utilizará el control desarrollado en este ejercicio para recoger la pelota con el pusher.



Ejercicio 2

OBJETIVO DEL EJERCICIO

- Recuerda que la coordenada X que se obtiene cuando se detecta un blob, nos indica la posición en la que se encuentra el punto central del blob de color.
- Por tanto, la pelota se encuentra de frente, si el valor que obtenemos en X está cercano a 50 (recuerda que nunca comparamos un valor por igualdad sino que usamos un rango de valores de X, por ejemplo, entre 45 y 55 se considera que están en el centro).



Ejercicio 2

CONFIGURACIÓN ROBOBOSIM

- Mundo
  - CYLINDER o FOUR CYLINDERS.
- Opciones
  - Activate Random Behaviour in World → Activada
  - Horizontal Flip Front Camera → Desactivada

Entrega de los Ejercicios

Detalles de la entrega

- La entrega debe de realizarse en un archivo comprimido .zip que contenga los archivos con el código Python de cada ejercicio.
- Agregar comentarios en el código para facilitar la comprensión de los ejercicios.
- Agregar documento .pdf explicando cómo se ha resuelto cada ejercicio y cómo se ha ajustado cada controlador.