Práctica 3: Detección en tiempo real

Marcelo Ferreiro Sánchez
Pepe Romero Conde

28 de noviembre de 2025

1. Práctica 3.1: Seguimiento de Objeto Estático

1.1. Definición del Problema

Esta práctica que se presenta en dos partes, la primera consiste de la implementación de un sistema de detección de posición corporal mediante el cual podamos controlar al robot robobo, mientras que el objetivo de la siguiente es dotarlo de un sistema de detección de objetos gracias al cual poder adaptar la política de la Práctica 1 para que, al encontrar un objeto objetivo, se active la misma y se acerque activamente al mismo.

1.2. Estructura de la solución propuesta

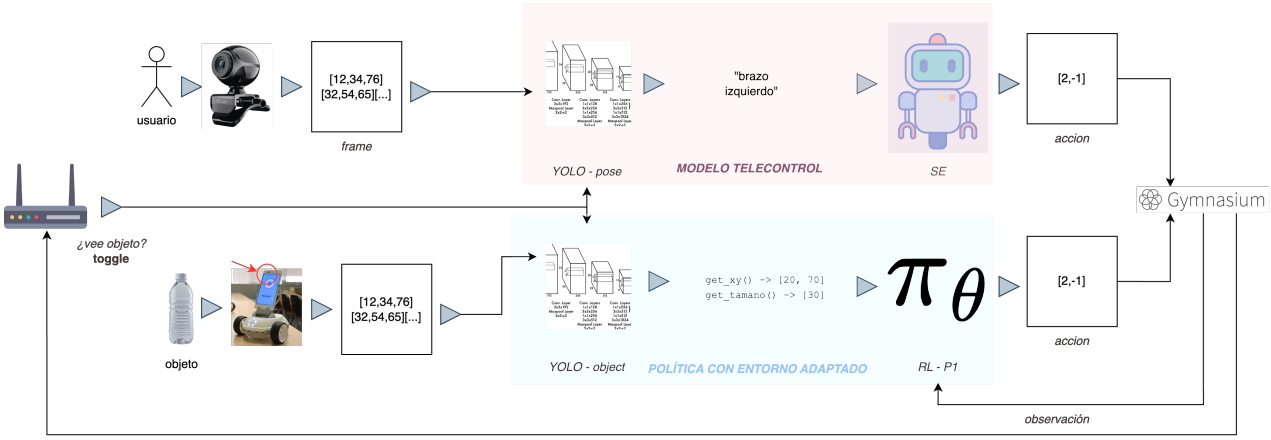

Figura 1: Diagrama de flujo del sistema. Para cada `step()`, la `observación` del mundo real (conformada por el estado interno del Robot y de lo que ve por la cámara) se usa para decidir que subsistema tomará la siguiente acción. Si no se ve al objeto, tomará el control el modelo telecontrol, si no la política adaptada de la Práctica 1.

En esta práctica, para mantener la coherencia con la P1, decidimos continuar utilizando el entorno definido en la misma (Clase entorno que hereda de la clase `gym.Env`) tanto para la ejecución de la política como para el movimiento por telecontrol.

Para gestionar el cuándo y cómo cambiar del modelo de telecontrol se creó una clase Modelo con un método `predict()`, el cual, en caso de estar viendo el objeto objetivo, llamará a la política de la P1 y en caso contrario manejará el robobo por telecontrol (YOLO - pose). La lógica para saber si el robot ve el objeto objetivo se maneja gracias a la función `esta_viendo()`, que llamará al YOLO de detección de objetos para averiguar si existe una instancia del objetivo.

1.3. Adaptaciones a entorno real

Hubieron de hacerse algunos cambios a la representación de las acciones y del espacio de observaciones del entorno, tales como, por parte de las acciones del telecontrol, transformarlas al formato en que el entorno las representa y, por parte de la detección de objetos, se tuvo que buscar formas de representar las coordenadas en pantalla del objetivo y su tamaño.

La forma de conseguir las coordenadas en pantalla, fue la del calcular el centro del rectángulo en que YOLO encuadra el objeto, y normalizar esa posición a valores en [0,100] para el valor de cada eje. De forma similar, el tamaño se calculó como el área del mismo rectángulo.

La noción de 3D para Robobo se ve supeditada por el `Enorno.observacion['tamano_blob']`, es lo unico que tiene para saber si esta cerca o lejos. Por eso hemos tenido que diseñar el sistema de una forma modular y con un archivo `config.yaml` para la fácil configuración de estos (muchos) hiperparámetros . También cuestiones relacionadas con la brusquedad del giro han necesitado ser adaptadas. Todos estos cambios se hicieron com miras a conseguir un correcto desempeño del robot en el mundo real, haciéndolo parecerse al máximo a su comportamiento en simulador.

1.4. Resultados y conclusiones

Realizando las pruebas finales en el mundo real, se pudo comprobar como se consiguió un sistema de telecontrol adecuado, si bien con algo de latencia debido al tiempo que consume el sistema YOLO en detectar los puntos clave del cuerpo del usuario y la latencia inherente a la comunicación inalámbrica.

En cuanto al comportamiento de la detección de objetos y activación de la política, el robot la activa con bastante rapidez y es capaz de acercarse al objeto. También, en caso de perder al mismo, el sistema de telecontrol vuelve a ser el vigente rápidamente, permitiendo así salvar algunas situaciones en las que el YOLO puede fallar, prediciendo que no hay botella durante la ejecución de la política.