Práctica 3: Detección en tiempo real

Marcelo Ferreiro Sánchez & Pepe Romero Conde

1. Definición del Problema

Esta práctica se presenta en dos partes. La primera consiste de la implementación de un sistema de detección de pose corporal mediante el cual podamos controlar al robot robobo, el telecontrol. El objetivo de la siguiente parte 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 (en nuestro caso una botella), se active la misma y se acerque activamente a la botella.

2. Estructura de la solución propuesta

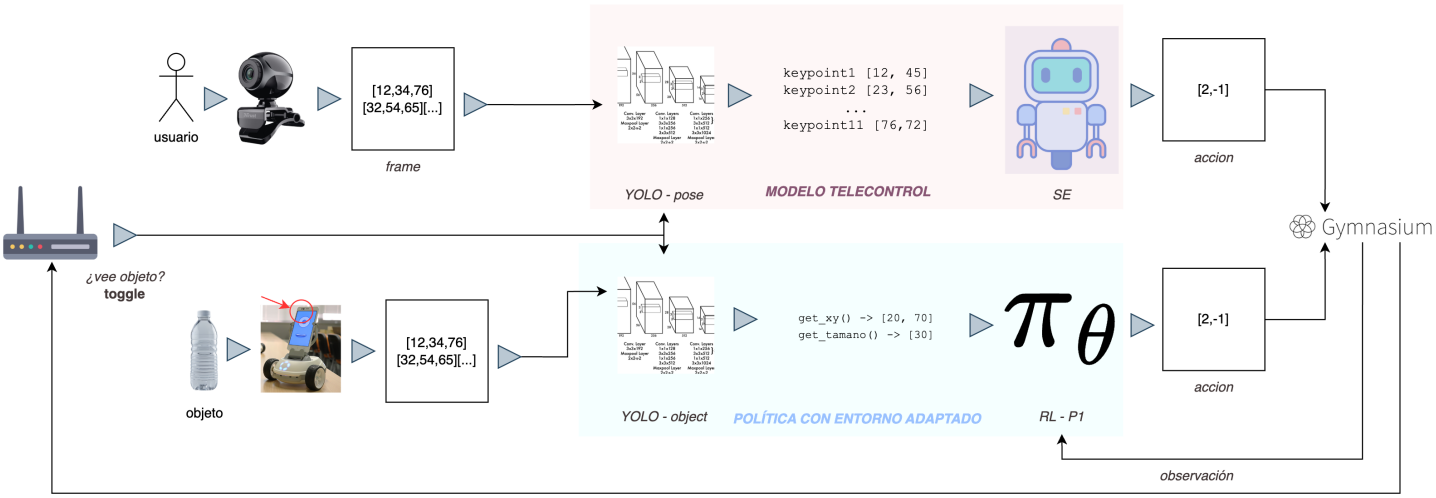

Figura 1: Diagrama de flujo del sistema. Para cada `step()`, la `observación` del mundo real (conformada por el estado interno del Robot y de lo que ve por la cámara) se usa para decidir que subsistema tomará la siguiente acción. Si no se ve al objeto, tomará el control el modelo telecontrol, si no la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

En esta práctica, para mantener la coherencia con la P1 (y en general gestionar exitosamente las dos partes de la práctica) decidimos continuar utilizando el entorno definido en la P1 (clase Entorno disponible en `Entorno.py` que hereda de la clase `gym.Env` de Gymnasium) tanto para la ejecución de la política como para el movimiento por telecontrol. Como se aprecia en la figura 1 este componente devuelve la observación al *router* para saber que subsistema hará `.predict()` en el siguiente paso (si fuese la política, esta también recibiría la observación del entorno). Además recibe las acciones de ambos sistemas, según corresponda.

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. 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.

Una vez discutidos el Entorno y el *router*, nos queda hablar de los subsistemas:

- *Modelo Telecontrol:* Cuenta con un modelo YOLO que devuelve una serie de once *keypoints*. Con estos a través de un Sistema Experto (SE) se transforma esa entrada de $\mathbb{R}^{11\times2}$ a la acción a tomar. Como vimos en las anteriores prácticas, nosotros la definimos como $(avance, giro)$, de forma que podríamos decir que:
$$SE : \mathbb{R}^{11\times2} \rightarrow \mathbb{R}^2$$
Si bien esta caracterización parece compleja, los valores del SE, fácilmente se pudieron encontrar a mano.
- *Modelo con política de la P1:* este subsistema comparte la componente de YOLO (en este caso de detección de objetos y en el anterior de pose), la cual nos vale para subministrarle a la política π_{theta} lo que necesita, en nuestro caso es responder a estas dos preguntas: ¿Dónde está la botella? ¿Cuánto de grande es la botella? (ver figura 1). Adicionalemente hemos implentado que el YOLO de la webcam, es decir, el del telecontrol solo esté activo (y en general las lecturas de la webcam) cuando no se vea a la botella. Por eso en el video subministrado podréis ver que cuando en la terminal pone `Robot dirigido por: POLITICA P1` solo se graba con la cámara de la política.

3. Adaptaciones al entorno real

Se hicieron algunos cambios a la representación al espacio de las acciones y 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 `Entorno.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.

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 la política puede fallar, prediciendo que no hay botella durante la ejecución de la política.