Práctica 2. Sistema de teleoperación por gestos

Félix Escalona Moncholí
Ciencia de la Computación e Inteligencia Artificial. Universidad de Alicante.

Mayo, 2024

1 Introducción

En esta práctica, el objetivo es desarrollar un sistema de teleoperación en ROS que permita controlar al robot mediante los gestos de la mano del operador.

Para ello, se utilizará un método de reconocimiento de gestos a partir de imagen basado en Deep Learning, utilizando como entrada una imagen de la cámara del operador. Posteriormente, estos gestos reconocidos deberán traducirse en órdenes de control sobre el robot.

Con el fin de garantizar la integridad del robot en el entorno, se deberá implementar una capa de seguridad que procese la orden de control enviada por el sistema de detección de gestos con el fin de evitar enviar al robot órdenes de movimiento que produzcan colisiones.

Utilizaremos como punto de partida la simulación de la práctica anterior, donde nuestro sistema sustituirá al sistema de navegación autónoma.

En las siguientes secciones se explicarán los módulos que se deben implementar para la realización de la práctica.

2 Configuración del entorno

Se requiere la instalación de varios paquetes adicionales sobre la configuración previa del docker de la práctica 1:

```
sudo apt-get install pip
sudo apt-get install python3.8-tk
```

Una vez instalados, se requiere incluir los siguientes paquetes mediante pip (aquí se pueden añadir paquetes adicionales que consideréis necesarios):

```
pip install mediapipe==0.10.9
```

Así mismo, deberéis añadir los nuevos ficheros python sobre el directorio /home/docker/ sobre los que tenéis que trabajar, para lo que se recomienda que sigáis las recomendaciones incluidas en la práctica 1 para dar acceso desde docker a la carpeta donde tengáis los códigos en vuestro ordenador local.

En el caso de que no podáis ejecutar los nuevos scripts de python, deberéis darles permisos de ejecución mediante el siguiente comando:

```
chmod +x script.py
```

Debéis tener presente que los cambios que hagáis sobre el docker no son persistentes, por lo que no permanecen guardados cuando salgáis. Para solventar esta circunstancia, podéis generar otra imagen docker a partir de los cambios realizados (la instalación de cualquier paquete) utilizando el comando docker commit ().

3 Visualización de la cámara del entorno

El funcionamiento de esta sección consiste en conectarse al tópico de la imagen de la cámara del entorno (/camera/color/image_raw) y abrirla en una ventana de OpenCV para visualizar el entorno del robot y poder controlarlo. Esta ventana se utilizará por parte del operador como visualizador del entorno del robot para poder enviar las órdenes de control necesarias.

En la Figura 1 se muestra un ejemplo de visualización de la cámara del entorno.

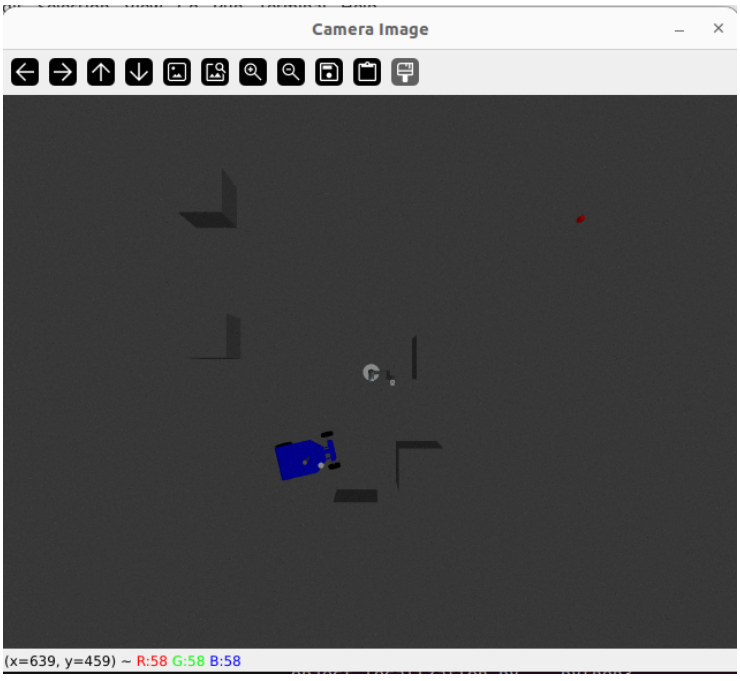


Figure 1: Ejemplo de visualización desde la cámara del entorno.

El código a modificar se encuentra en el fichero show_camera_robot.py

4 Obtención de la imagen del operador

La imagen del operador se obtendrá desde una webcam enfocándole directamente a una distancia suficiente para poder ver sus manos.

Para ello, se creará un nodo en ROS que se conecte a la webcam mediante OpenCV y publique esas imágenes en el tópico /operator/image.

Para hacer pruebas controladas, o bien en situaciones de las que no se disponga de una webcam para conectar al ordenador, se puede modificar este nodo para que capture las imágenes desde un vídeo.

El código a modificar se encuentra en el fichero capture_video.py

5 Captura e interpretación de los gestos del operador

Este módulo deberá recibir la imagen del operador, publicada en /operator/image, y aplicar una red de detección de articulaciones sobre esa imagen. El sistema propuesto para la detección de gestos es MediaPipe [1].

MediaPipe es una librería que proporciona distintas técnicas de inteligencia artificial y machine learning parar aplica sobre imagen. Concretamente, el módulo de detección de landmarks de la mano proporciona las coordenadas de 21 puntos de la mano, como se muestra en la Figura 2.

Para el funcionamiento de este sistema, se deben definir una serie de gestos basándose en las coordenadas obtenidas por Mediapipe, y que se correspondan con las siguientes órdenes de movimiento del robot:

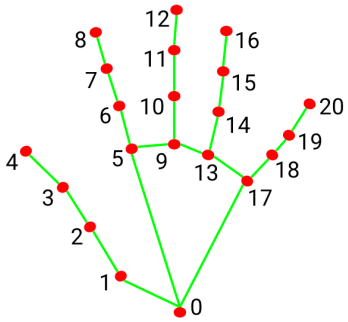
- Moverse hacia delante.
- Girar hacia la izquierda.
- Girar hacia la derecha.

Las coordenadas de la mano se deberán procesar para obtener medidas angulares entre distintas articulaciones que después se deberán utilizar para reconocer los gestos. Adicionalmente, se propone que la otra mano pueda controlar otro tipo de aspectos del movimiento, como puede ser el módulo de la velocidad.

Como salida, este nodo de ROS deberá publicar un mensaje de control sobre el tópico /blue/preorder_ackermann_cmd del robot dónde se incluya la orden de movimiento que deberá aplicar el robot.

Por motivos de comprobación del funcionamiento, se deberá abrir una ventana de OpenCV donde se muestre la imagen del operador con los gestos y las articulaciones detectadas por el sistema, como se muestra en la Figura 3.

El código a modificar se encuentra en el fichero pose_estimation.py



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 2: Modelo de landmarks de la mano.

Figure 3: Ejemplo de detección de landmarks de la mano.

6 Sistema de seguridad del robot

Finalmente, se debe desarrollar un nodo en ROS que, una vez recibida la orden de control en el tópico /blue/preorder_ackermann_cmd, consulte la información de los obstáculos detectados en la práctica 1 (/obstacles) para evitar colisiones con el entorno, y enviar la orden definitiva de control sobre /blue/ackermann_cmd.

En caso de detectarse un obstáculo en la trayectoria que va a realizar el robot, el robot debe enviar una orden de movimiento para que se detenga con el fin de prevenir la colisión. Si por el contrario el espacio está libre, se deberá transmitir la orden de control original.

El código a modificar se encuentra en el fichero obstacle_avoidance.py

Con el fin de comprobar el funcionamiento correcto de este sistema, se recomienda hacer pruebas añadiendo diferentes objetos sobre la simulación de gazebo y moviendo el robot por el entorno intentando forzar las colisiones con ellos mediante las órdenes de control generadas por el sistema de reconocimiento de gestos.

7 Consideraciones

Se debe realizar la correspondiente experimentación para valorar el funcionamiento del sistema, tanto del reconocimiento de gestos como del sistema de seguridad ante colisiones. Toda esta experimentación debe recogerse en la memoria para que se pueda ver la evolución que ha experimentado el trabajo.

Para facilitar el seguimiento de la práctica se recomienda la utilización de un sistema de control de versiones como Git. Bitbucket es una buena opción gratuita que permite la creación de repositorios privados.

Finalmente, se recuerda que la memoria es una parte muy importante de la práctica y que todas las decisiones tomadas sobre la implementación deberán recogerse y justificarse adecuadamente. En caso de entregarse un proyecto cuya implementación no esté debidamente justificada, se considerará como si no se hubiese entregado.

8 Documentación a entregar

La documentación de la práctica es una parte muy importante en la puntuación final. El código debe estar debidamente comentado, indicando qué se hace en cada punto. La entrega final deberá constar de:

- Documentación en PDF. Se aconseja incluir las siguientes secciones: introducción, estado del arte, desarrollo, experimentación y conclusiones.
- Código Python ejecutable sin errores que cumpla completamente con los requisitos de la práctica.
- Vídeo demostrativo del funcionamiento de la práctica.

Otras mejoras adicionales que se incorporen en la práctica podrán ser valoradas positivamente según su nivel de complejidad y adecuación al problema.

9 Fecha de entrega

La fecha de entrega máxima para la práctica será el viernes 24 de mayo a las 23:59 a través de Moodle.

References

[1] Google. Hand landmarks detection guide. `https://developers.google.com/mediapipe/solutions/vision/hand_landmarker`, 2023. Accessed: 01-03-2024.