Raspberry Pi 5 & RealSense D435 ROS Noetic in Docker – Full Workflow

Legend

- **Laptop** – local terminal on your laptop
- **SSH-Pi** – SSH session connected to the Raspberry Pi
- **Docker-Pi** – shell *inside* the Docker container `ros_project`
- **Pi-local** – physical console attached to the Pi (optional)

1. Prerequisites

- Raspberry Pi 5 running Ubuntu **24.04 LTS** (64-bit)
- Intel RealSense D435 connected via USB 3
- ROS Noetic inside a Docker container named `ros_project`
- Pi and laptop on the same LAN

2. Boot & SSH login

Laptop → SSH-Pi

1: Laptop → SSH-Pi

```
ssh ubuntu@<PI_IP_ADDRESS>
# password: ubuntu (or your SSH key)
```

SSH-Pi → Docker-Pi

2: SSH-Pi → Docker-Pi

```
# start container without attaching to its primary STDIN
docker start ros_project

# open a separate shell inside it
docker exec -it ros_project bash
```

3. Configure ROS in the container

Docker-Pi

3: Source overlays

```
# load ROS overlays
source /opt/ros/noetic/setup.bash
source ~/catkin_ws/devel/setup.bash
```

Add the two lines above to the container's ~/`.bashrc`.

4. Install `tmux` and create a persistent session

Docker-Pi

4: Install & start tmux

```
# fix Intel repo and install tmux
curl -s https://librealsense.intel.com/Debian/librealsense.pgp | \
  gpg --dearmor -o /usr/share/keyrings/librealsense-archive-keyring.gpg
echo "deb␣[signed-by=/usr/share/keyrings/librealsense-archive-keyring.gpg]␣\
␣␣https://librealsense.intel.com/Debian␣$(lsb_release␣-cs)␣main" | tee \
  /etc/apt/sources.list.d/librealsense.list
apt update
apt install -y tmux

# launch a tmux session named 'cam'
tmux new -s cam
```

Inside the new *cam* session:

5: Run RealSense node

```
roslaunch realsense2_camera rs_camera.launch
```

Detach without stopping the camera: press `Ctrl-b` then `d`.

5. Record a `rosbag`

Docker-Pi – new tmux window

6: Record bag

```
mkdir -p ~/bags && cd ~/bags

# record video, camera_info and TF
rosbag record /camera/color/image_raw \
             /camera/color/camera_info \
             /tf /tf_static \
             -O realsense_$(date +%Y%m%d_%H%M%S).bag
# press Ctrl-C to stop
```

6. Copy the `.bag` out of the container

SSH-Pi

7: Extract bag

```
CID=$(docker ps -qf name=ros_project)
docker cp "${CID}":/root/bags ./bags_from_container
```

Bag files are now in `/home/ubuntu/bags_from_container/` on the Pi host.

7. Transfer the bag to the laptop

Laptop

8: Download bag

```
scp ubuntu@<PI_IP_ADDRESS>:/home/ubuntu/bags_from_container/*.bag ./
```

8. Play the `rosbag` on the laptop

Laptop

9: Replay locally

```
# 1) start the ROS master
roscore # keep this window open

# 2) in a second terminal
rosbag play realsense_20250528_151718.bag
rqt_image_view /camera/color/image_raw
```

If the topic is compressed, choose `/camera/color/image_raw/compressed` in *rqt_image_view*.

9. Best practices & troubleshooting

- Never run `exit` in the shell started with `docker start -ai`; instead use `docker exec` + `tmux`.
- Ensure stable power for the Pi ($\geq$ 5 V / 5 A) to avoid undervoltage.
- Check free space before long recordings with `df -h`.
- For very long sessions use `rosbag record . . . --split --duration=120` (new file every 2 min).

Following these steps you can reboot the Pi, start everything from scratch, record a RealSense stream in Docker, and replay it on your laptop – all without interrupting the camera feed.

10. YOLOv8 integration and visualisation

Inside Docker (YOLO node)

10: Docker-Pi → YOLO tmux window

```
# Create a new tmux window inside the 'cam' session:
Ctrl-b c

# Source ROS overlays
source /opt/ros/noetic/setup.bash
source ~/catkin_ws/devel/setup.bash

# Launch YOLO node
roslaunch yolov8_ros det_cam.launch
```

The YOLO node will publish:

- `/yolo/image_annotated`
- `/yolo/detections`
- `/yolo/fps`

YOLO Python node (inside Docker)

`yolov8_ros/scripts/yolo_node.py`:

- Loads `yolov8n.pt` model
- Subscribes to `/camera/color/image_raw`
- Publishes annotated image + detections + FPS

Make sure to make the script executable:

11:

```
chmod +x ~/catkin_ws/src/yolov8_ros/scripts/yolo_node.py
```

Record YOLO + camera ROSBAG

12: Docker-Pi – new tmux window

```
mkdir -p ~/bags && cd ~/bags

rosbag record /camera/color/image_raw \
            /camera/color/camera_info \
            /yolo/image_annotated \
            /yolo/fps /yolo/detections \
            /tf /tf_static \
            -O yolov8n_$(date +%Y%m%d_%H%M%S).bag
```

Replay and analyse on laptop

13: Laptop – Visualise with rqt tools

```
roscore # terminal 1
rosbag play yolov8n_28_05_v2.bag # terminal 2

# Visualise annotated image
rqt_image_view /yolo/image_annotated

# Check FPS and detections
rostopic echo /yolo/fps
rostopic echo /yolo/detections
```

YOLO detections may run at lower FPS due to:

- Resource limitations (Raspberry Pi CPU)
- Large image size or high confidence threshold

For improved FPS, consider:

- Lowering `img_size` in the launch file
- Using the `yolov8n` (nano) model as done here