Step-by-Step Guide: YOLOv8-Pose Integration in ROS2 (Empty Simulation)

This section documents the complete workflow followed to implement a real-time human pose estimation pipeline using YOLOv8-Pose integrated within a ROS2 and Gazebo simulation environment.

---

1. Creating the Rosject (The Construct)

1. Log in to **The Construct AI** platform.
2. Navigate to **My Rosjects → Create Rosject**.
3. Select:
   - **ROS2**
   - **ROS2 Humble**
4. Name the Rosject:

   ```
   yolo_pose_empty_sim
   ```
5. Choose **Empty Simulation**.
6. Create the Rosject and wait for the environment to load.

---

2. Starting the Simulation Runtime

1. Click **Play ▶** to start the Rosject simulation.
2. Open a **Terminal** inside the environment.
3. Verify that ROS2 is running:

   ```
   ros2 topic list
   ```

At this stage, only basic system topics (e.g. /rosout) are expected.

---

3. Preparing the ROS2 Workspace

1. Navigate to the ROS2 workspace:

```
cd ~/ros2_ws/src
```

2. Add the robot packages so the structure is:

```
ros2_ws/src/
├── robot_description
├── robot_bringup
├── robot_control
```

3. Build the workspace:

```
cd ~/ros2_ws
colcon build
source install/setup.bash
```

4. Verify the packages are detected:

```
ros2 pkg list | grep robot
```

---

4. Launching Gazebo and Spawning the Robot

4.1 Launch Gazebo (Empty World)

```
ros2 launch gazebo_ros gazebo.launch.py
```

Wait until Gazebo opens fully.

4.2 Spawn the Robot

Open a new terminal and run:

```
cd ~/ros2_ws
source install/setup.bash
ros2 launch robot_bringup spawn_robot.launch.py
```

At this point:

- The two-wheeled robot appears in Gazebo
- Wheel joints and LiDAR are active

---

5. Adding an RGB Camera to the Robot (Critical Step)

5.1 Edit the Robot URDF

```
vi ~/ros2_ws/src/robot_description/urdf/two_wheeler_robot.urdf
```

5.2 Add Camera Link, Joint, and Gazebo Plugin

Insert the following **before** the closing `</robot>` tag:

```
<!-- Camera -->
<link name="camera_link">
    <visual>
        <geometry>
            <box size="0.05 0.05 0.05"/>
        </geometry>
    </visual>
</link>

<joint name="camera_joint" type="fixed">
    <parent link="chassis"/>
    <child link="camera_link"/>
    <origin xyz="0.15 0 0.25" rpy="0 0 0"/>
</joint>

<gazebo reference="camera_link">
    <sensor type="camera" name="camera">
        <always_on>true</always_on>
        <update_rate>30</update_rate>
        <camera>
            <horizontal_fov>1.396</horizontal_fov>
            <image>
                <width>640</width>
                <height>480</height>
                <format>R8G8B8</format>
            </image>
            <clip>
                <near>0.1</near>
                <far>100</far>
            </clip>
        </camera>
        <plugin name="gazebo_ros_camera"
    filename="libgazebo_ros_camera.so">
            <image_topic_name>image_raw</image_topic_name>
            <frame_name>camera_link</frame_name>
        </plugin>
    </sensor>
</gazebo>
```

Save and exit:

```
Esc → :wq → Enter
```

---

6. Rebuild and Restart Simulation

1. Rebuild the workspace:

   ```
   cd ~/ros2_ws
   colcon build
   source install/setup.bash
   ```

2. Restart Gazebo:

   ```
   ros2 launch gazebo_ros gazebo.launch.py
   ```

3. Spawn the robot again:

   ```
   ros2 launch robot_bringup spawn_robot.launch.py
   ```

---

7. Verifying the Camera Topic

```
ros2 topic list
```

Expected output includes:

```
/camera/image_raw
/camera/camera_info
```

This confirms the camera sensor is correctly integrated.

---

8. Visualising the Camera in RViz

1. Launch RViz:

```
rviz2
```

2. Click **Add** → **Image**.
3. Set the topic to:

```
/camera/image_raw
```

---

9. Installing YOLOv8 and Dependencies

```
pip install ultralytics opencv-python
pip uninstall numpy -y
pip install "numpy<2"
```

This ensures compatibility with ROS2 `cv_bridge`.

---

10. Creating the YOLOv8 ROS2 Node

10.1 Create the Package

```
cd ~/ros2_ws/src
ros2 pkg create yolo_pose --build-type ament_python
```

10.2 Create the Node (with ROS publishing)

```
cd yolo_pose/yolo_pose
vi yolo_pose_node.py
```

Paste:

```
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from geometry_msgs.msg import PoseArray, Pose
from cv_bridge import CvBridge
from ultralytics import YOLO
import cv2
```

```
class YoloPoseNode(Node):
    def __init__(self):
        super().__init__('yolo_pose_node')
        self.bridge = CvBridge()
        self.model = YOLO("yolov8n-pose.pt")

        self.subscription = self.create_subscription( Image,
    '/camera/image_raw', self.image_callback, 10)
        self.pose_pub = self.create_publisher(PoseArray,
    '/pose_estimation', 10)

    def image_callback(self, msg):
        frame = self.bridge.imgmsg_to_cv2(msg, 'bgr8')
        results = self.model(frame)

        if results[0].keypoints is None:
            return
        pose_array = PoseArray()
        pose_array.header.stamp = self.get_clock().now().to_msg()
        pose_array.header.frame_id = "camera_link"

        keypoints = results[0].keypoints.xy.cpu().numpy()
        for person in keypoints:
            for kp in person:
                pose = Pose()
                pose.position.x = float(kp[0])
                pose.position.y = float(kp[1])
                pose.position.z = 0.0
                pose_array.poses.append(pose)

        self.pose_pub.publish(pose_array)

        annotated = results[0].plot()
        cv2.imshow("YOLOv8 Pose Detection", annotated)
```

```
                cv2.waitKey(1)
def main():
        rclpy.init()
        node = YoloPoseNode()
        rclpy.spin(node)
        rclpy.shutdown()

if __name__ == '__main__':
        main()
```

10.3 Register the Node

Edit `setup.py`:

```
'console_scripts' : [ 'yolo_pose_node = yolo_pose.yolo_pose_node:main', ],
```

---

11. Build and Run YOLOv8-Pose

```
cd ~/ros2_ws
colcon build
source install/setup.bash
ros2 run yolo_pose yolo_pose_node
```

YOLOv8-Pose now runs in real time and publishes pose keypoints.

---

12. Adding a Human Model to the Simulation

1. In Gazebo, open the **Insert** panel.
2. Search for **Actor** or **Human**.
3. Insert the model in front of the robot.
4. Ensure the human is visible in /camera/image_raw.

---

13. Visualising Keypoints in RViz

1. In RViz, click **Add → PoseArray**.
2. Set the topic to:

```
/pose_estimation
```

Keypoints are now visible as ROS-native data.