PetWave: Interactive Robotic Arm for Human and Dog Interaction

Zhao Zhenhai,Yangxue,Wang Bohan, Zhang Zihao, Wei Yutong, Hong Yuying and Li Yunzhe

***Abstract*—This project aims to develop a virtual environment enabling effective interaction between a robotic arm and humans as well as dogs. By integrating advanced machine learning algorithms and the ROS framework, the project achieves accurate recognition of humans and dogs within the environment, facilitating simple interactive actions such as waving and petting. Utilizing the YOLO algorithm for real-time object detection and classification, experiments were conducted within the ROS Reality virtual reality framework. The paper synthesizes insights from three relevant studies, exploring the applications and advantages of the ROS framework, object detection algorithms, and long-range recognition models. Preliminary experimental results demonstrate successful recognition of humans and dogs, alongside the capability for simple interactions. This innovative combination of a robotic arm with advanced algorithms holds significant promise for improving the experience of pets in clinical settings, making hospital visits less stressful for animals and more manageable for veterinarians and pet owners alike. This project offers new perspectives and methods for efficient human-robot interaction, highlighting its potential impact on veterinary care.**

I. Introduction

Pets, particularly dogs, often experience significant anxiety and stress during visits to veterinary hospitals. This anxiety can complicate treatment procedures, making it difficult for veterinarians to perform necessary examinations and treatments. Furthermore, pet owners may feel distressed seeing their pets in discomfort, leading to a less satisfactory overall experience. Various technological advancements have been made to improve animal welfare in veterinary settings. However, these approaches often lack real-time interaction and adaptability to individual animals' needs.

The objective of this project is to create a simulated environment enabling interaction between a robotic arm, humans, and dogs. For example, it can wave in response to a person's presence and pat a dog's head upon detection. The core objective is to create a calming experience for pets in hospitals, utilizing advanced machine learning algorithms and the Robot Operating System (ROS) framework. The system is designed to recognize and interact with humans and dogs, performing actions such as waving and petting to provide comfort.By reducing anxiety, our project aims to improve the overall veterinary care experience, leading to better treatment outcomes and higher satisfaction for both pets and their owners.

The methodology encompasses the utilization of machine learning algorithms for object detection and classification, integration of a robotic arm model within the ROS framework, and deployment of path planning algorithms for navigation. This paper delineates the advancements and initial discoveries achieved until the milestone checkpoint. During subsequent phases of optimization and iteration, its recognition capabilities and interactive gestures are intended to be enhanced. These enhancements may include recognizing cats and engaging in playful interactions to entertain them, as figure.1 , as well as assessing the proximity between the robotic arm and individuals, waving when they are distant and offering a handshake when they are nearby.

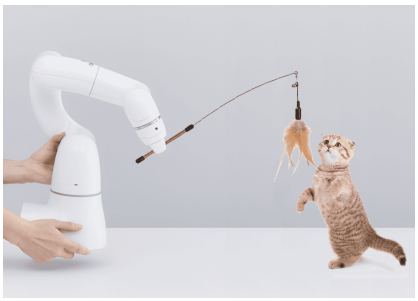

Fig. 1. Tease Cat Robot

II. Problem Statement

The objective of this project is to develop a virtual environment facilitating interactions between a robotic arm, humans, and dogs. Specifically, the aim is to enable accurate recognition of humans and dogs by the robotic arm within the environment, allowing for simple interactions such as waving to humans and patting dogs.

The project plan entails the study of YOLO algorithm principles and utilization of the COCO dataset for training a deep learning model capable of real-time detection and recognition, then procuring a controllable robotic arm within the ROS framework and integrate the image recognition algorithm with the robotic arm for simulation experiments conducted in the Gazebo simulator.

The anticipated outcome is for the robotic arm to accurately detect and differentiate between humans and dogs within the Gazebo simulator and interact with them based on predefined actions.Evaluation of the project will be based on the accuracy of human and dog detection, the precision of robotic arm interaction gestures, and the response time of the robotic arm in recognition and interaction.

III. Literature Review

[1] David Whitney; Eric Rosen; Daniel Ullman; Elizabeth Phillips; Stefanie Tellex ROS Reality: A Virtual Reality

Framework Using Consumer-Grade Hardware for ROS-Enabled Robots. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 1-9, doi: 10.1109/IROS.2018.8593513.*

This paper aims to introduce ROS Reality, which is an interface enabling remote operation of ROS-enabled robots using consumer-grade virtual reality (VR) and augmented reality (AR) hardware. The paper describes the system architecture and applications of ROS Reality, detailing how the system facilitates VR remote operation and integrates consumer-grade VR and AR hardware with the ROS system, allowing users to view and control robots over the internet.

This paper provides some insights for the interaction between the robotic arm and users in the project. Referring to the system architecture and methods in this paper, one can understand how to integrate with the ROS system and utilize consumer-grade VR and AR hardware for remote operation and user interface.

[2] Shaoqing Ren; Kaiming He; Ross Girshick; Jian Sun Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.*

This paper introduces a novel object detection method called the Region Proposal Network (RPN), which shares full-image convolutional features within the object detection network, thereby achieving nearly cost-free region proposals. This paper can provide an effective way to reduce the computational cost of region proposals and improve the speed and accuracy of object detection. Considering the application of this method to the project can enhance the robotic arm's recognition and interaction capabilities with humans and dogs.

[3] Felix Gunawan; Chih-Lyang Hwang; Zih-En Cheng ROI-YOLOv8-Based Far-Distance Face-Recognition. *2023 International Conference on Advanced Robotics and Intelligent Systems (ARIS), Taipei, Taiwan, 2023, pp. 1-6, doi: 10.1109/ARIS59192.2023.10268512.*

This paper introduces a model for far-distance face recognition utilizing ROI-YOLOv8, trained on custom datasets with various augmentation levels. The model adopts a two-stage recognition approach, initially employing a pre-trained YOLOv8 for human detection followed by ROI-YOLOv8 for face recognition. Evaluation metrics such as mAP50 are utilized to assess performance, with the trained model effectively recognizing faces at distances of 30 to 35 meters with high confidence. The methodology and insights provided by this study can inform the design and training of a recognition system for the robotic arm, particularly in scenarios necessitating far-distance detection and recognition of humans and dogs.

[4] Sumit Tariyal; Rahul Chauhan; Yogesh Bijalwan; Ruchira Rawat; Rupesh Gupta A comparitive study of MTCNN, Viola-Jones, SSD and YOLO face detection algorithms. *024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bangalore, India, 2024, pp. 1-7, doi: 10.1109/IITCEE59897.2024.10467445.*

The paper presents a comparative study of four prominent face detection algorithms: Viola-Jones, MTCNN, SSD, and YOLO. It emphasizes the balance between speed and accuracy, which are critical for applications in computer vision such as security and facial recognition. Viola-Jones, as a pioneering algorithm, offers high accuracy but may lack speed compared to newer methods. MTCNN excels in accuracy, particularly in facial landmark detection, but might struggle with real-time processing demands. SSD stands out for its balance of speed and accuracy, making it suitable for real-time applications. YOLO is celebrated for its real-time capabilities, offering a competitive combination of speed and accuracy, which is ideal for resource-constrained and dynamic environments.

IV. Technical Approach

A. Robot Operation System

ROS, short for Robot Operating System, is a flexible framework and toolkit for robot development. It provides a series of libraries and tools to assist developers in creating, managing, and deploying robot applications.

ROS provides a variety of usable robotic arm model packages, such as URDF models, MoveIt software package, etc. The Gazebo simulation platform can visualize the functions of robotic arms, facilitating qualitative evaluation of the results.

1) URDF: URDF (Unified Robot Description Format) is an XML syntax framework used to describe the structure and attributes of robots. In ROS, URDF plays a crucial role, primarily used to describe the geometric shapes, link structures, joint types, and connection relationships of robots.

The process of leveraging URDF in ROS involves creating URDF files, utilizing XML syntax to describe the geometric structure, linkages, joint connections, and physical properties of robots. Subsequently, loading URDF models into the ROS environment can be accomplished using the roslaunch command or Python scripts. Initiating a simulation environment in ROS, such as the Gazebo simulation platform, loading URDF models, and conducting simulation and analysis to assess the robot's performance are crucial steps. Moreover, utilizing loaded

2) MoveIt: MoveIt is an integrated package designed for robot motion planning and control within the Robot Operating System (ROS). It offers functionalities including motion planning, manipulation control, 3D perception, kinematics, and navigation algorithms. The typical process of utilizing MoveIt within ROS involves assembling the robot's URDF model, configuring parameters using the MoveIt! Setup Assistant tool, integrating controller plugins, and controlling robot motion through MoveIt's API or GUI, enabling developers to efficiently develop and deploy robust robot motion control applications.

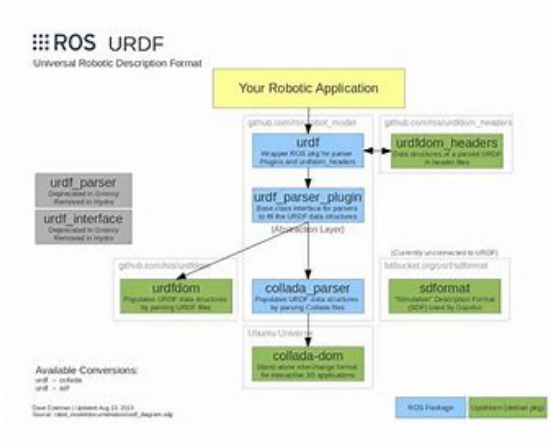

Fig. 2. URDF

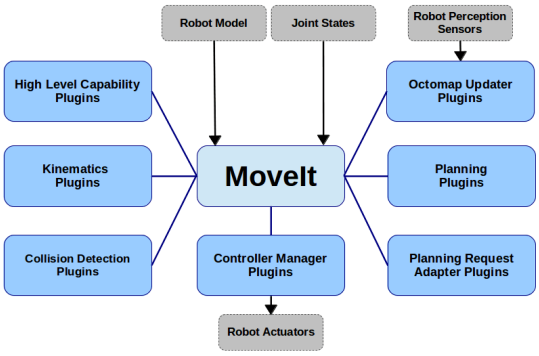

Fig. 3. MoveIt

3) Gazebo: Gazebo, an open-source robot simulation software widely utilized in robotics, offers a comprehensive suite of functionalities. These include facilitating the creation of intricate robot models using basic geometric shapes or imported CAD/Blender drawings, constructing diverse simulation scenes with objects from libraries or custom-built structures, simulating a wide array of sensors such as cameras and LiDARs, enabling the incorporation of real-world physical properties like gravity and friction into models, and providing a realistic physics simulation engine. The workflow typically involves users creating or importing models, constructing scenes, adding sensors, defining physical properties, conducting simulations, and analyzing results. In essence, Gazebo serves as a versatile platform for designing, testing, and validating robot systems through realistic simulation environments and rigorous testing procedures.

B. Image Recognition Algorithm–YOLO

YOLO(You Only Look Once)is a popular object detection algorithm known for its speed and accuracy.The core principle of YOLO (You Only Look Once) is to treat the object detection task as a single end-to-end regression problem. It employs a single neural network to predict the bounding boxes and class probabilities of objects within an image. Compared to traditional object detection methods, YOLO can directly output the positions and class probabilities of all objects in a single run, making it faster.

1) Approach: Whole Image Prediction: YOLO feeds the entire image into a neural network and outputs bounding boxes and class probabilities in a single forward pass.

Grid Segmentation: The image is divided into fixed-size grids, with each grid responsible for predicting the presence of objects, along with their bounding boxes and classes.

Feature Extraction: Deep Convolutional Neural Networks (CNNs) are applied to each grid cell to extract features.

Regression Output: The CNN outputs multiple bounding boxes and class probabilities for each grid cell.

Non-Maximum Suppression (NMS): Overlapping bounding boxes are merged using the NMS algorithm to reduce duplicate detections.

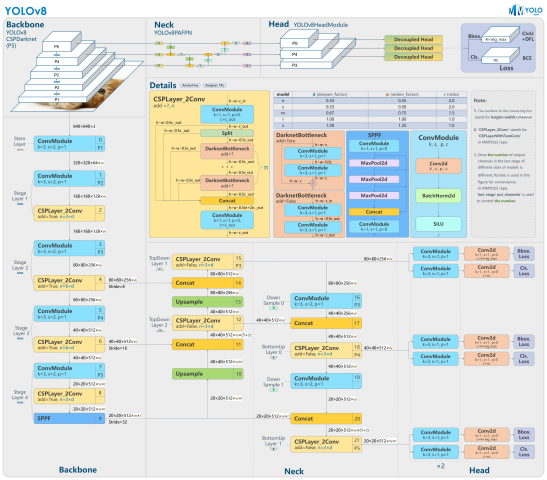

Fig. 4. YoloV8 Approach

2) Workflow: Input Image: Feed the image to be detected into the YOLO model.

Feature Extraction: Extract image features using CNN.

Prediction: For each grid cell, predict bounding boxes and class probabilities using CNN.

NMS: Apply non-maximum suppression to filter out overlapping bounding boxes.

Output: Output the final bounding boxes and class probabilities.

3) Significance: Object Detection: Primarily used for object detection tasks in images, capable of detecting multiple objects and providing their positions and classes.

Real-time Performance: Due to its efficient design, YOLO performs exceptionally well in real-time applications such as autonomous driving, surveillance, etc.

Single-stage Detection: Compared to two-stage detection methods like Faster R-CNN, YOLO only requires a single forward pass to complete the detection task, making it more efficient.

Multi-scale Detection: YOLO can handle objects of different sizes, making it suitable for various scenarios.

Fig. 5. Yolo Significance

C. Path Planning——Linear Interpolation Method

Linear interpolation is a fundamental technique used in various fields such as mathematics, computer graphics, robotics, and signal processing. It is employed to estimate values between two known data points by assuming a linear relationship between them.

1) Mathematical Formulation: Consider two known points $P_0 = (x_0, y_0)$ and $P_1 = (x_1, y_1)$, with $(x_0 \neq x_1)$. Linear interpolation calculates the value y at a point x lying between x_0 and x_1 using the equation:

$$y = y_0 + \frac{x - x_0}{x_1 - x_0} \times (y_1 - y_0) \tag{1}$$

This formula represents the linear relationship between x and y within the interval $[x_0, x_1]$.

2) Key Concepts: Interpolation Parameter α: The interpolation parameter α represents the relative position of the interpolated point between the two known points. It ranges from 0 to 1, where $\alpha = 0$ corresponds to the first point P_0 and $\alpha = 1$ corresponds to the second point P_1.

Linear Relationship: Linear interpolation assumes that the relationship between the independent variable (e.g., time, distance) and the dependent variable (e.g., temperature, position) is linear over the interval between two known points.

V. Preliminary Results

A. Mechanical Arm

To ensure the robot has the ability to wave to human and pat dogs, an open-sourced model is used as shown in the fig.6.

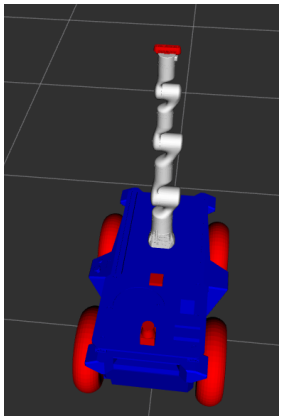

Fig. 6. The Robot

The model includes a car base as well as a robotic arm with multi-degree of freedom.

With the use of ROS, advantage is taken of Moveit to control the robot arm and plan the moving path, as shown as fig.7

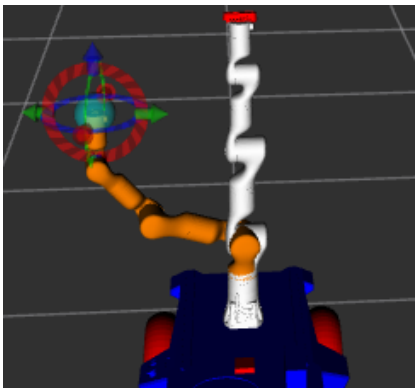

Fig. 7. The Control of Robot

The path of the robot arm can be effectively planned through this method, thus ensuring the specified actions are correctly performed by the robotic arm in gazebo.

However, given the lack of aesthetic appeal of the model and its inability to meet the requirements of the actions designed for our project, we opted to utilize the SolidWorks tool for constructing the mechanical arm model, as depicted in fig.8

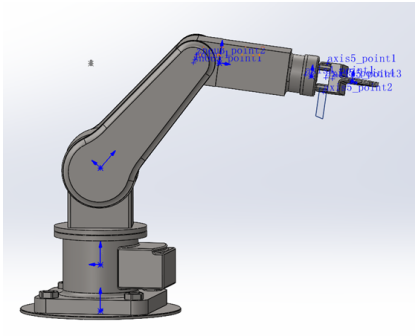

Fig. 8. Mechanical Arm in Solidworks

This is a 6-degree-of-freedom robotic arm, capable of achieving corresponding movements through the rotation, pitch, yaw, elbow joint, and wrist joint of its five components.

Subsequently, configuring the appearance of the robot body, object properties, joint types, and other parameters, the SolidWorks model is exported as a URDF model, facilitating its seamless integration into ROS.

B. Real-Time Detection by YOLO

Based on evaluations of real-time performance, simplicity, resource efficiency, and applicability, the decision was made to use the YOLO algorithm and forego the Faster R-CNN and SSD algorithms in the project. Initially, a model capable of recognizing images was trained,as shown as fig.9.

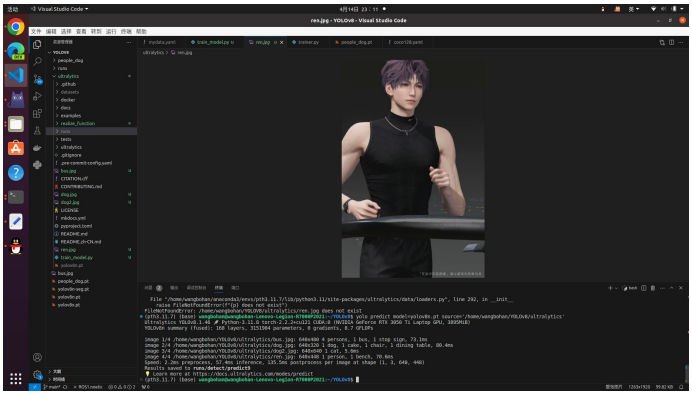

Fig. 9. Recognizing Images

Afterwards, a camera was integrated into the YOLO algorithm to perform real-time detection of classes captured by the camera,as shown as fig.10.

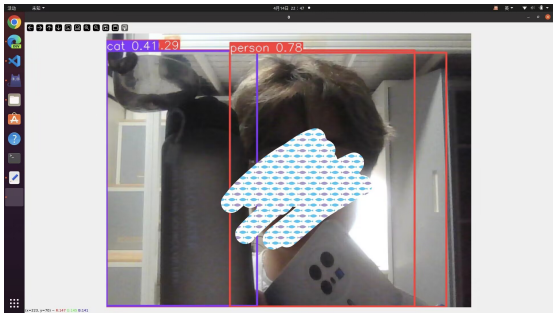

Fig. 10. Real-Time Detection

Currently, a new dataset containing images of only humans and dogs has been re-collected, and a model with higher accuracy, capable of detecting only these two classes, has been traine, as shown as fig.11.

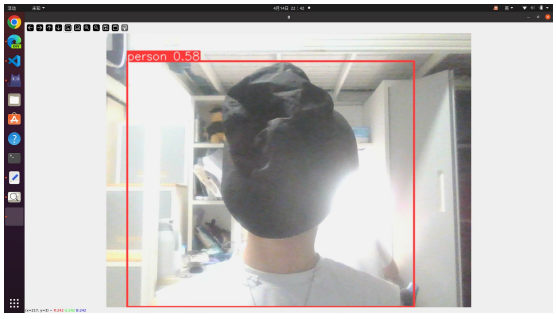

Fig. 11. Higher Accuracy

Subsequently, datasets tailored to scenarios involving humans, dogs, and cats will be gathered to train respective models, with the goal of further enhancing their accuracy. Moreover, within ROS, actions like nodding, bowing, and dancing have been devised to correspond with the recognition of various objects.

C. Combining YOLO and ROS

After discovering a software package online enabling YOLO integration into ROS, we autonomously developed a robotic arm model. We compiled node controllers to facilitate the robotic arm's operation based on commands, received and processed YOLO recognition information commands from the robotic arm, and customized the original YOLO package code. Ultimately, we attained visual recognition control for our robotic arm.

Regarding the programming files of the provided yolov8_ros package, we implemented a check for the interval between two image timestamps. If the elapsed time is less than the specified duration, the subsequent check is deferred. Conversely, if the elapsed time reaches the specified duration, the next image recognition check is initiated.

```
# Set inference interval
self.last_inference_time = time()
self.inference_interval = rospy.get_param('~inference_interval', 3)  # Default 0.1 seconds
```

Fig. 12. Time Interval

The original inspection frequency was too high, which might cause the robotic arm to start the next loop before completing a specified action, resulting in unsmooth movements (the interval for the above check time is set to 3 seconds, which ensures the robotic arm completes a clear motion trajectory).

Subsequently, we can also train and import new models to achieve more accurate recognition for specific scenarios.

```
<param name="weight_path"       value="$(find yolov8_ros)/weights/people_dog_2.pt"/>
<param name="image_topic"       value="/usb_cam/image_raw" />
```

Fig. 13. Importing Paths

Simply select and replace the new configuration file in the launch file.

Control the movement of the robotic arm by publishing joint state messages to each node.

```
while not rospy.is_shutdown():
    joint_msg = JointState()
    joint_msg.header.stamp = rospy.Time.now()
    joint_msg.name = ["joint_z", "joint_o", "joint_t", "joint_th", "joint_f", "joint_fi"]
    joint_msg.position = [0] * 6
```

Fig. 14. Joint state

Specifically, this is demonstrated by selecting six keyframes for each action and employing interpolation functions to achieve smooth and seamless operation of the robotic arm.

The interpolation function is as follows:

During operation, a significant challenge we faced was the robotic arm's stuttering movement. After thorough exploration and debugging, we discovered that increasing the loop frequency could mitigate this issue. With a higher loop frequency, the robotic arm can receive new positional commands

```
# 舞蹈动作的关键帧（6个自由度）
dance_positions = [
    [0, 0, 0, 0, 0, 0],        # 初始位置
    [0, 0.5, -0.5, -0.5, 0.5, -0.5], # 舞蹈动作1
    [0, -0.5, 0.5, 0.5, -0.5, 0.5], # 舞蹈动作2
    [0, -0.5, 0.5, -0.5, 0.5, -0.5], # 舞蹈动作3
    [0, 0.5, -0.5, 0.5, -0.5, 0.5], # 舞蹈动作4
    [0, 0, 0, 0, 0, 0]         # 回到初始位置
]
```

Fig. 15. Important Node

```
def interpolate_joint_positions(start_positions, end_positions, alpha):
    return [(1 - alpha) * start + alpha * end for start, end in zip(start_positions, end_positions)]
```

Fig. 16. Interpolation Function

more frequently, thus achieving smoother and more natural movements. (Insufficient loop frequency may result in the robotic arm not receiving new commands after completing one instruction, leading to uncertainty about its next move). Nonetheless, solely increasing the loop frequency might cause the robotic arm to move too swiftly. To tackle this, we can proportionally increase the step size.

Subsequently, we subscribe to the class messages of the YOLO-recognized images and classify them for discussion, aligning with the execution of the specified robotic arm actions.

```
def yolov8_callback(data):
    global detected_object
    detected_object = None  # 初始化为None
    for box in data.bounding_boxes:
        if box.Class == "person":
            detected_object = "person"
            break  # 只要识别到人，就跳出循环
        elif box.Class == "dog":
            detected_object = "dog"
            break  # 只要识别到狗，就跳出循环

def interpolate_joint_positions(start_positions, end_positions, alpha):
    return [(1 - alpha) * start + alpha * end for start, end in zip(start_positions, end_positions)]

if __name__ == "__main__":
    rospy.init_node("joint_state_publisher_sy")

    joint_state_pub = rospy.Publisher("joint_states", JointState, queue_size=10)
    rospy.Subscriber("/yolov8/BoundingBoxes", BoundingBoxes, yolov8_callback)
```

Fig. 17. Image recognition

Finally, we consolidate the startup of YOLO recognition and the execution of the RViz simulation into a single launch file.