EMOTIONAL ARM - ROBOT ARM INTERATIONS WITH HUMAN

Jiaming Qi 224040202, Shiwei He 224040234, Tianshun Zhou 224040264
School of Data Science, MAIR
Chinese University of Hong Kong, Shenzhen

GitHub repository address https://github.com/Clipperrr/sagittariu_ws.git

1 PROJECT OVERVIEW

1.1 BACKGROUND

With the rapid development of artificial intelligence technology, Large Language Models (LLMs) have become a cornerstone in various cutting-edge applications. These models are now widely integrated into search engines, online education platforms, and human-computer interaction systems due to their powerful capabilities in understanding and generating natural language. Their ability to process vast amounts of data and provide contextually relevant responses makes them essential tools for improving user experience, facilitating intelligent communication, and enabling personalized digital services.

In light of these advancements, we propose the innovative integration of a robotic arm with a large-scale language prediction model. This integration aims to create a robotic system capable of understanding and responding to human natural language commands. By combining the physical manipulation capabilities of a robotic arm with the cognitive and semantic processing power of a fine-tuned LLM, the system will be able to interpret user intentions, perform precise tasks, and engage in interactive dialogues, paving the way for more intuitive and natural human-robot collaboration.

1.2 OBJECTIVES

The core objective is to develop an intelligent robotic arm system that not only understands semantics but also responds with emotional awareness. This system will leverage a 4-bit quantized and fine-tuned LLM to ensure efficient deployment on resource-constrained hardware, without sacrificing language comprehension. Coupled with computer vision, the robotic arm will be able to recognize objects, analyze its environment, and execute manipulation tasks. Furthermore, by incorporating emotional feedback mechanisms—such as recognizing user tone or facial expressions—the robot can adjust its behavior accordingly, enabling more empathetic and human-like interactions.

1.3 TECHNICAL HIGHLIGHTS

Multimodal Fusion Language understanding + Visual perception + Robotic control

The intelligent robotic arm system is built upon a multimodal fusion framework that seamlessly integrates natural language understanding, computer vision, and robotic control. By combining these three capabilities, the system can interpret complex verbal instructions, perceive its physical environment through cameras and sensors, and execute precise motor actions. This fusion allows the robot to go beyond simple scripted behavior—enabling it to interact dynamically with objects and users based on both what it sees and what it hears. The result is a more flexible and adaptive human-robot interaction system capable of performing real-world tasks with contextual awareness.

Efficient Fine-tuning QLoRA 4-bit quantization reduces GPU memory requirements

To deploy large language models efficiently on edge devices or systems with limited computational resources, we adopt QLoRA (Quantized Low-Rank Adaptation), a state-of-the-art parameter-efficient fine-tuning technique. By applying 4-bit quantization, QLoRA significantly reduces GPU memory usage during training and inference without compromising the model's performance. This allows for the fine-tuning of powerful LLMs on modest hardware while maintaining high-quality language understanding. As a result, the system achieves both cost-effectiveness and deployment scalability, making advanced AI applications more accessible in robotics.

Real-time Interaction Low-latency coordination between semantic parsing and robotic control

For a natural and fluid user experience, the system is designed to support real-time interaction through tightly synchronized coordination between semantic parsing and robotic control. Once the user gives a command, the language model rapidly interprets the intent and translates it into executable robotic actions with minimal delay. The low-latency pipeline ensures the robotic arm responds promptly, enabling smooth dialogues and instant reactions to environmental changes. This real-time responsiveness is essential for tasks that require quick decision-making, especially in interactive scenarios involving humans.

2 Core Technical Implementation

2.1 LLM Interface Module

Key Function Semantic parsing and object matching

```
# Quantized Inference Optimization
payload = {
    "model": "deepseek-r1:14b",
    "options": {
        "temperature": 0.1,  # Low randomness ensures stability
        "top_p": 0.9,
        "max_tokens": 600
    }
}

# Semantic Parsing Workflow
def deepseek_chat():
    → Receive user command ("I need the blue block")
    → Extract object keyword ("blue")
    → Return highest relevance object
```

Figure 1: LLM Interface Module

The core function of this module is semantic parsing and object matching. It utilizes a large language model (LLM) to interpret natural language commands from users and extract key object-related information to guide subsequent robotic control tasks. The payload configuration shown uses the "deepset-rl:14b" model, with parameters such as a low temperature (0.1) to ensure response stability, and top_p and max_tokens to control output diversity and length. This setup balances output quality with the constraints of quantized inference, optimizing both memory efficiency and response latency.

The semantic parsing process is implemented through the deepspeak_chat() function, which follows three main steps: first, it receives a natural language command from the user (e.g., "I need the blue block"); second, it extracts the key object-related keyword (e.g., "blue"); and third, it returns the object with the highest relevance. This workflow forms a closed loop from language understanding to physical action, enabling the robotic system to flexibly interpret user intentions and perform accurate tasks. It significantly enhances the system's naturalness and effectiveness in human-computer interaction.

2.2 Vision Control Module

YOLO-based Object Detection Using YOLO model to obtain object names and coordinates

```python
global current_target, obj_list
try:
    cv_image = CvBridge().imgmsg_to_cv2(data, "bgr8")
except CvBridgeError as e:
    print(e)
cv2.imshow("source", cv_image)
cv2.waitKey(1)

if current_target['Found']:
    return

try:
    results = model.predict(cv_image)
except Exception as e:
    print(f"YOLO预测异常: {str(e)}")
    return

annotated_frame = results[0].plot()
cv2.imshow("YOLO Detection", annotated_frame)
obj_list = {}
obj_names = ''
for result in results:
    boxes = result.boxes
    names = model.names
    for box in boxes:
        x1, y1, x2, y2 = box.xyxy[0]
        cls_id = int(box.cls[0])
        obj_name = names[cls_id]
        obj_names += str(obj_name) + ' , '
        obj_list[obj_name] = {
            'x': float((x1 + x2) / 2),
            'y': float((y1 + y2) / 2),
        }
```

Figure 2: YOLO-based Object Detection

Three-stage Motion Stability Optimization

Morphological filtering (Erosion + Dilation)

To enhance the stability of object detection in dynamic video streams, we apply morphological filtering during the image preprocessing stage—specifically, erosion followed by dilation. The erosion operation effectively removes small noise particles and edge artifacts, helping to eliminate false positives or unstable contours. Dilation then restores the core structure of the detected object to ensure that the region of interest remains complete. This combination helps maintain consistent object shapes across frames, providing cleaner and more reliable inputs to the YOLO detection model.

Gaussian noise reduction

In order to reduce the interference of sensor noise, lighting changes, or background disturbances on object detection, we apply Gaussian blur processing to each frame of the image. Gaussian filters can smooth out random noise in images, reduce pixel level abrupt changes, and thus improve image quality and consistency of target edges. This operation is particularly crucial for YOLO models, as it can effectively reduce the probability of misjudgment caused by background clutter or image jitter, and improve the stability of target position and category recognition.

Frame validation (more than 30 stable frames)

After completing object detection, we designed a frame level stability verification mechanism to ensure that the robotic arm does not produce unstable responses due to instantaneous false detections or brief target appearances. Specifically, the system only considers the same target as a ”valid target” and sends it to the control module after stably detecting it for more than 30 consecutive frames (with consistent category labels and coordinate change ranges). This

strategy greatly enhances the robustness of the system to temporary occlusion, motion blur, and environmental interference, ensuring that the robotic arm only makes operational decisions on stable and persistent targets.

2.3 Model Fine-tuning Module

Quantization Configuration:

```python
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,  # Nested quantization
    bnb_4bit_quant_type="nf4",       # 4-bit NormalFloat
    bnb_4bit_compute_dtype=torch.bfloat16
)

# LoRA Adapter Configuration
peft_config = LoraConfig(
    r=24,               # Rank dimension
    lora_alpha=48,      # Scaling factor
    target_modules=["q_proj", "v_proj", "k_proj"]  # Attention adaptation
)
```

Figure 3: Quantization Configuration

Training Optimization:

To ensure efficient training of the intelligent robotic system, we have implemented a series of optimization techniques aimed at improving both training speed and memory usage, enabling the system to handle complex tasks with limited computational resources. The following optimization strategies have been adopted.

Cosine annealing learning rate scheduling (with restarts)

The training process utilizes a cosine annealing learning rate schedule, a technique designed to gradually reduce the learning rate as training progresses. This method helps the model converge more smoothly and avoid overshooting the optimal solution, leading to better generalization. Unlike traditional learning rate schedules, the cosine annealing strategy adjusts the learning rate in a way that allows the model to explore the parameter space more effectively in the early stages and fine-tune more precisely as it approaches convergence.

Gradient checkpointing for memory reduction

Given the memory constraints of edge devices, particularly when training large models, gradient checkpointing is employed to optimize memory usage during the training process. This technique reduces the memory footprint by storing only a subset of intermediate activations during the forward pass and recomputing them during the backward pass as needed. While this introduces additional computation, the trade-off allows for training models with larger architectures without running out of memory. By strategically selecting which activations to store and which to recompute, gradient checkpointing makes it feasible to train large models on devices with limited GPU memory.

Mixed-precision training acceleration

To further accelerate training while reducing memory usage, mixed-precision training is adopted. This technique involves using lower-precision arithmetic for parts of the model's computation, while keeping critical operations in higher precision. By leveraging the benefits of both lower-precision and higher-precision calculations, mixed-precision training achieves faster computation and lower memory usage without significantly sacrificing model accuracy. In the context of robotic control, where responsiveness is key, this acceleration ensures that the robotic arm can make decisions quickly and act with minimal latency, all while reducing the computational load.

3 Experimental Results

3.1 Performance Metrics

Metric	Before Fine-tuning	After Fine-tuning
Inference Latency	540ms	320ms
Object Recognition Accuracy	85.4%	92.4%
Grasping Success Rate	73.6%	88.7%

The experimental results clearly demonstrate the effectiveness of the proposed optimization strategies and system design. Improvements are evident across all key performance metrics, reflecting enhancements in inference speed, perception accuracy, and robotic manipulation reliability.

Inference Latency: Reduced from 540ms to 320ms

The reduction in inference latency by over 40% highlights the efficiency gains brought by QLoRA 4-bit quantization, mixed-precision training, and a streamlined semantic-to-control pipeline. This latency reduction is crucial for real-time interaction, as it enables the robotic arm to respond more promptly to user commands and dynamic environmental changes. In high-speed human-robot collaboration scenarios, this improvement significantly enhances fluidity and responsiveness.

Object Recognition Accuracy: Improved from 85.4% to 92.4%

The 7% increase in object recognition accuracy can be attributed to the enhanced image preprocessing pipeline—including morphological filtering, Gaussian noise reduction, and temporal frame validation. These techniques collectively ensure that the YOLO detection model receives cleaner and more stable input, leading to more consistent and accurate recognition. Higher accuracy in perception directly contributes to the reliability of downstream tasks such as grasping and object manipu Grasping lation.

Success Rate: Increased from 73.6% to 88.7%

A 15.1% improvement in grasping success rate represents a significant leap in the robotic system' s practical effectiveness. This result is closely tied to better object detection and recognition, as well as the real-time coordination between semantic parsing and robotic control. By ensuring that only consistently validated targets are acted upon—and by minimizing execution delay—the system can perform more stable and precise grasping actions, even in dynamic or partially occluded environments.

3.2 Use Case Example

```
User Input: "Bring me the block that makes me happy"
→ Model Output: "green" (green represents happiness)
→ Vision locates green block
→ Robotic arm picks up and performs celebratory motion
```

Figure 4: Use Case Example

4 Challenges Improvements

4.1 Current Limitations

Recognition stability in multi object environment

While the object detection pipeline demonstrates high accuracy in single-object scenarios, its stability declines when multiple similar or overlapping objects are present in the scene. The YOLO model occasionally produces inconsistent class predictions or bounding box jitter across frames

when objects are partially occluded or closely clustered. This instability can lead to unreliable inputs for downstream robotic control, such as selecting the wrong target or miscalculating grasping positions. Addressing this challenge may require advanced instance segmentation techniques or temporal consistency models to maintain robust object identities across time and space.

Intent misunderstanding in long-text commands

The natural language understanding module occasionally struggles with parsing long or compound user instructions that involve multiple steps, conditional logic, or ambiguous phrasing. In such cases, the language model may extract only partial intent or misinterpret the sequence of required actions, leading to incorrect or incomplete execution by the robotic arm. While the QLoRA-fine-tuned language model performs well on short, direct commands, future work should incorporate task planning and dialogue clarification modules to improve comprehension of more complex linguistic inputs.

Robotic arm trajectory optimization needs

The current control policy for arm movement focuses primarily on reaching and grasping targets but does not always generate the most efficient or smoothest trajectories. This may result in unnecessary joint movements, slower task completion, or reduced mechanical lifespan due to repetitive stress. Moreover, in cluttered environments, the trajectory planning may lack collision-awareness, increasing the risk of unintended contact with surrounding objects. Incorporating optimization algorithms such as model predictive control (MPC), reinforcement learning-based trajectory refinement, or dynamic obstacle avoidance would significantly enhance motion planning efficiency and safety.

4.2 Optimization Roadmap

Vision Enhancement: Add RGB-D depth sensing

To address current limitations in perception—especially in multi-object and cluttered environments—we plan to enhance the visual sensing module by integrating RGB-D cameras that capture both color (RGB) and depth (D) information. Depth data provides crucial spatial context, allowing the system to distinguish overlapping objects, estimate object distances more accurately, and improve 3D scene understanding. This enhancement will not only improve object detection stability but also enable more accurate grasp planning by identifying object geometries and surface orientations. The fusion of RGB and depth modalities will lay the foundation for more advanced perception tasks such as object segmentation, spatial reasoning, and environmental mapping.

Control Optimization: Implement collision detection and develop emotional motion library (10+ preset actions)

On the control side, we aim to improve both the safety and expressiveness of the robotic arm. First, we will implement a collision detection mechanism, using sensor feedback and spatial modeling to prevent unintended contact with objects or humans. This will make the robot safer to operate in dynamic or human-shared environments. Second, we plan to develop an emotional motion library comprising over 10 preset actions, such as "wave," "nod," "present object," or "express hesitation." These predefined, human-friendly motion patterns can be triggered contextually based on user intent or task outcome, making the robot more intuitive and emotionally engaging. This dual focus on technical safety and affective interaction will greatly improve the human-robot interaction experience.

5 Appendices

5.1 GitHub repository address

https://github.com/Clipperrr/sagittariu_ws.git

5.2 Contribution Statement

Jiaming Qi Code writing and interface debugging of robot control program

Shiwei He Programming and fine-tuning development related to large models

Tianshun Zhou Report writing, code testing, and presentation