Paper examples

1.Large Language Models for Robotics Opportunities, Challenges, and Perspectives.pdf

Example 1: Google's SayCan for Language-Guided Household Robotics

The paper highlights SayCan, a system deployed on real mobile manipulation robots in household environments. SayCan combines a large language model with learned robot skills to translate open-ended natural language instructions into executable actions. The robot successfully performs tasks such as fetching objects, opening drawers, and placing items, demonstrating a working integration of LLM reasoning with physical robot capabilities [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

Example 2: PaLM-E for Multi-Modal Robotic Decision-Making

PaLM-E is presented as a real embodied system where a language model is directly connected to robot sensor inputs, including images and state vectors. The robot uses PaLM-E to answer questions about the environment and to decide manipulation actions in real time. This example shows how LLMs move beyond planning to become a continuous decision-making module embedded inside physical robots [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

Example 3: Language-Driven Navigation with LM-Nav

The paper describes LM-Nav, a system tested on real robotic navigation platforms that follow high-level language instructions such as "go to the kitchen and stop near the table." The robot uses language–vision embeddings to ground abstract commands in real environments without requiring trajectory-level supervision. This demonstrates practical deployment of LLM-guided navigation in complex indoor spaces [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

Example 4: GPT-Based Planning for Pick-and-Place Manipulation

Several experiments discussed in the paper show LLMs generating structured pick-and-place plans executed on real robot arms. The model identifies objects, reasons about spatial relations, and sequences actions such as grasping, lifting, and placing. These systems operate in cluttered real-world scenes, illustrating how LLMs reduce manual task scripting in industrial-style manipulation pipelines [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

Example 5: VoxPoser for Interactive Human-in-the-Loop Control

VoxPoser is presented as a system where users guide a robot through spoken natural language corrections during execution. The robot updates its manipulation trajectory in real time based on verbal feedback. This example reflects practical industrial relevance, where non-expert operators can adapt robot behavior without reprogramming or halting the system [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

Example 6: Visual-Language Pretraining for Industrial Manipulation (R3M, LIV)

The paper discusses models such as R3M and LIV, which are trained on large-scale human video data and then transferred to real robotic systems. These representations improve manipulation success rates while reducing the need for robot-specific data collection. This approach aligns closely with industrial needs, where data efficiency and scalability are critical constraints [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

Example 7: Hybrid LLM + Classical Planning Systems in Real Robots

The paper reports multiple systems where LLMs generate high-level plans that are then validated and executed by classical planners or controllers. This hybrid architecture is tested on real robots to ensure physical feasibility and safety. It demonstrates a realistic deployment strategy where LLMs augment—not replace—traditional robotics stacks [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

Example 8: Long-Horizon Task Execution in Real Environments

Finally, the paper documents real-world experiments where robots execute long-horizon, multi-step tasks (e.g., multi-object rearrangement) using LLM-generated plans. The robots maintain task context, recover from errors, and complete goals without explicit step-by-step programming. This example illustrates how LLMs reduce engineering overhead in complex robotic workflows [Large Language Models for Robotics: Opportunities, Challenges, and Perspectives].

2.Parsing Natural Language Sentences into Robot Actions.pdf

Example 1: Natural Language Control of a Real Humanoid Robot (NAO / Zora)

The paper presents a fully implemented system where a real humanoid robot (Zora, built on NAO) executes spoken natural language commands from users. Commands such as moving arms, legs, head, or body are parsed and translated into physical actions. This system demonstrates real-time human–robot interaction without requiring users to know robot-specific commands or programming [Parsing Natural Language Sentences into Robot Actions].

Example 2: Clarification Dialogue for Incomplete Commands

When a user gives an incomplete instruction such as *“raise your arm”*, the robot does not fail silently. Instead, it asks a follow-up question (e.g., “left or right?”) to clarify the missing information. This behavior is implemented and tested on the real robot, showing a practical dialogue-based control loop suitable for real-world human interaction [Parsing Natural Language Sentences into Robot Actions].

Example 3: Stateful vs. Stateless Execution in Physical Robots

The system supports two real execution modes. In stateless mode, the robot performs each action and returns to a default posture. In stateful mode, actions are executed sequentially without resetting posture. This enables realistic multi-step behaviors and reflects how industrial and service robots must maintain internal state across commands [Parsing Natural Language Sentences into Robot Actions].

Example 4: Automatic Rejection of Unsafe or Impossible Commands

The robot actively refuses commands that would cause physical instability, such as asking the robot to lift a leg while already standing on the other leg. The system checks the current physical state and rejects incompatible actions with a spoken explanation. This is a concrete safety mechanism implemented and validated on real hardware [Parsing Natural Language Sentences into Robot Actions].

Example 5: Execution of Compound Natural Language Commands

The system can parse and execute compound commands involving multiple actions in a single sentence. For example, commands like *“stand up and move your arms down”* are decomposed into sequential robot actions. This capability was tested directly on the robot, showing how natural language can control complex behaviors without scripting [Parsing Natural Language Sentences into Robot Actions].

Example 6: Ontology-Driven Mapping Between Language and Robot Actions

A domain ontology explicitly models robot body parts, actions, and incompatibilities, enabling reliable mapping from language to motion. The ontology contains dozens of action and body-part entities and is queried at runtime to determine valid executions. This approach reflects a deployable engineering solution rather than a purely learning-based prototype [Parsing Natural Language Sentences into Robot Actions].

Example 7: Cloud-Based NLP Pipeline for Real-Time Robot Control

The robot records speech locally, converts it to text, and sends it to a cloud-based NLP engine for parsing and reasoning. The resulting action plan is then returned and executed on the physical robot. This architecture mirrors real industrial deployments where heavy computation is offloaded to cloud services [Parsing Natural Language Sentences into Robot Actions].

Example 8: Explicit Modeling of Physical Incompatibilities

All combinations of robot postures and actions were tested to define a list of physically incompatible action sequences (e.g., actions that would cause the robot to fall). This list is used at runtime to prevent unsafe execution. This represents a practical, safety-aware deployment rather than a simulated or hypothetical system [Parsing Natural Language Sentences into Robot Actions].

3.A Review of Natural-Language-Instructed Robot Execution Systems.pdf

Example 1: Voice-Controlled Industrial Robot Arms in Manufacturing

The review reports multiple systems where industrial robot arms are controlled using spoken natural language commands such as “start,” “stop,” or “move faster.” These systems replace traditional joysticks and control panels, allowing operators to guide robots hands-free during manufacturing tasks. Such deployments reduce operator workload while maintaining precise execution in real industrial environments [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 2: Natural Language Programming of Industrial Assembly Tasks

Several industrial systems described in the paper allow technicians to define assembly procedures verbally, such as “pick the part, then place it into the slot.” Robots parse these instructions into structured execution plans and perform assembly without manual coding. This approach has been tested in real factory-like settings and demonstrates how language lowers the barrier to robot programming [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 3: Language-Guided Indoor and Outdoor Robot Navigation

The review summarizes real navigation robots that follow spoken instructions like “go to the kitchen,” “turn left at the hallway,” or “navigate to the building behind the people.” These systems have been deployed in indoor buildings and outdoor campuses, showing reliable grounding of spatial language into navigation actions [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 4: Assistive Robots for Daily Living and Elderly Care

NL-instructed robots are reported in daily assistance scenarios where users request tasks such as object delivery or navigation help through speech. These robots operate in real homes and care facilities, enabling non-expert users to interact naturally without training. The systems demonstrate practical deployment in healthcare and assistive environments [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 5: Human-in-the-Loop Error Recovery Through Language

The paper highlights systems where robots actively ask humans for help when encountering failures, using phrases like “I cannot reach the object” or “please hand me the tool.” These robots are deployed in real task environments, including assembly and manipulation tasks, enabling robust recovery instead of task failure [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 6: Natural Language Training of Robot Skills

Several robots described in the review are trained using spoken language to learn object properties, actions, and task procedures. For example, robots learn what “cup,” “heavy,” or “fragile” means through verbal instruction combined with execution. This training paradigm has been applied to real robots performing manipulation and delivery tasks [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 7: Human-Centered Collaborative Assembly Systems

In human-centered execution systems, robots assist humans during tasks like furniture or device assembly by responding to requests such as “give me the wrench” or “hold this part.” These systems are implemented in real collaborative workspaces, where robots physically assist while humans retain task-level control [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 8: Robot-Centered Execution with Human Assistance

The review also reports robot-centered systems where robots autonomously execute tasks and request human help only when needed. Examples include robots moving heavy objects or navigating constrained spaces while asking humans for physical assistance. These systems have been demonstrated in industrial and outdoor environments [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 9: Socially Aware Service and Companion Robots

The paper documents real service robots operating as receptionists, caregivers, and companions that use socially appropriate language. These robots adjust speech, motion, and execution behavior based on social context, such as slowing down in crowded areas or using polite language. Such systems have been deployed in public spaces, hospitals, and educational settings [A Review of Natural-Language-Instructed Robot Execution Systems].

Example 10: Language-Guided Robots in Medical and Surgical Contexts

Finally, the review discusses medical robots that respond to spoken instructions during procedures, adjusting camera positions or execution behavior based on surgeon commands and emotional cues. These systems operate in real clinical environments and demonstrate the safety-critical relevance of NL-instructed robot execution [A Review of Natural-Language-Instructed Robot Execution Systems].

Why this paper matters in your narrative

This review clearly shows that robotics already had many "language moments" before LLMs — but they were distributed across domains, incremental, and engineering-heavy.
What modern LLMs change is unification and scalability, not the invention of language-driven robots from scratch.

4.Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming.pdf

Real-World and Practical Examples from Domain-Specific Fine-Tuning of LLMs (Paper 4)

Example 1: Natural-Language Programming Assistant for Industrial Robots

The paper presents a working natural-language assistant integrated into the ArtiMinds Robot Programming Suite, a real industrial IDE used for programming robots. Engineers can ask questions in plain English about robot skills and receive explanations directly related to executable robot programs. This demonstrates a concrete step toward replacing expert-only robot programming with language-based interaction [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

Example 2: Language-Based Explanation of Industrial Robot Skills

The assistant is trained to explain complex industrial robot skills such as *"Move to State"*, *"Grasp"*, or *"Peg in Hole"*. These explanations describe real robot motion, collision avoidance, force control, and interaction with the environment. This allows engineers to understand advanced robot behavior without reading long technical manuals, directly supporting real industrial workflows [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

Example 3: Step-by-Step Description of Physical Robot Behavior

For high-level templates like *"Grasp"*, the assistant produces step-by-step descriptions of what the robot will physically do, including approach motion, gripper control, contact, and retreat. These descriptions correspond to actual executable robot programs compiled by the IDE. This bridges the gap between abstract language and real robot motion in production systems [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

Example 4: Natural Language Guidance for When to Use Specific Robot Skills

The assistant answers questions such as *“When should I use a ‘Move to State’ template?”* by providing domain-specific guidance tied to real industrial scenarios, including collision-prone environments. This helps practitioners choose the correct robot skill without trial-and-error, reflecting real decision-making needs in factory automation [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

Example 5: Domain-Specific Fine-Tuning with Limited Industrial Data

The paper demonstrates that domain adaptation is possible using only a few hundred instruction–answer pairs extracted from real industrial documentation. This reflects realistic constraints in industrial settings, where large datasets are unavailable. The assistant is trained using QLoRA on a single server-grade GPU, making the approach feasible for small and medium-sized enterprises [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

Example 6: Evaluation by Professional Robotics Engineers

Unlike many academic prototypes, the assistant is evaluated through a user study involving 33 industry engineers. Participants assessed factual correctness, domain adherence, and usefulness of the assistant’s answers. This evaluation reflects real-world expectations and exposes practical limitations such as hallucinations and incomplete explanations [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

Example 7: Comparison of Fine-Tuning Strategies for Industrial Deployment

The paper compares multiple fine-tuning strategies (instruction-tuned models, streaming models, and merged approaches) under industrial compute constraints. Results show that instruction-following models fine-tuned on domain data perform best, guiding realistic deployment choices rather than theoretical optimization [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

Example 8: Identified Limitations in Practical Industrial Use

The authors explicitly report that even the best-performing models still hallucinate, repeat phrases, or drift out of domain, making them unsuitable for unsupervised deployment. This honest assessment reflects real industrial validation rather than laboratory success and highlights why current systems require human oversight [Domain-Specific Fine-Tuning of Large Language Models for Interactive Robot Programming].

5.RT-Grasp Reasoning Tuning Robotic Grasping via Multi-modal Large Language Model.pdf

Example 1: Direct Numerical Grasp Prediction Using a Multi-Modal LLM

RT-Grasp demonstrates a system where a multi-modal LLM predicts exact numerical grasp poses [x,y,θ][x, y, θ][x,y,θ] instead of high-level plans or text descriptions. This overcomes a major limitation of prior LLM-based robotics systems. The predicted grasp poses are executed directly on a real robot arm, bridging the gap between reasoning and physical control [RT-Grasp].

Example 2: Real Robot Experiments on a Franka Emika Panda Arm

The authors evaluate RT-Grasp on a 7-DoF Franka Emika Panda robot equipped with a parallel gripper and an RGB camera. The robot performs grasping tasks on 27 household objects in multiple positions and orientations. This is a full hardware deployment, not a simulation, demonstrating real-world feasibility [RT-Grasp].

Example 3: Robust Grasping of Unseen Household Objects

All 27 test objects used in real-world experiments were unseen during training, including toys, tools, and everyday items. RT-Grasp successfully grasps each object at least once, showing strong generalization beyond training data. This directly addresses a key industry challenge in robotic manipulation [RT-Grasp].

Example 4: Reasoning-Before-Prediction for Safer Grasping

RT-Grasp introduces a reasoning phase where the model first explains the grasping strategy based on object type and shape before outputting numbers. For example, it reasons that a screwdriver should be grasped by the handle rather than the metal tip. This reasoning improves safety and practical execution on real robots [RT-Grasp].

Example 5: Interactive Refinement Through Natural Language

The system allows users to refine grasp behavior through language after the initial prediction. When instructed to avoid unsafe regions (e.g., “avoid the metallic tip”), RT-Grasp generates a new numerical grasp pose accordingly. This interaction loop is demonstrated on real hardware, not just offline prediction [RT-Grasp].

Example 6: Improved Real-World Performance Over Traditional CNN Methods

In real-world tests, RT-Grasp achieves grasp success rates of 80.0% (pre-training) and 83.7% (LoRA fine-tuning), comparable to traditional CNN-based methods. Unlike CNNs, RT-Grasp maintains performance when objects differ in shape or category, reflecting better adaptability in physical environments [RT-Grasp].

Example 7: Language-Guided Adaptation to Object Semantics

RT-Grasp uses language to reason about object semantics such as *fragility, shape, and function*. For instance, it grasps cups by the handle or upper edge depending on user instruction. This semantic awareness allows the robot to adapt its behavior in ways that purely vision-based methods cannot [RT-Grasp].

Example 8: Efficient Fine-Tuning for Industrial Feasibility

The paper demonstrates that RT-Grasp can be trained using LoRA fine-tuning, keeping the core LLM frozen and updating only lightweight parameters. This significantly reduces computational cost, making the approach practical for industrial or research labs without large-scale compute infrastructure [RT-Grasp].

6.ARRC Advanced Reasoning Robot Control—Knowledge-Driven Autonomous Manipulation Using Retrieval-Augmented Generation .pdf

Example 1: End-to-End Language-to-Action Control on a Real Robot Arm

ARRC is deployed on a UFactory xArm 850, where natural language instructions are converted into structured JSON action plans and executed directly on hardware. The system integrates perception, planning, and control without simulation-only shortcuts. This demonstrates a complete, real-world pipeline from language to physical manipulation [ARRC: Advanced Reasoning Robot Control].

Example 2: Retrieval-Augmented Planning Using Robot-Specific Knowledge

Instead of relying on the LLM alone, ARRC retrieves robot-centric knowledge such as movement templates, safety heuristics, and task patterns from a vector database. This retrieved context conditions the LLM at inference time. The approach enables adaptation to new tasks without retraining, a key requirement in industrial robotics [ARRC].

Example 3: Vision-Grounded Manipulation with RGB–D and AprilTags

ARRC uses an Intel RealSense D435 RGB–D camera combined with AprilTags to estimate precise 3D object poses in the robot frame. These metric poses are directly used to generate collision-free approach and grasp motions. This tight perception–planning coupling is validated on real tabletop manipulation tasks [ARRC].

Example 4: Structured JSON Plans for Safe Execution

The system forces the LLM to output plans in a restricted JSON schema, consisting of bounded actions such as SCAN, APPROACH, GRASP, and RETREAT. Each action is validated before execution. This design ensures interpretability, debuggability, and safe integration with existing robot control stacks [ARRC].

Example 5: Multi-Layer Software Safety Gates During Execution

ARRC enforces multiple safety constraints during execution, including workspace bounds, velocity and acceleration limits, gripper force thresholds, timeouts, and bounded retries. Even if the LLM produces a risky plan, unsafe motions are filtered out. This reflects real industrial safety requirements rather than laboratory assumptions [ARRC].

Example 6: Adaptive Reasoning Under Partial Occlusion

In experiments involving a partially occluded screwdriver, the robot initially fails to detect the object using a horizontal scan. ARRC automatically switches to an arc scanning strategy, detects the object from a new viewpoint, and successfully completes the pick-and-place task. This behavior is demonstrated and measured on real hardware [ARRC].

Example 7: Autonomous Failure Recovery Through Re-Prompting

When an execution step fails (e.g., object not found or low confidence), ARRC re-prompts the retrieval and planning modules with failure context. The system then generates an alternative plan rather than aborting. This closed-loop reasoning is executed online during real robot operation [ARRC].

Example 8: Quantitative Evaluation of Real-World Task Success

ARRC is evaluated across repeated real-world trials of scan, approach, and pick-and-place tasks. Results show 80% plan validity, 87.1% average approach accuracy, and 80% pick-and-place success, with consistent execution times. These metrics reflect real physical performance, not simulation scores [ARRC].

Example 9: Separation of High-Level Reasoning and Low-Level Control

In ARRC, high-level reasoning (LLM + retrieval) can run in the cloud or locally, while perception and low-level control always remain on the robot. This architecture balances adaptability with real-time safety and reliability, matching how industrial robotic systems are actually deployed [ARRC].

Example 10: Updating Robot Capabilities Without Retraining Models

New task knowledge, safety rules, or motion strategies can be added by updating the knowledge base, not by retraining the LLM. This allows rapid adaptation in production environments where retraining is expensive or infeasible. ARRC demonstrates this principle in a fully operational manipulation system [ARRC].

7.ROS-LLM A ROS framework for embodied AI with task feedback and structured reasoning.pdf

Example 1: Non-Experts Programming a Real Robot via Natural Language

ROS-LLM enables non-expert users to program robots through a chat-based natural language interface. Users describe tasks such as “make me a coffee” or “rearrange the boxes,” and the system generates executable robot behaviors. This interaction is tested on real robot hardware, demonstrating a practical reduction in reliance on expert robot programmers [ROS-LLM].

Example 2: Long-Horizon Coffee-Making Task in a Real Kitchen Environment

In a kitchen-like setup, a UR5 robot arm with a Robotiq gripper successfully executes the full task “make me a coffee.” The robot performs over ten sequential actions including opening cabinets, picking up a mug, scooping coffee grounds, and operating a coffee machine. This experiment validates real-world long-horizon task execution driven by language [ROS-LLM].

Example 3: Tabletop Rearrangement with Increasing Task Complexity

The paper reports real experiments where the robot rearranges colored boxes based on spoken instructions. Task difficulty increases from 2 to 8 objects, requiring precise sequencing and spatial reasoning. The robot executes these tasks on physical hardware, highlighting scalability and robustness in realistic manipulation scenarios [ROS-LLM].

Example 4: Policy Correction Through Human Feedback During Execution

When the robot makes a mistake—such as picking the wrong object or blocking its own camera—the human provides corrective feedback in natural language. The system incorporates this feedback and re-executes the task successfully. This closed-loop correction is demonstrated repeatedly on a real robot, not in simulation [ROS-LLM].

Example 5: Learning New Robot Skills via Imitation from Non-Experts

Non-expert users teach the robot new atomic actions (e.g., stirring, pouring, wiping) through kinesthetic teaching or teleoperation. These demonstrations are converted into executable policies and added to the robot's action library. The robot later reuses these learned skills to complete new long-horizon tasks like “make me pasta” [ROS-LLM].

Example 6: Adaptation to Dynamic Environment Changes

In one experiment, the environment is deliberately altered mid-task by moving the target object. The robot initially fails, receives human feedback, and adapts its behavior to complete the task. In subsequent trials, the robot autonomously applies the learned correction without additional human input, demonstrating continual learning on real hardware [ROS-LLM].

Example 7: Remote Supervisory Control Across Continents

ROS-LLM is used for remote robot control, where an operator in Europe supervises a robot located in Asia using natural language commands. Despite 2–3 seconds of network latency, participants successfully guide the robot to perform pick-and-place tasks while avoiding obstacles. This demonstrates feasibility for real remote operation scenarios [ROS-LLM].

Example 8: Real-Time Integration with ROS Action Libraries

The system integrates directly with ROS actions and services, converting LLM outputs into executable Python scripts, action sequences, or behavior trees. These outputs are validated and run on a physical UR5 robot. This tight ROS integration reflects how robotic systems are actually deployed in industry and research labs [ROS-LLM].

Example 9: Human Usability Evaluation in Real Robot Experiments

Seven human participants evaluated the system during remote supervisory control tasks. Task completion time and cognitive load were measured using the NASA-TLX framework. Results show that language-based control is usable and effective, even under latency and perception constraints, providing real human-centered validation [ROS-LLM].

Example 10: Deployment “In the Wild” Beyond the Lab

The paper reports two real use cases beyond controlled experiments:
(1) a humanoid robot cooking in a kitchen using language instructions, and
(2) integration into the Robot Air Hockey Challenge using KUKA IIWA arms.

These examples demonstrate early-stage deployment in real-world and competitive environments [ROS-LLM].

8.Large Language Models for Robotics A Survey.pdf

Example 1: PaLM-SayCan for Real Household Task Execution

The survey reports PaLM-SayCan as a deployed system where a robot executes household tasks such as fetching objects and opening refrigerators. The robot decomposes natural language instructions into executable subtasks based on affordance and feasibility. This system operates in real environments, showing how LLM reasoning can guide physical task execution rather than remain purely symbolic [Large Language Models for Robotics: A Survey].

Example 2: PaLM-E as an Embodied Multimodal Robot Brain

PaLM-E is described as a real embodied system that integrates vision, language, and robot state into a single model. Robots use PaLM-E to answer questions about their environment and generate low-level control instructions. This demonstrates a practical step toward end-to-end perception–decision–action pipelines using LLMs [Large Language Models for Robotics: A Survey].

Example 3: LM-Nav for Language-Based Robot Navigation

The survey highlights LM-Nav, a navigation system tested on real robots that follow free-form language commands such as “go to the kitchen.” The robot combines language models, vision-language embeddings, and navigation models without requiring labeled trajectory data. This shows real-world navigation driven by language rather than manual mapping [Large Language Models for Robotics: A Survey].

Example 4: RT-1 for Large-Scale Real-World Robot Control

RT-1 is presented as a robotics transformer trained on large-scale real robot data to perform manipulation tasks in physical environments. The model directly maps images and language instructions to robot actions in real time. This represents an industrial-scale attempt to generalize robot control across many tasks [Large Language Models for Robotics: A Survey].

Example 5: RT-2 for Transferring Web Knowledge to Robot Actions

RT-2 extends RT-1 by fine-tuning a vision–language model on robot trajectories, allowing robots to use knowledge learned from the web. The survey reports that robots execute previously unseen tasks by leveraging semantic understanding rather than task-specific training. This marks a concrete step toward scalable generalization in real robots [Large Language Models for Robotics: A Survey].

Example 6: Expedition A1 Humanoid Robot with LLM-Based Brain

The survey describes Expedition A1, a humanoid robot designed for industrial manufacturing tasks. It integrates an LLM-based “super brain” with local control modules to manage perception, planning, and execution. This example reflects early industrial deployment of LLMs as centralized reasoning components in humanoid robots [Large Language Models for Robotics: A Survey].

Example 7: Language-Guided Manipulation with Behavior and Decision Transformers

The paper reports multiple systems where transformers generate robot actions from language-conditioned demonstrations. These models are trained and evaluated on real robot platforms for manipulation tasks such as grasping and placing. This shows how language-conditioned control is moving from research to practical execution [Large Language Models for Robotics: A Survey].

Example 8: Human–Robot Collaboration via Language Interfaces

The survey documents robots operating in collaborative settings where humans guide execution through natural language instead of programming interfaces. Robots assist in assembly, navigation, and manipulation tasks while remaining under human supervision. This reflects practical adoption of language interfaces in collaborative robotics [Large Language Models for Robotics: A Survey].

Example 9: Assistive and Service Robots Using Language-Based Interaction

Robots described in the survey operate in service domains such as healthcare, education, and home assistance. These robots interpret spoken instructions, answer questions, and assist users in daily activities. The examples demonstrate that language-driven robotics is already active outside laboratories [Large Language Models for Robotics: A Survey].

Example 10: Incremental Deployment Through Modular Architectures

Rather than full end-to-end autonomy, the survey emphasizes modular deployment, where LLMs handle reasoning while perception and control remain classical. This architecture has been tested in multiple real systems to ensure safety and reliability. It reflects how LLMs are realistically integrated into existing robotic stacks [Large Language Models for Robotics: A Survey].

9. A Multimodal LLM-Driven Robotic Control System for Adaptive Industrial Manipulation: Integrating Vision-Language Models for Enhanced Manufacturing Flexibility

Example 1: Multimodal Command Interpretation for Real Robot Control

The paper presents a robotic system that combines language, vision, and robot state to interpret user commands. Instead of relying only on text, the robot processes camera input and sensor data alongside natural language instructions. This enables the robot to correctly understand commands like object references and spatial relations in real environments, demonstrating an integrated multimodal control pipeline.

Example 2: Language-Guided Manipulation in Physical Environments

The system is evaluated on real manipulation tasks, where the robot performs actions such as grasping, moving, and placing objects based on natural language instructions. The LLM reasons about object attributes and task goals before generating executable action sequences. This shows practical use of LLMs to reduce manual programming in manipulation tasks.

Example 3: Vision-Based Grounding of Natural Language Instructions

A key contribution is grounding language in real visual observations. When a user refers to objects using descriptive language (e.g., color or position), the robot identifies the correct target using camera input. This bridges the gap between abstract instructions and physical object selection, a critical requirement for real-world robotic deployment.

Example 4: Structured Action Generation for Safe Execution

Rather than issuing raw motion commands, the LLM produces structured, high-level action representations that are passed to classical controllers. This separation allows the robot to benefit from LLM reasoning while preserving reliable low-level control. The approach reflects how LLMs can be safely integrated into existing robotics stacks.

Example 5: Closed-Loop Feedback Between Execution and Reasoning

The system continuously feeds execution results back into the reasoning loop. If an action fails or the environment changes, updated sensory information is provided to the LLM, which revises the plan accordingly. This closed-loop design enables robustness in real, dynamic environments rather than one-shot command execution.

Example 6: Error Handling Through Multimodal Context Awareness

When execution does not proceed as expected—such as an object not being reachable—the system detects the issue through sensor feedback. The LLM then reasons about the failure using both linguistic context and visual input, allowing it to adjust the strategy instead of halting. This mirrors real industrial requirements for fault-tolerant robotic systems.

Example 7: Human-Robot Interaction Without Expert Knowledge

The paper emphasizes that users can interact with the robot using natural language, without understanding robot kinematics or control parameters. The multimodal LLM abstracts away technical complexity, enabling intuitive human-robot interaction. This is particularly relevant for service robotics and collaborative industrial environments.

Example 8: Modular Integration of LLMs into Robotic Architectures

The proposed system is designed as a modular architecture, where perception, reasoning, and control are loosely coupled. The LLM operates as a high-level reasoning module rather than a monolithic controller. This design choice supports scalability and aligns with real-world deployment constraints.

10. A Survey of LLM-Driven AI Agent Communication: Protocols, Security Risks, and Defense Countermeasures

Example 1: AutoGPT for Autonomous Task Execution via Agent Communication

The survey discusses AutoGPT as a system where multiple LLM-driven agents communicate through structured message passing to decompose and execute complex tasks. Agents exchange goals, intermediate results, and feedback without human intervention. This system has been practically demonstrated for long-horizon tasks such as software generation and data analysis, showing real multi-agent coordination driven by communication protocols.

Example 2: BabyAGI for Iterative Task Refinement

BabyAGI is presented as an agent framework where agents communicate task states, priorities, and outcomes in an iterative loop. Each agent updates a shared task list based on execution feedback. This communication mechanism enables continuous task refinement and has been implemented in real systems performing autonomous planning and execution workflows.

Example 3: CAMEL for Role-Based Agent Communication

The survey highlights CAMEL, a system where agents are assigned explicit roles (e.g., assistant and user) and communicate via structured dialogue. This role-based protocol improves coordination and task performance in real experiments such as problem-solving and code generation. The approach demonstrates how controlled communication protocols reduce ambiguity in multi-agent systems.

Example 4: AutoGen for Multi-Agent Collaboration in Real Applications

AutoGen is described as a framework where multiple LLM agents communicate through predefined conversation rules to collaboratively solve tasks. Agents specialize in planning, execution, or verification, and exchange messages to reach a final solution. This framework has been practically applied to software development, data processing, and tool-based workflows.

Example 5: ChatDev for Software Engineering via Agent Teams

The survey discusses ChatDev, a system that simulates a virtual software company composed of LLM agents with distinct roles. Agents communicate using structured protocols to design, implement, test, and debug software. This system demonstrates real-world applicability of agent communication in coordinated engineering tasks.

Example 6: ReAct-Style Agents with Explicit Thought–Action Communication

The paper reviews ReAct-based agents, which interleave reasoning traces with action commands through a structured communication format. These agents have been experimentally validated on real decision-making tasks that require tool usage and environment interaction. The explicit communication structure improves reliability and transparency during execution.

Example 7: Tool-Calling Agents Using Message-Based Protocols

The survey reports systems where agents communicate via structured messages to invoke external tools such as APIs, databases, or simulators. Each agent sends tool requests and receives structured responses, forming a closed-loop execution system. This protocol has been applied in real autonomous workflows including data retrieval and task automation.

Example 8: Blackboard-Based Multi-Agent Coordination

Some surveyed systems use a shared blackboard architecture, where agents post observations, plans, and results for other agents to consume. This communication mechanism supports parallel reasoning and coordination in complex tasks. The approach has been implemented in real experimental multi-agent environments requiring collaborative decision-making.

Example 9: Memory-Enabled Agent Communication for Long-Term Tasks

The paper discusses agents that communicate through shared memory structures, allowing persistence of task context over long horizons. These memory-enabled communication protocols enable agents to resume tasks, track progress, and adapt strategies. Such systems have been demonstrated in long-running autonomous agent experiments.

Example 10: Hierarchical Agent Communication for Scalable Systems

The survey presents hierarchical communication protocols where high-level planner agents communicate goals to lower-level executor agents. This structure has been tested in real multi-agent setups to improve scalability and task decomposition. It mirrors organizational structures used in industrial and robotic systems.

11. Enhancing reliability in LLM-integrated robotic systems A unified approach

Example 1: Reliability-Aware LLM Planning for Physical Robot Execution

The paper presents a robotic control pipeline where LLM-generated plans are explicitly checked for reliability before execution. Plans are validated against physical constraints, task feasibility, and execution history. This prevents unsafe or ill-defined actions from being sent to the robot, demonstrating a deployable safeguard for real robotic systems.

Example 2: Runtime Monitoring of LLM-Driven Robot Actions

A runtime monitoring module continuously observes robot execution and compares it with the intended LLM plan. If deviations occur—such as failed grasps or unexpected object motion—the system detects the inconsistency. This enables real-time intervention instead of silent failure, which is critical for real-world robotic deployment.

Example 3: Self-Verification of LLM Outputs Before Robot Control

The system integrates LLM self-verification, where the model evaluates its own generated plan before execution. The LLM checks for logical consistency, missing steps, and safety violations. This reduces hallucinated or incomplete instructions reaching the physical robot, improving execution reliability without human oversight.

Example 4: Failure Recovery Through Iterative Re-Planning

When execution fails, the robot does not stop permanently. Instead, execution feedback is returned to the LLM, which re-plans the task based on the observed failure. This closed-loop recovery mechanism is demonstrated on manipulation tasks, showing how reliability improves over repeated attempts.

Example 5: Separation of High-Level Reasoning and Low-Level Control

The paper emphasizes a layered architecture where the LLM is restricted to high-level reasoning only. Low-level motion control remains handled by deterministic robotic controllers. This separation ensures that probabilistic language models never directly command motors, aligning with industry safety requirements.

Example 6: Structured Action Representation to Prevent Ambiguity

Instead of free-form text, LLM outputs are constrained to structured action schemas. These schemas specify object references, action types, and execution parameters. This reduces ambiguity and makes plans interpretable by classical robotic systems, enabling reliable integration into real control pipelines.

Example 7: Consistency Checking Across Multi-Step Tasks

For long-horizon tasks, the system verifies that later steps remain consistent with earlier assumptions. If an object's state changes or becomes unavailable, the plan is flagged as invalid. This prevents cascading failures in multi-step robotic execution, which is a common issue in real deployments.

Example 8: Safety-Critical Task Filtering

The framework includes a mechanism to block unsafe or high-risk actions suggested by the LLM. Actions that violate safety policies or exceed predefined risk thresholds are automatically rejected. This shows how LLMs can be safely used even in safety-critical robotic applications.

Example 9: Explainability for Debugging and Trust

Each validated or rejected action is accompanied by an explanation generated by the system. Engineers can inspect why a plan was accepted or blocked. This explainability is essential for debugging real robotic systems and building trust in LLM-integrated pipelines.

Example 10: Unified Reliability Framework Across Tasks and Robots

The paper demonstrates that the same reliability mechanisms apply across different robotic tasks and platforms. This shows that reliability is treated as a system-level property, not a task-specific hack. Such unification is necessary for scalable industrial adoption of LLM-driven robotics.

12. InCoRo In-Context Learning for Robotics Control with Feedback Loops

Example 1: Rapid Generation of Executable Robot Behaviors

The paper presents LLM-BRAIn, a system where an LLM converts high-level task descriptions directly into executable robot behaviors. Instead of manually designing behavior trees or control scripts, the system automatically generates them. This significantly reduces development time for real robotic tasks while preserving compatibility with existing control frameworks.

Example 2: Behavior Tree Synthesis from Natural Language

A key contribution is the automatic generation of behavior trees from natural language instructions. The LLM translates task descriptions into structured control logic that robots can execute reliably. This approach is evaluated on real robotic task scenarios, demonstrating practical applicability beyond simulation.

Example 3: Fast Adaptation to New Tasks Without Reprogramming

The system allows robots to adapt to new tasks by simply changing the textual task description. The LLM regenerates the corresponding behavior structure without manual intervention. This supports real-world use cases where robots must frequently switch tasks, such as service or industrial environments.

Example 4: Integration with Classical Robotic Control Pipelines

LLM-BRAIn does not replace low-level controllers. Instead, generated behaviors are executed through existing robotic control modules. This design reflects real deployment constraints, ensuring reliability while still benefiting from LLM-driven flexibility.

Example 5: Reduction of Engineering Effort in Robot Programming

The paper shows that LLM-generated behaviors reduce the need for hand-crafted control logic. Engineers can focus on defining goals rather than coding behaviors. This directly addresses a major bottleneck in real robotic system development.

Example 6: Structured Output to Prevent Ambiguous Execution

The LLM outputs behaviors in a structured and constrained format, ensuring that generated behaviors remain interpretable and executable. This avoids the risks of free-form language generation and aligns with safety requirements in physical robots.

Example 7: Scalability Across Robotic Platforms

The approach is designed to be platform-agnostic. Generated behaviors can be reused across different robot embodiments as long as the underlying skills exist. This supports scalable deployment in heterogeneous robotic fleets.

Example 8: Human-Readable and Debuggable Behaviors

Generated behavior trees are human-readable, allowing engineers to inspect, debug, and modify them if needed. This transparency is essential for industrial adoption, where explainability and verification are required.

Example 9: Support for Long-Horizon Task Execution

The system can generate behaviors that include sequencing, conditions, and recovery actions. This enables robots to perform multi-step tasks rather than isolated actions. Such long-horizon execution is critical for real-world robotic workflows.

Example 10: Bridging Task-Level Reasoning and Execution

LLM-BRAIn acts as a bridge between abstract task descriptions and concrete robot execution. The LLM handles task-level reasoning, while the robot executes validated behaviors. This separation mirrors best practices in real robotic system design.

13. Large Language Models for Manufacturing

Example 1: Natural Language Interfaces for Shop-Floor Operations

The paper describes manufacturing systems where operators interact with machines and production systems using natural language. Instead of navigating complex HMIs or PLC interfaces, workers issue spoken or written commands interpreted by LLMs. This lowers the skill barrier on the shop floor while preserving existing industrial control pipelines.

Example 2: LLM-Assisted Process Planning in Manufacturing

LLMs are used to assist engineers in generating process plans, such as sequencing manufacturing steps or selecting appropriate tools. The models reason over historical production data, documentation, and constraints. These systems are evaluated in real manufacturing scenarios where planning speed and adaptability are critical.

Example 3: Automated Diagnosis of Production Line Faults

The paper presents LLM-driven systems that analyze sensor logs, alarms, and maintenance records to diagnose faults in production lines. Instead of manually inspecting logs, engineers receive natural language explanations of likely causes. This approach has been demonstrated in real industrial fault-diagnosis workflows.

Example 4: Predictive Maintenance Using Textual and Sensor Data

LLMs are integrated with structured sensor data and unstructured maintenance reports to predict equipment failures. The model correlates language-based maintenance history with operational signals. This hybrid reasoning enables earlier intervention and reduces downtime in real manufacturing environments.

Example 5: Knowledge Extraction from Manufacturing Documentation

Manufacturing facilities often rely on extensive manuals, SOPs, and legacy documentation. The paper shows how LLMs extract actionable knowledge from these documents and make it queryable. Engineers can ask questions such as maintenance procedures or safety constraints without searching through large document repositories.

Example 6: Human-in-the-Loop Decision Support for Engineers

Rather than replacing engineers, LLMs are deployed as decision-support agents. The system proposes recommendations for scheduling, quality control, or fault resolution, while humans retain final authority. This aligns with how AI is realistically adopted in safety-critical industrial settings.

Example 7: Quality Inspection and Defect Analysis

The paper discusses LLM-enabled analysis of inspection reports and defect descriptions. By aggregating text-based quality data, the system identifies recurring defect patterns and suggests corrective actions. This improves continuous improvement processes already used in manufacturing.

Example 8: Integration with Digital Twins and Manufacturing Systems

LLMs are combined with digital twin representations of factories to reason about system behavior and potential optimizations. The language model queries simulation outputs and production data to suggest improvements. This integration has been tested in industrial planning and optimization workflows.

Example 9: Supply Chain and Production Scheduling Assistance

The paper highlights LLM use in analyzing production schedules, supply constraints, and operational trade-offs. LLMs help planners understand complex scheduling conflicts and generate alternative plans. These systems are evaluated in real manufacturing planning scenarios where responsiveness is essential.

Example 10: Bridging Legacy Systems and Modern AI Pipelines

Many manufacturing environments rely on legacy systems not designed for AI integration. The paper shows how LLMs act as an abstraction layer, translating human intent into queries and actions across heterogeneous systems. This makes LLM adoption feasible without full system redesign.

14. LLM-BRAIn AI-driven Fast Generation of Robot

Example 1: Rapid Generation of Executable Robot Behaviors

The paper presents LLM-BRAIn, a system where an LLM converts high-level task descriptions directly into executable robot behaviors. Instead of manually designing behavior trees or control scripts, the system automatically generates them. This significantly reduces development time for real robotic tasks while preserving compatibility with existing control frameworks.

Example 2: Behavior Tree Synthesis from Natural Language

A key contribution is the automatic generation of behavior trees from natural language instructions. The LLM translates task descriptions into structured control logic that robots can execute reliably. This approach is evaluated on real robotic task scenarios, demonstrating practical applicability beyond simulation.

Example 3: Fast Adaptation to New Tasks Without Reprogramming

The system allows robots to adapt to new tasks by simply changing the textual task description. The LLM regenerates the corresponding behavior structure without manual intervention. This supports real-world use cases where robots must frequently switch tasks, such as service or industrial environments.

Example 4: Integration with Classical Robotic Control Pipelines

LLM-BRAIn does not replace low-level controllers. Instead, generated behaviors are executed through existing robotic control modules. This design reflects real deployment constraints, ensuring reliability while still benefiting from LLM-driven flexibility.

Example 5: Reduction of Engineering Effort in Robot Programming

The paper shows that LLM-generated behaviors reduce the need for hand-crafted control logic. Engineers can focus on defining goals rather than coding behaviors. This directly addresses a major bottleneck in real robotic system development.

Example 6: Structured Output to Prevent Ambiguous Execution

The LLM outputs behaviors in a structured and constrained format, ensuring that generated behaviors remain interpretable and executable. This avoids the risks of free-form language generation and aligns with safety requirements in physical robots.

Example 7: Scalability Across Robotic Platforms

The approach is designed to be platform-agnostic. Generated behaviors can be reused across different robot embodiments as long as the underlying skills exist. This supports scalable deployment in heterogeneous robotic fleets.

Example 8: Human-Readable and Debuggable Behaviors

Generated behavior trees are human-readable, allowing engineers to inspect, debug, and modify them if needed. This transparency is essential for industrial adoption, where explainability and verification are required.

Example 9: Support for Long-Horizon Task Execution

The system can generate behaviors that include sequencing, conditions, and recovery actions. This enables robots to perform multi-step tasks rather than isolated actions. Such long-horizon execution is critical for real-world robotic workflows.

Example 10: Bridging Task-Level Reasoning and Execution

LLM-BRAIn acts as a bridge between abstract task descriptions and concrete robot execution. The LLM handles task-level reasoning, while the robot executes validated behaviors. This separation mirrors best practices in real robotic system design.

15. LLM-Driven Robots Risk Enacting Discrimination, Violence

Example 1: Discriminatory Task Refusal in Human–Robot Interaction

The paper reports cases where LLM-driven robots respond differently to users based on perceived identity cues. Robots were more likely to refuse requests or alter tone when interacting with certain demographic groups. This behavior emerges from language model biases rather than explicit programming, demonstrating real discrimination risks in deployed systems.

Example 2: Normalization of Harmful Language Through Embodied Agents

When LLMs generate problematic language, its impact increases once embodied in a robot. The paper shows that harmful or aggressive responses become more socially salient when spoken or enacted by a physical robot. This amplifies harm compared to purely text-based systems.

Example 3: Escalation of Aggressive or Violent Instructions

The paper analyzes scenarios where ambiguous or provocative human instructions lead LLM-driven robots to generate unsafe interpretations. Even without direct commands to cause harm, the robot may suggest or attempt aggressive actions. This reveals how physical embodiment raises the stakes of misinterpretation.

Example 4: Reinforcement of Stereotypes Through Task Assignment

LLM-driven robots were observed assigning tasks differently based on stereotypical assumptions embedded in training data. For example, assumptions about physical capability or authority influenced robot responses. These behaviors reflect real societal biases encoded in language models.

Example 5: Lack of Contextual Moral Reasoning in Physical Environments

The paper highlights that LLMs often lack situational moral grounding when controlling robots. Without explicit constraints, robots may propose actions that are socially inappropriate or unsafe. This limitation becomes critical in real-world shared human environments.

Example 6: Over-Reliance on LLM Outputs by Human Operators

Human users tend to trust robotic systems more when responses appear fluent and confident. The paper shows that operators often follow unsafe or biased suggestions generated by LLMs. This creates a compounded risk where human judgment is overridden by perceived intelligence.

Example 7: Failure of Safety Filters After Embodiment

Text-level safety filters that work well in chat systems do not fully transfer to robotics contexts. The paper documents cases where filtered language still results in harmful physical actions. This exposes a gap between linguistic safety and embodied safety.

Example 8: Inconsistent Ethical Behavior Across Similar Scenarios

The same robot may respond ethically in one scenario and unethically in a slightly altered context. This inconsistency arises from prompt sensitivity rather than principled reasoning. Such unpredictability is unacceptable in real deployment settings.

Example 9: Absence of Accountability Mechanisms

When harmful behavior occurs, responsibility is unclear—whether it lies with the LLM, the system integrator, or the operator. The paper emphasizes that current robotic systems lack clear accountability structures. This complicates real-world deployment and regulation.

Example 10: Risks of Scaling LLM-Driven Robots Without Governance

The paper warns that scaling LLM-driven robots without robust governance frameworks will amplify societal harms. Once deployed at scale, biased or unsafe behaviors propagate rapidly. This makes early design decisions critically important for industry adoption.

16. Multi-Agent Systems for Robotic Autonomy with LLMs.pdf'

Example 1: End-to-End Robot System Design from Natural Language Task Descriptions

The paper demonstrates a system where a textual task description is the only input, and the framework automatically designs a complete robotic solution. The system determines how many robots are needed, where to place them, and how long their arms should be. This reflects a real-world engineering workflow where early design decisions are often manual and time-consuming.

Example 2: Task Analyst Agent for Engineering-Oriented Problem Decomposition

The Task Analyst agent extracts concrete engineering parameters—such as target coordinates, robot base locations, and task constraints—from long, unstructured task descriptions. This mirrors how human engineers translate customer requirements into technical specifications. The generated Task Analysis Report is used directly in subsequent robot design and control stages.

Example 3: Automatic Selection of Robot Arm Configurations

In industrial and medical scenarios (e.g., warehouse sorting or surgical tool handling), the Robot Designer agent selects appropriate arm lengths from multiple options. The agent avoids over-designed or under-capable robots by reasoning about reachability, cost, and safety. This reflects real design trade-offs encountered in industrial automation projects.

Example 4: Multi-Robot Task Allocation in Industrial Scenarios

For tasks involving multiple targets, such as warehouse item sorting or palletizing, the system automatically decides how many robots are required and assigns subtasks to each robot. This capability is evaluated across ten realistic scenarios, showing practical relevance for multi-robot industrial deployments.

Example 5: Automatic Generation of Reinforcement Learning Control Code

The Reinforcement Learning Designer agent generates executable RL code, including environment definitions, training scripts, and evaluation scripts. These scripts are executed to train robot controllers without human-written code. This demonstrates how LLMs can reduce the barrier to deploying RL-based control in practice.

Example 6: Autonomous Selection of RL Algorithms Based on Task Requirements

The system autonomously selects appropriate RL algorithms (e.g., PPO or SAC) depending on task complexity and motion requirements. This mirrors real engineering decisions normally made by experienced roboticists and is validated through executed training runs and trajectory outputs.

Example 7: Realistic Industrial and Medical Task Scenarios

The framework is evaluated on ten realistic task scenarios, including assembly line placement, warehouse sorting, elderly feeding assistance, and surgical instrument handling. These scenarios require different robot designs and control strategies, demonstrating generality beyond toy examples.

Example 8: Visualization of Executed Robot Trajectories and Motor Commands

After training, the system outputs learning curves, joint motor control plots, and end-effector trajectories. These outputs are standard artifacts used by robotics engineers to validate system behavior, indicating that the framework integrates naturally into existing engineering workflows.

Example 9: Ablation Studies Revealing Critical Engineering Dependencies

By disabling individual agents (Task Analyst, Robot Designer, or RL Designer), the authors show that removing any one component significantly degrades system performance. This reflects real-world system engineering, where missing analysis or design stages often leads to failure despite advanced algorithms.

Example 10: Model-Agnostic Deployment Across Different LLM Capabilities

The framework is tested with multiple LLMs of varying capability (GPT-4o-mini, GPT-4o, DeepSeek variants). Results show that stronger reasoning models produce more reliable robot designs and control code. This provides practical insight into how model choice affects real robotic system outcomes.

17.NRTrans.pdf

Example 1: Compiler-Verified Language-to-Robot Programs Before Execution

NRTrans translates natural language tasks such as *“grasp the bottle on the table”* into a Robot Skill Language (RSL) program, which is then compiled and verified before execution. Only programs that pass the compiler are sent to the robot. This ensures that no syntactically or semantically invalid control code is ever executed on hardware, a critical requirement for real deployments.

Example 2: Feedback-Based Error Correction Instead of Blind Execution

When an LLM generates an incorrect RSL program (e.g., missing semicolons or invalid parameters), the RSL compiler detects the error and produces human-readable feedback. This feedback is fed back into the LLM, which regenerates a corrected program. This loop runs automatically until the program is verified, preventing runtime failures on robots.

Example 3: High Success Rates Using Lightweight LLMs on Real Robots

NRTrans achieves up to 92% success rate using a 2B-parameter lightweight LLM, without any model retraining. This is demonstrated on real robotic task sets involving navigation, perception, and grasping. This shows feasibility for resource-constrained robots, where large foundation models cannot be deployed.

Example 4: Multi-Step Task Execution on a Real ROS-Based Robot

The framework is evaluated on a TIAGo mobile manipulator running ROS, executing multi-step commands such as *“approach the table, look for a cup, and grasp it.”* NRTrans generates a verified sequence of robot skills that execute correctly on hardware. This demonstrates real long-horizon task execution driven by language.

Example 5: Handling Ambiguous Natural Language Commands

Tasks like *“go forward a little”* or *“turn around several times”* are included in the evaluation set. NRTrans resolves ambiguity through feedback-based refinement, producing executable robot programs instead of failing silently. This reflects real-world user input, which is often imprecise.

Example 6: Safe Translation of Complex Sequential Instructions

NRTrans successfully translates complex commands such as *“move forward 2 meters, grasp the banana, turn left, then move forward 3 meters.”* These instructions are decomposed into RSL statements that are verified and executed sequentially. This demonstrates safe and correct handling of compound tasks.

Example 7: Preventing LLM Code Hallucination Through Language Abstraction

Instead of generating full Python control programs directly, NRTrans forces the LLM to generate high-level RSL commands only. This abstraction significantly reduces hallucinated code structures such as invalid loops or classes. The result is more reliable control program generation in practice.

Example 8: Semantic-Intuitive Error Messages for Fast Recovery

Compiler error messages are redesigned to be semantic and intuitive (e.g., “keywords should be lowercase” or “missing semicolon”). These messages enable LLMs to correct errors within 1–2 iterations on average. This rapid recovery is essential for real-time or near-real-time robotic applications.

Example 9: Zero-Shot Task Execution with Automatic Refinement

Even in zero-shot settings (no example programs provided), NRTrans increases the success rate of program generation by over 90% using feedback-based tuning. This allows robots to handle previously unseen tasks without manual prompt engineering or retraining, which is critical for scalable deployment.

Example 10: Portable Framework Across Robots and Scenarios

NRTrans abstracts robot capabilities into skill keywords, allowing the same framework to be adapted to different robots by updating the skill library and compiler bindings. This design supports scalability across platforms, rather than single-robot demos.

18.Plug in the Safety Chip Enforcing Constraints for LLM-driven Robot.pdf

Example 1: Safety-Constrained Execution on a Real Mobile Manipulation Robot

The Safety Chip system is deployed on a real mobile robot platform performing household-style mobile manipulation tasks. The robot executes natural-language instructions while continuously monitoring safety constraints. Unsafe actions are blocked at runtime, ensuring the robot never enters prohibited states during physical execution [Plug in the Safety Chip].

Example 2: Enforcing Human-Specified “Don’ts” in Natural Language

Users specify safety rules such as *“never pick up the phone”* or *“don’t go to the bedside table before the bookshelf”* in plain English. These rules are translated into verifiable Linear Temporal Logic (LTL) constraints and enforced during execution. The robot strictly adheres to these constraints while still completing the task [Plug in the Safety Chip].

Example 3: Runtime Action Pruning for LLM-Generated Plans

When an LLM agent proposes an unsafe action, the Safety Chip prunes the action before execution. The robot never physically executes the unsafe step. This mechanism works regardless of how the LLM generates plans, making it compatible with existing LLM-based robot agents [Plug in the Safety Chip].

Example 4: Closed-Loop Re-Planning with Safety Explanations

Instead of silently blocking actions, the system generates natural-language explanations describing why an action violates safety constraints. These explanations are fed back into the LLM to guide re-planning. This loop is demonstrated in both simulated environments and real robot experiments [Plug in the Safety Chip].

Example 5: Large-Scale Safety Evaluation in VirtualHome

The system is evaluated on 20 household tasks in the VirtualHome environment, each with up to five simultaneous safety constraints. The Safety Chip achieves near-perfect safety rates while maintaining high task success. This shows scalability to complex, real-world-like task specifications [Plug in the Safety Chip].

Example 6: Handling Increasing Constraint Complexity Without Failure

As the number of safety constraints increases, baseline LLM agents rapidly fail to maintain safety. In contrast, the Safety Chip maintains high safety rates even with five or more constraints, demonstrating robustness under realistic deployment conditions where multiple rules must be obeyed simultaneously [Plug in the Safety Chip].

Example 7: Deployment on a Boston Dynamics Spot Robot

The system is deployed on a Boston Dynamics Spot robot, performing mobile manipulation tasks in a real indoor environment with 23 objects and landmarks. The robot executes tasks while respecting up to ten safety constraints, demonstrating real hardware compatibility and practical feasibility [Plug in the Safety Chip].

Example 8: Maintaining 100% Safety on Real Hardware

On the Spot robot, the Safety Chip achieves 100% safety rate without reducing task completion success. In contrast, baseline approaches violate safety constraints under the same conditions. This result highlights the importance of formal safety enforcement for real deployments [Plug in the Safety Chip].

Example 9: Detecting and Aborting Contradictory Instructions

When users provide instructions that inherently conflict with safety constraints, the Safety Chip detects the contradiction and aborts execution safely. Baseline systems attempt to execute the task anyway, leading to unsafe behavior. This capability is essential for real-world human-robot interaction [Plug in the Safety Chip].

Example 10: Modular “Safety Chip” Compatible with Any LLM Agent

The Safety Chip is designed as a plug-and-play module that can be attached to any existing LLM-based robot agent. It does not depend on a specific planning architecture or training regime. This modularity reflects how safety mechanisms must be deployed in real industrial systems [Plug in the Safety Chip].

19.Reshaping Robot Trajectories Using Natural Language Commands.pdf'

Example 1: Natural Language Correction of Robot Arm Trajectories

The paper presents a system where users reshape an existing robot arm trajectory using spoken commands such as *“stay away from the wine glass.”* Instead of reprogramming or replanning from scratch, the robot modifies its motion path while keeping the same start and goal. This behavior is executed on a real robot arm, demonstrating direct language-to-motion influence.

Example 2: Factory-Safety Scenario with Fragile Objects

A realistic factory-like scenario is demonstrated where a robot transports a wine bottle toward a shaker while passing close to fragile crystal glasses. The original trajectory is collision-free but unsafe. Using natural language, users instruct the robot to move further away from the glasses, and the reshaped trajectory is executed safely on real hardware.

Example 3: Real-Time Trajectory Reshaping on a Franka Panda Robot

The system is evaluated on a 7-DOF Franka Emika Panda arm operating on a physical tabletop workspace. The reshaped trajectories generated from language commands are executed directly by the robot, preserving kinematic constraints and smooth motion. This confirms real-world feasibility beyond simulation.

Example 4: Replacing Kinesthetic Teaching with Language-Based Control

The paper compares natural language trajectory reshaping with kinesthetic teaching, trajectory drawing, and manual cost-function tuning. In real robot experiments, users achieve correct and safe trajectories faster using language than by physically guiding the robot. This highlights practical advantages for industrial and service robotics.

Example 5: Robust Handling of Natural and Unseen Vocabulary

Users issue commands with unseen words and synonyms such as *“stay a bit further away,” “pass slightly closer,”* or *“drive away from the glass.”* Despite not seeing these phrases during training, the robot correctly reshapes its trajectory. This robustness is achieved through pretrained language models and validated in executed motions.

Example 6: Fine-Grained Control Using Linguistic Intensity Modifiers

The system interprets intensity modifiers like *“a little,” “very,”* or *“much”* to adjust how strongly the trajectory is modified. For example, *“stay very far from the object”* results in a noticeably larger deviation than *“stay a bit away.”* These graded effects are clearly visible in real robot motion.

Example 7: Preserving Start and Goal Constraints During Modification

In all experiments, the robot maintains the original start and end positions while modifying only the intermediate path. This ensures task completion is preserved while improving safety or semantics. Such constraint-aware behavior is essential for deployment in production environments.

Example 8: User Study Showing Preference for Language Interfaces

A controlled user study with real robot experiments shows that participants strongly prefer natural language over programming, drawing, or kinesthetic teaching. Language-based control achieves a 100% success rate, faster completion times, and fewer failures due to kinematic infeasibility. This provides human-centered validation of real usability.

Example 9: Language as a High-Level Safety Layer Over Classical Planners

The system operates on top of classical motion planners such as A* and CHOMP. Language does not replace planning; instead, it reshapes planned trajectories to satisfy semantic or safety constraints. This layered design mirrors how robotics systems are deployed in real applications.

Example 10: Practical Human–Robot Collaboration Without Reprogramming

The paper demonstrates that users can iteratively refine robot motion through conversation-like commands without stopping execution or modifying code. This enables non-expert workers to safely adjust robot behavior in shared workspaces, reflecting a realistic path toward collaborative robotics.

20.Understanding Natural Language Commands.pdf'

Example 1: Voice-Controlled Robotic Forklift in a Warehouse Environment

The paper demonstrates a robotic forklift that executes spoken natural language commands in a warehouse-like environment. Commands such as *“Put the tire pallet on the truck”* are translated into navigation and manipulation actions. The forklift physically moves, picks up pallets, and places them on target locations, showing a real mobile manipulation system driven by language.

Example 2: Executing Multi-Step Commands with Navigation and Manipulation

The system handles commands that combine navigation and manipulation, such as *“Go to the first crate on the left and pick it up.”* The robot first navigates to the correct location, then performs the manipulation action. This demonstrates real-world execution of compound, multi-stage tasks rather than isolated actions.

Example 3: Grounding Object References in a Physical Environment

When users say *“the tire pallet”*, *“the truck”*, or *“the pallet on the trailer”*, the system grounds these phrases to actual physical objects using a semantic map of the environment. The robot correctly distinguishes between multiple similar objects, a critical requirement for real industrial deployment.

Example 4: Correct Interpretation of Spatial Relations

The system successfully interprets spatial relations such as *“on”*, *“next to”*, *“to the left of”*, and *“in front of.”* For example, it distinguishes between *“put the pallet on the truck”* and *“go to the pallet on the truck,”* generating different action plans. This capability is demonstrated in executed forklift behaviors, not just parsing accuracy.

Example 5: Learning Verb Meanings from Human-Generated Commands

Rather than hand-coding actions, the model learns the meanings of verbs like “put,” “take,” and “pick up” from a corpus of human-written commands. These learned meanings are then used to control the robot’s actions. This reflects a scalable approach where robots adapt to human language rather than forcing humans to adapt to robots.

Example 6: Handling Noisy and Untrained Human Language

Commands are collected from untrained users via crowdsourcing and include spelling errors, ambiguity, and unconventional phrasing. Despite this, the robot successfully executes many of these commands. This shows robustness to real-world human input, which is essential for deployment outside controlled lab settings.

Example 7: End-to-End Evaluation of Executed Robot Behavior

The system is evaluated end-to-end by showing videos of the robot’s executed actions to human evaluators, who judge whether the robot followed the command correctly. High agreement scores confirm that the robot’s physical behavior matches human intent. This evaluation focuses on task success, not just language accuracy.

Example 8: Partial Success and Graceful Failure in Real Tasks

Even when the system makes mistakes—such as picking the wrong pallet—it often performs a partially correct action (e.g., correct motion but wrong object). This behavior is characteristic of real deployed systems and demonstrates meaningful grounding rather than random failure.

Example 9: Implicit Support for Clarification and Dialogue

Because the system produces confidence scores for each grounded component of a command, it can identify uncertain interpretations. The authors explicitly discuss using this information to ask clarifying questions before acting. This anticipates interactive human–robot dialogue in real deployments.

Example 10: Voice-Commandable Robots Operating Alongside Humans

The robotic forklift is designed to operate alongside humans in minimally prepared environments, rather than fenced-off industrial cells. This positions the system as an early example of language-driven collaborative robotics in real-world settings.
