2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Fly by Book: How to Train a Humanoid Robot to Fly an Airplane using Large Language Models

Hyungjoo Kim, Sungjae Min, Gyuree Kang, Jihyeok Kim and David Hyunchul Shim†

***Abstract*— A pilot needs to manipulate various gadgets in the cockpit based on vast knowledge of rules and procedures while verbally communicating with air traffic controllers. While precision manipulation in the cockpit during the flight is already a difficult task, a far more difficult thing is how to make a robot learn all the knowledge needed to fly an airplane in accordance with all the rules and regulations. As a pioneering effort, this paper introduces LLM-PIBOT, which leverages the latest advances in Large Language Models (LLMs) to empower a humanoid pilot robot (PIBOT) to take the full authority of an airplane by understanding and executing complex procedures outlined in Pilot's Operating Handbooks (POHs). Unlike traditional rule-based methods, LLM-PIBOT system infers suitable flight procedures, employs an embedding process to accurately identify relevant procedures within documents, and structures the text-extracted flight tasks into tuples using our carefully crafted prompts. This approach enables PIBOT to adapt to the given POHs, generating and executing task plans in real-time in response to commands and situations. Experimental results show that LLM-PIBOT can comprehend and follow the complex procedures specified in the manuals and to fly the airplane on a full-scale simulator using the generated flight plans.**

I. INTRODUCTION

It has been a long standing dream to create a robot in human form that can be delegated to perform any given tasks ranging from daily chores to sophisticated tasks that require expertise. Humanoid robots are considered to be the ideal form for such applications: after decades of research, now they can not only walk, run and jump, but also carry boxes and operate complex coffee machines.

One relatively less investigated application for humanoid robot is the sedentary tasks where humans (robots) sit at a console and perform tasks by manipulating various controls on the console. Tasking humanoid robots to perform such works has many merits: existing devices and systems originally designed for humans can be readily automated without any modification, hence reducing time and effort. An interesting case in that category is piloting an aircraft, where a pilot (robot [1]) sits in a cockpit and manipulate the various control devices in real time based on the complex knowledge of flight manuals and numerous procedures. When building such robots, there are three major topics to address: perception, precision manipulation, and implementing the task intelligence. While the two former topics have been relatively well researched and resolved to certain degrees in recent years, the last have remained unresolved: programming the complex rules and handle all the cases that are encountered during flights is virtually impossible.

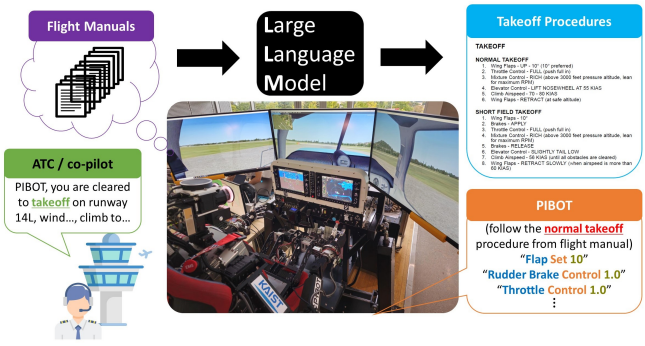

Fig. 1. The proposed LLM-PIBOT utilizes the strengths of LLM, enabling PIBOT to comprehend flight procedures based on POHs and generate corresponding task plans. This approach demonstrates the potential for generalization across extensive POHs and presents the feasibility of integrating natural language capabilities with humanoid robot.

However, the landscape of artificial intelligence offers promising solutions to these challenges, particularly with the advent of LLMs. Recent advancements in LLMs have significantly transformed the capabilities of natural language processing and understanding [2], [3]. LLMs utilize vast amounts of text data to learn complex language patterns. This enables them to generate coherent and contextually relevant text, comprehend intricate instructions, and engage in meaningful dialogue. Such breakthroughs not only enhance a robot's ability to understand and interact using human language but also pave the way for automation and intelligent system design across various domains, including aviation. The integration of LLMs into systems as PIBOT represents a pivotal step towards achieving a shift from rule-based approaches to more flexible, understanding-based interactions.

In this paper, we introduce LLM-PIBOT, a novel approach that enables PIBOT to understand various flight commands and POHs. LLM-PIBOT utilizes our designed prompts to infer the most appropriate flight procedures for given commands. The inferred procedures are accurately matched through an embedding process from POHs. Relevant procedures are searched, and the flight tasks within these procedures are extracted and text-rendered. The extracted flight tasks are structured into tuples by accepting prompts based on knowledge of cockpit function types,

This research was supported by the Challengeable Future Defense Technology Research and Development Program (No.915102201) of Agency for Defense Development in 2024.

All authors are with The School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, 34131 Rep. of Korea. {hyungjoo_kim, sungjae_min, fingb20, jihyeokkim, geninfty}@kaist.ac.kr
† Corresponding Author.

thereby simplifying and clarifying the task planning process. Moreover, prompts for flight control are provided by LLM-PIBOT, enabling the execution of control tasks during flight operations to be conducted in real-time. Applying this methodology to PIBOT, the performance is evaluated based on the appropriateness of flight task planning for the received commands and the completion of these tasks. The capability of the LLM-PIBOT system to adapt to POHs of various aircraft types is also assessed. This system is anticipated to significantly contribute to the development of a fully autonomous flight system for PIBOT based on POHs, by intepreting and inferring proper flight procedures and tasks for any given commands.

II. RELATED WORKS

A. Prompt Designed with LLM

The task of hierarchically and manually programming extensive volumes of predefined POHs is notably inefficient. To address this issue, recent research has actively engaged in developing prompts for LLMs to convert these into robot task plans [4]. "Prompt-based learning" has emerged as a new paradigm in NLP [5], [6]. In this approach, language models directly model text probabilities and derive final outputs through modified input prompts. This method facilitates few-shot or zero-shot learning on datasets with minimal or no labels [7], [8]. It achieves this by pre-training language models on vast amounts of raw text and leveraging new prompt functions.

SayCan [9] proposed prompt to break down high-level instructions into sequences of low-level skills. The prompt was designed to include examples within themselves, guiding language model to mimic specific response structures. However, this alone proved insufficient to fully restrict outputs to tasks executable by actual agents. Code as Policies (CaP) [10] highlighted the importance of designing prompts free from ambiguities and bugs. They validated that the use of specific function and variable names could elicit reliable and consistent results from LLM. ProgPrompt [11] was proposed that centered on the programming structure of robot tasks. It utilized a combination of prompt and python code to integrate programming structures and logic, thereby offering sophisticated task plans. LLM-based Incremental Learning [12] proposes a dynamic and incremental prompt design to enable robots to learn improved behaviors from user interactions, utilizing a specific function. It incorporates examples within prompts that include user instructions for improving situations previously failed. This method allows robots to learn from mistakes, equipping them to avoid the same errors in future situations. LLM-GROP [13] focused on the development of prompts that enable robots to formulate symbolic and geometric spatial relationships among objects. These methods are aimed at providing robots with common-sense knowledge, which is instrumental in the development of task plans and the execution of missions.

Similarly, our proposed PIBOTPROMPT provides basic and simplified aviation knowledge, such as lists of flight procedures according to flight situations, and cockpit function types. This approach enhances the contextual understanding of commands entered into the LLM based on the prompt and facilitates the extraction of keywords. Our work improves consistency and accuracy by utilizing predefined aviation-standard POHs, which provide clear reference materials and guidelines. In contrast to the aforementioned studies, which predict and plan subsequent tasks, our approach generates reliable task plans by transforming POHs into the desired format. Furthermore, the proposed prompt simplifies the flight procedures in these manuals, ensuring comprehension across various types of aircraft manuals. This method is capable of corresponding to a wide array of POHs for different aircraft types, enabling PIBOT to interpret natural language commands and respond appropriately.

B. Task Planning with LLM

The recent advancements in various LLMs [14]–[16] have significantly propelled experiments in task planning based on robotic operations. Inferences about subsequent tasks and the feasibility of actions are provided by these LLMs through the application of common-sense knowledge in prompts based on zero-shot learning [17]–[19]. Integrated with robots, these models are being used to validate the performance of task and motion planning [13], [20], [21]. LLM-POP [22] proposes a method for fine-tuning LLMs for interactive planning in robotic tasks. It utilizes situational examples to guide LLMs in generating structured outputs, which comprise tasks, instructions, explanations, and actions. The application of structured outputs for fine-tuning and optimization of LLM in partially observable robotic environments enhances performance and aids in the generation of executable plans. Similarly, ProgPrompt [11] structures task planning into tuples comprising objects, their properties, executable actions, transition models, and initial and goal states. This approach facilitates the extraction of keywords that aid the robot in clearly understanding and accurately executing tasks.

Our work also structures task planning into tuples based on PIBOTPROMPT information for commands such as flight procedures and controls. However, the tuple structure proposed by LLM-PIBOT is simplified and is directly transmitted to PIBOT for action implementation. It takes into account communication, including the current situation and status reporting, as well as clear mission execution based on the flight procedures. This approach facilitates effective task planning across various scenarios related to aircraft operations.

III. LLM-PIBOT APPROACH

We propose LLM-PIBOT, which enables PIBOT to generate task plans based on POHs for operating aircraft in compliance with aviation regulations. A method is proposed for designing prompts to enable PIBOT to comprehend POHs for various types of aircraft and to establish task plans based on inferred flight procedures. The proposed prompts are composed of three parts, each serving a distinct purpose.

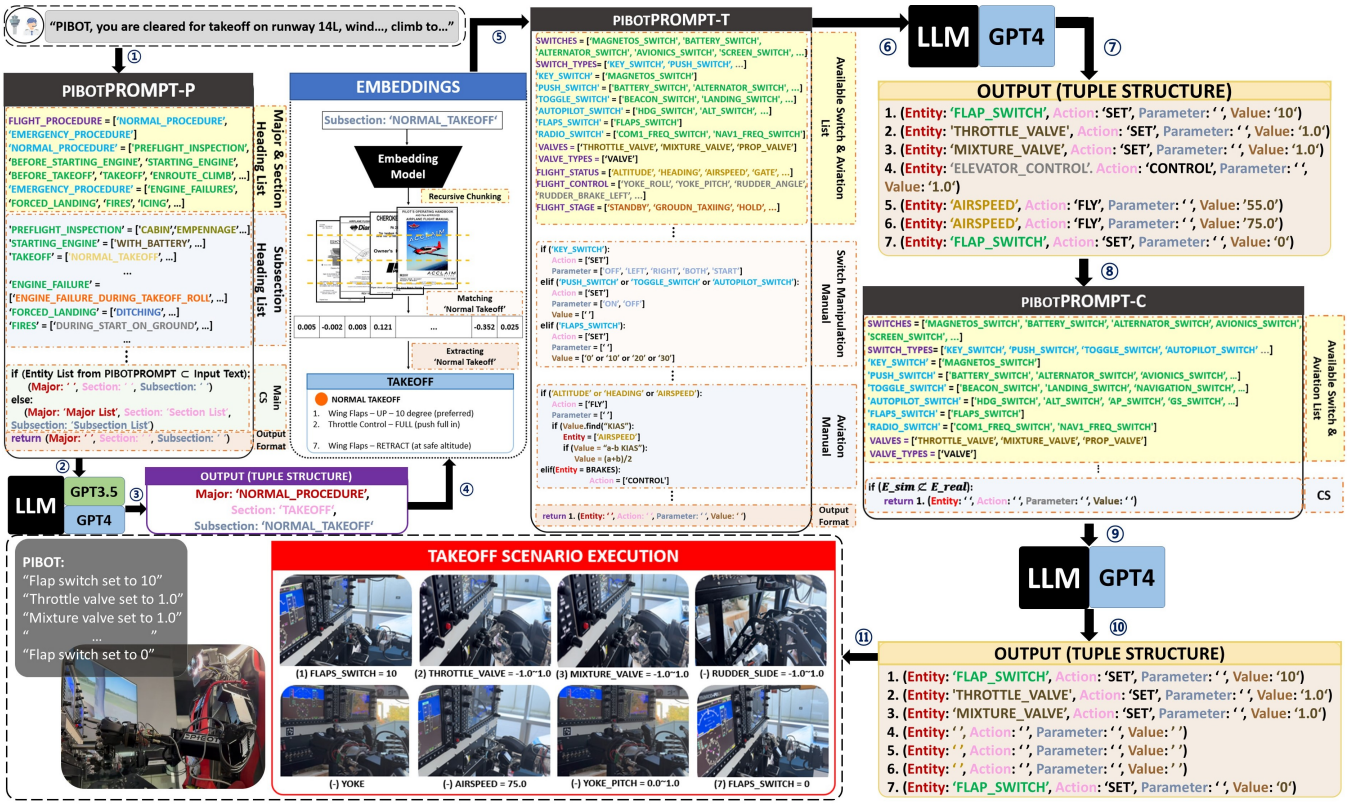

Fig. 2. The complete execution process of LLM-PIBOT is represented. In the top left, the input command illustrates a scenario of takeoff permission. The designed PIBOTPROMPT-P, PIBOTPROMPT-T, and PIBOTPROMPT-C are responsible for inferring flight procedures based on commands, generating task plans, and structuring tuples. Related flight tasks for the procedure are extracted as text through embedding using POHs in PDF format. The bottom right section shows PIBOT sequentially informing and executing the process based on the generated task plans.

A. Communication with ATC & Copilot

The methodology by which PIBOT communicates with ATC & copilots and performs 'read-back' of the generated task plans is comprehensively described. This encompasses Flow 1 and 11 of the system operation as presented in Fig. 2. In Flow 1, speech recognition is applied to transform human speech into text. The transformed text is then conveyed to **PIBOTPROMPT-P**rocedure (P-P), enabling LLM to understand and infer the context of commands. Flow 11 involves the role of 'read-back', where PIBOT performs the tasks received while also reporting to the ATC or copilots. This highlights the significance of clear communication, error detection during task execution, and mutual verification in terms of aviation safety procedures. The final outputs generated via **PIBOTPROMPT-T**ask (P-T) and **PIBOTPROMPT-C**ondition (P-C) are structured in tuples, each comprising four components: 'Entity', 'Action', 'Parameter', and 'Value'. The information derived from these tuple structures is sequentially interpreted through TTS. For instance, (Entity: 'BATTERY SWITCH', Action: 'SET', Parameter: 'ON', Value: ' ') is interpreted as "battery switch set on", involving direct verbalization and the action of turning on the device (battery switch). This structured data approach plays a crucial role in facilitating clear directive transmission and interpretation, thus minimizing potential errors during task execution.

B. PIBOTPROMPT-P

POHs are broadly classified into normal (N) and emergency (Ξ) procedures. These procedures are further divided into major (M), section (S), and subsection (s) headings, each defined in detail according to the flight situation. The purpose of this prompt is to encourage the accurate and specific output of appropriate flight procedures for the current flight situation based on the received command. This prompt is categorized into four main areas. Firstly, it lists major and section headings. Secondly, it enumerates each procedure comprised of subsection headings. Thirdly, it includes conditional statements (CS) for generating accurate flight procedures. Finally, it defines the format of the final output. To clearly present flight procedures, the prompt specifically categorizes them into normal $(\nu_1, ..., \nu_n) \subset N$ and emergency $(\xi_1, ..., \xi_n) \subset \Xi$ procedures, and examples are provided for structuring them in the tuple format <**M**, **S**, **s**>. This format enables the LLM to provide the most appropriate flight procedures through prompt-based logical reasoning, evaluating candidates generated based on the input. This ability demonstrates the LLM's capacity to discern the essence of a command's meaning. The inferred result values are output in a tuple format. For instance, it outputs <**M:** 'NORMAL_PROCEDURE', **S:** 'TAKEOFF', **s:** 'NORMAL_TAKEOFF'> for a command "Cleared to takeoff, wind ..., climb ...". However, this signifies that simple

Fig. 3. Illustrates the systematic organization and definition of the simulator's operable features by entity within the PIBOTPROMPT.

flight control commands could be misinterpreted by the LLM as being related to flight procedures. For examples, a flight control command "Control airspeed to 75 KIAS" might prompt the provision of a cruising procedure from N. This occurs because LLM is focused on flight procedure knowledge based on this prompt. To address this issue, we apply CS, which incorporate a list of entities containing operational mechanical components and variables required for flight control within the prompt. It is defined that the tuple value should be null if a flight control command including these entities is entered. In other words, a null tuple signifies the need to directly generate task plans for flight control, irrespective of flight procedures derived from the flight manual.

C. PIBOTPROMPT-T

P-T encourages the generation of task plans by understanding flight procedures defined in different POHs, regardless of the type of aircraft. This prompt is categorized into four main areas: The first encompasses a list of entities, including operable mechanical components installed in the aircraft, variables related to flight control, and aviation stages; the second pertains to the knowledge required for operating switches based on their types; the third involves the basic aviation knowledge necessary for PIBOT to execute controls according to the manual; and finally, the format of the final output. To present the task plans clearly and concisely, we provide examples of structuring them in a tuple format $<\mathbf{E},\mathbf{A},\mathbf{P},\Upsilon>$. In this tuple, E represents the set of entities as shown in Fig. 3, comprising elements such as switches (E_s), valves (E_v), flight control (E_{fc}), flight status (E_{fs}), aviation stage (E_{as}). E_s is categorized by switch features, including S_{push}, S_{toggle}, S_{key}, $S_{autopilot}$, S_{radio}, and S_{flap}. A_i denotes a set of executable actions based on E, where $i = \{set, control, fly\}$. E_s and E_v are defined as 'SET', E_{fc} as 'CONTROL', and E_{fs} as 'FLY'. P represents the desired parameters based on E and A, while Υ signifies the target values according to E and A. Hence, $\{S_{push}, S_{toggle}, S_{autopilot}\} \in E_s(A_{set}, P)$ can be distinctly articulated as 'ON/OFF', whereas $\{S_{radio}, S_{flap}\} \in E_s(A_{set}, \Upsilon)$ necessitate quantified values. Similarly, $\{V_{throttle}, V_{propeller}, V_{mixture}\} \in E_v(A_{set}, \Upsilon)$, $\{FC_{yoke\_roll}, FC_{yoke\_pitch}, FC_{rudder\_slide}, ...\} \in E_{fc}(A_{control}, \Upsilon)$, and $\{FS_{standby}, FS_{taxiing}, ...\} \in E_{fs}(A_{fly}, P)$ can be defined. The knowledge required for manipulating these entities is defined as follows: S_{push}, S_{toggle}, and $S_{autopilot}$ have 'ON' and 'OFF' states, S_{key} includes 'OFF', 'RIGHT', 'LEFT', 'BOTH', 'START' positions, and S_{flap} ranges from '0', '10', '20', to '30'. S_{radio} is adjusted according to frequency values, and functions pertaining to E_v and E_{fc} operate between -1.0 and 1.0. For instance, the prompt provides information that E includes the battery switch and classifies its feature as a push switch when the command "PIBOT, please turn the battery switch on" is given. This leads the LLM to infer the sequences as '*battery switch*' $\rightarrow$ S_{push} $\rightarrow$ $E_s(A_{set}, P(=$'*ON*'$))$, thus generating the final output as <**E:** 'BATTERY_SWITCH', **A:** 'SET', **P:** 'ON', Υ**:** ->. This provision of knowledge prevents errors in the final tuples and aids in adapting to various POHs. For example, a flight task such as 'Throttle Control - FULL' lead the LLM to consider 'FULL' as P. However, by incorporating operational knowledge into the prompt 'FULL' is inferred as 1.0 which leads to the generation of the tuple as Υ, thus enabling PIBOT's action execution.

D. PIBOTPROMPT-C

Considering the differences between actual aircraft and flight motion simulator cockpit, we present effective application methods for POH-based task (flight procedures) planning. While the list of entities specified in the POHs (E_{real}) and installed in the simulator (E_{sim}) is similar, the existence of some uninstalled components in the simulator restricts PIBOT's task performance. To address this, the final output is restricted to being null if the output based on E_{real} is not included in the set of entities provided in P-C. Null tuple results are not transmitted to PIBOT, allowing the continuation of subsequent flight tasks.

E. LLM-PIBOT

The system employs embedding techniques using the subsection title (s) output in tuple format from Flow 4. POHs in PDF format are loaded, and recursive chunking is utilized to segment the manuals composed of multiple pages. The inferred 's' is matched with the segmented data to identify the most similar subsection titles, from which related flight tasks are extracted. The structured text resulting from this extraction is then input into P-T. Flight task plans in tuple format output from P-T and P-C are executed by PIBOT. However, given the critical importance of safe aircraft operation, a feedback correction loop was added to ensure that the final flight tasks output by the P-C are executed accurately in accordance with the flight manual. This feedback loop includes a procedure where the copilot verifies that the generated tasks align with the manual after they have been produced. Once verification is complete, a confirmation signal is sent via the microphone, which is then

TABLE I

THE SPECIFICATIONS OF THE MODELS EMPLOYED IN LLM-PIBOT.

Model		Name
Speech-to-Text (STT)		Whisper (v2-large)
Text-to-Speech (TTS)		GoogleTTS
LLM	GPT3.5	gpt-3.5-turbo-1106
	GPT4	gpt-4-0313
Embedding	LangChain with OpenAI	text-embedding-3-large
POH	1	Mooney-20V
	2	Cessna-172S
	3	DA40-180
	4	Cirrus-SR22
Flight Motion Simulator		C172 Motion
Flight Simulation		Prepar3D v3

TABLE II

FLIGHT PROCEDURE CLASSIFICATION AND SYSTEM EVALUATION CRITERIA ACCORDING TO TASK 1-9.

Task	ID	Required Procedure
Normal Procedure (N)	1	Starting Engine
	2	Normal Takeoff
	3	Securing Airplane
Emergency Procedure (Ξ)	4	Engine Failure during Takeoff Roll
	5	Wing Fire
	6	Fire during Start on Ground if Engine Fails
Aviation (Flight Control)	7	Heading
	8	Altitude
	9	Own Navigation

relayed to PIBOT to execute the flight tasks. On the other hand, the command is resent for reprocessing if the generated tasks do not match the manual. This mechanism promotes the autonomous generation of flight task plans, ensuring that PIBOT adheres to aviation regulations and safely executes real-time commands with clarity.

IV. EXPERIMENTS

The performance of LLM-PIBOT is evaluated, and the capability of PIBOT to execute generated task plans while seated in a flight motion simulator is also verified.

A. Experiment Setup

For accurate command transcription, the validated Whisper Large model [23] is employed to convert received missions into text. Google TTS [24] is utilized for performing PIBOT's read-back. The LLM models used in the experiment are OpenAI's GPT-3.5 [14] and GPT-4 [15] engines. The use of these two models is aimed at the experimental evaluation of P-P, focusing on the inferential capabilities of the LLM based on the designed prompts. A model engine demonstrating superior performance is employed in P-T to enhance the accuracy of the generated task plans. Subsequently, the loaded POHs are segmented using LangChain TextSplitter , and the text is transformed into vectors using OpenAI's text-embedding-3-large model [25]. The performance of the designed prompts is evaluated using PDF-format POHs for the four most popular aircraft types listed in Table I. To execute task plans for the generated flight procedures, a flight motion simulator (C172 Motion) depicting the Cessna 172 with G1000 model is used. Lockheed Martin's Prepar3D flight simulation software is utilized to verify performance by executing the tasks through PIBOT in real-time.

B. Real-world Experiment

The experiment sets a flight scenario that starts on runway 14L at Seoul Gimpo Airport (RKSS). PIBOT is assigned 9 different tasks as illustrated in Table II. Task 1-3 are categorized as N, task 4-6 as Ξ, and task 7-9 as flight control procedures. The performance of this system is evaluated based on the required command inputs for each procedure specified in task 1-9. This approach provides PIBOT with the flexibility to adapt to tasks in real-time and modify flight conditions as desired. Task 1-2 encompass the stages from engine startup to takeoff. Task 3 represents the action of turning off the aircraft engine after landing. Task 4-6 assume emergency situations during flight and assign related commands. Task 7-9 perform flight control commands during cruising, turning, and descending aviation stages. The commands involve applying different heading and altitude values from the current aircraft state to verify PIBOT's task performance. Subsequently, a command to return to the original waypoint is issued upon deviation from the set flight scenario. Therefore, task 1-6 involve assigning commands while the aircraft is positioned on the runway, and task 7-9 entail assigning commands when the aircraft is in the air. The experiment is conducted repeatedly for four types of aircraft, applying POHs relevant to each type.

C. Evaluation Metrics

We have established five metrics to evaluate the performance of the LLM-PIBOT system: Inferred Procedures (IP), Embedding Results (ER), Generated Task Plans (GTP), Conditional Statement (CS), and Real Executions (RE). Our evaluation utilizes the tasks of each flight procedure specified in the POHs as ground truth.

- IP categorizes commands into two types. Commands with subsection titles of flight procedures are labeled as 'Explicit'. Those implying the meaning without mentioning subsection titles are termed 'Implicit'.
- ER evaluates the embedding outcomes based on the IP result, from a given POH (PDF file). ER = 1 when the specific part of the procedure is accurately extracted from the manual; otherwise, ER = 0.
- GTP is calculated as the ratio of the total number of task plans outputted in tuple format based on ER to the ground truth. GTP = 1 indicates a match in the content and number of flight tasks.
- CS quantifies the conversion frequency of entities present in E_{real} but absent in E_{sim} into null tuples, facilitated by the application of conditional statements. It compares the set of cockpit function entities in the simulator E_{sim} with that of the actual aircraft E_{real}, where $E_{sim} \not\subset E_{real}$.
- RE represents the success rate of actual tasks performed by PIBOT in the simulator, derived from the flight task plans generated based on GTP and CS. Success

in execution is considered when the corresponding functionalities for each task operate within the flight simulation.

These metrics offer a comprehensive framework for evaluating the LLM-PIBOT system's performance. They highlight the system's capability to infer, process, and execute flight tasks as directed by PIBOT, based on the flight command.

V. RESULTS

Each prompt designed within our LLM-PIBOT system successfully guides LLM-based flight task planners. This enables PIBOT to autonomously conduct flights based on POHs.

A. Inferred Flight Procedure

Table III presents the experimental results of prompt performance within the LLM-PIBOT system utilizing P-P across multiple GPT models. The focus was on two input command types: those using direct flight procedure terms and those indirectly implying meaning without the explicit use of terms. The aim was to assess how the LLM infers flight procedures from given commands and to examine the impact of prompt presence on each GPT model's performance.

TABLE III
COMPARATIVE ANALYSIS OF FLIGHT PROCEDURE INFERENCE PERFORMANCE BY COMMAND (EXPLICIT VS. IMPLICIT MEANING).

Methods		Inferred Procedure (Task #ID)								
		1	2	3	4	5	6	7	8	9
Explicit	GPT-3.5	O	X	O	O	O	X	X	X	X
	with P-P	O	O	O	O	O	O	O	O	X
	GPT-4	O	O	O	O	X	X	X	O	X
	with P-P	O	O	O	O	O	O	O	O	O
Implicit	GPT-3.5	X	X	X	X	X	X	X	X	X
	with P-P	O	O	O	O	O	X	X	X	X
	GPT-4	X	O	X	X	X	X	X	X	X
	with P-P	O	O	O	O	O	O	O	O	O

For Task 2, which required N, commands were categorized into explicit commands, such as "PIBOT, cleared for takeoff at runway 14L," and implicit commands, such as "PIBOT, winds are calm, cleared for departure on runway 14L." Task 4, addressing Ξ, used "Mayday, follow the engine failure during takeoff roll procedure" and "Mayday, aborting takeoff, engine failure" as examples of explicit and implicit commands, respectively. Therefore, commands for tasks 1-9 were applied to the models to derive various conclusions.

The baseline LLM faced significant limitations in inferring flight procedures. This was chiefly due to the evaluation metric IP, which required that the outcomes of the inferences align with the terminology for flight procedures as stipulated in the POH's table of contents. To ameliorate this, the introduction of P-P provides LLMs with detailed classifications for main, section, and subsection titles of N and Ξ listed in POHs. This approach narrows the focus of the inference process to data provided by P-P. Furthermore, the use of LLM with P-P has been demonstrated to excel in inferring flight procedures through explicit and implicit commands. However, a degradation in performance was observed in tasks 7-9, which involved inferring flight control. For instance, implicit commands for tasks 7-9 such as "Maintaining direction/level at a certain angle/height" and "Returning to the original waypoint" were intended to be recognized as flight control procedures but were instead inferred as N such as v_{climb}, v_{cruise}, and $v_{descent}$. The GPT-4 with P-P model has been successfully utilized to integrate CS and accurately infer flight procedures and controls across all tasks. This indicates that the model's ability to enhance natural language guidance and leverage data within prompts is particularly outstanding. Therefore, GPT-4 plays a pivotal role in enhancing the accuracy of inferring flight procedures and controls from input commands through the LLM-PIBOT system.

ENGINE FIRE – DURING START ON GROUND
Magneto/Starter Switch .....CONTINUE cranking or until fire is extinguished.

If engine starts:
Power .....1500 RPM for several minutes
Engine .....SHUTDOWN: Inspect for damage
If engine does not start:
Magneto/Starter Switch .....CONTINUE CRANKING
Mixture .....IDLE CUTOFF
Low Fuel Boost Pump Switch .....OFF
⋮
FIRE .....EXTINGUISH with Fire Extinguisher

Fig. 4. Structure of the Flight Manual POH-1: Mooney-20V for the Emergency Procedure 'Engine Fire During Start on Ground'.

B. Real-world Experiment

Table IV presents the performance of flight task planning and execution using the GPT-4 engine in the LLM-PIBOT system environment, conducted across four different POHs. The results are averaged over five repetitive experiments for each task.

Embedding Result showed good performance for all POHs except POH-3. This outcome points to the limitations of the Recursive Chunk-based Embedding. The structure of POH-3 involves flight tasks for specific procedures that are composed over multiple pages. This highlights the limitation in extracting multiple pages at once, which is due to the chunk size parameter set for splitting extensive POHs. Moreover, titles (M, S, s) of flight procedures in the manuals were similar, but the title composition of POH-3 differed from the subsections of flight procedures provided to P-P. This difference led to decreased accuracy during the embedding matching process. In the most experiments for task 6, an $ER = 0$ was indicated, reflecting significant performance degradation based on the specific manual's writing structure. For instance, task 6 required the extraction of all content except for the section 'If engine starts' as shown in Fig. 4; however, only the segment pertaining to 'If engine does not start' was extracted.

Generated Task Plans also demonstrated good performance for POHs other than POH-3. However, there were instances where the LLM misidentified entities similar to E_{sim} provided to P-T, such as mistaking 'Standby battery

TABLE IV

RESULTS OF EVALUATION METRICS FOR TASK 1-6 BASED ON FOUR DIFFERENT PILOT'S OPERATING HANDBOOKS.

Task (#ID)	LLM(GPT-4)-PIBOT															
	POH-1: Mooney-20V				POH-2: Cessna-172S				POH-3: DA40-180				POH-4: Cirrus-SR22			
P-P	ER	GTP	CS	RE	ER	GTP	CS	RE	ER	GTP	CS	RE	ER	GTP	CS	RE
1	1.00	0.89	1.00	1.00	1.00	0.89	1.00	0.94	0.00	0.36	1.00	0.60	1.00	0.84	1.00	0.85
2	1.00	0.92	1.00	0.83	1.00	1.00	1.00	0.88	0.00	0.71	0.92	0.63	1.00	0.89	1.00	0.84
3	1.00	0.78	1.00	0.76	1.00	0.98	1.00	1.00	1.00	0.91	0.98	0.74	1.00	0.98	1.00	1.00
4	1.00	0.90	1.00	0.79	1.00	0.89	1.00	0.95	0.00	0.86	1.00	0.95	1.00	0.88	1.00	0.83
5	1.00	1.00	1.00	1.00	1.00	0.80	1.00	0.77	1.00	1.00	1.00	1.00	1.00	0.84	1.00	0.76
6	0.00	0.75	1.00	0.84	0.00	0.93	1.00	0.95	0.00	0.29	1.00	0.90	1.00	1.00	1.00	1.00
AS	0.83	0.87	**1.00**	0.87	0.83	**0.92**	**1.00**	**0.92**	0.33	0.69	0.98	0.80	**1.00**	0.91	**1.00**	0.88

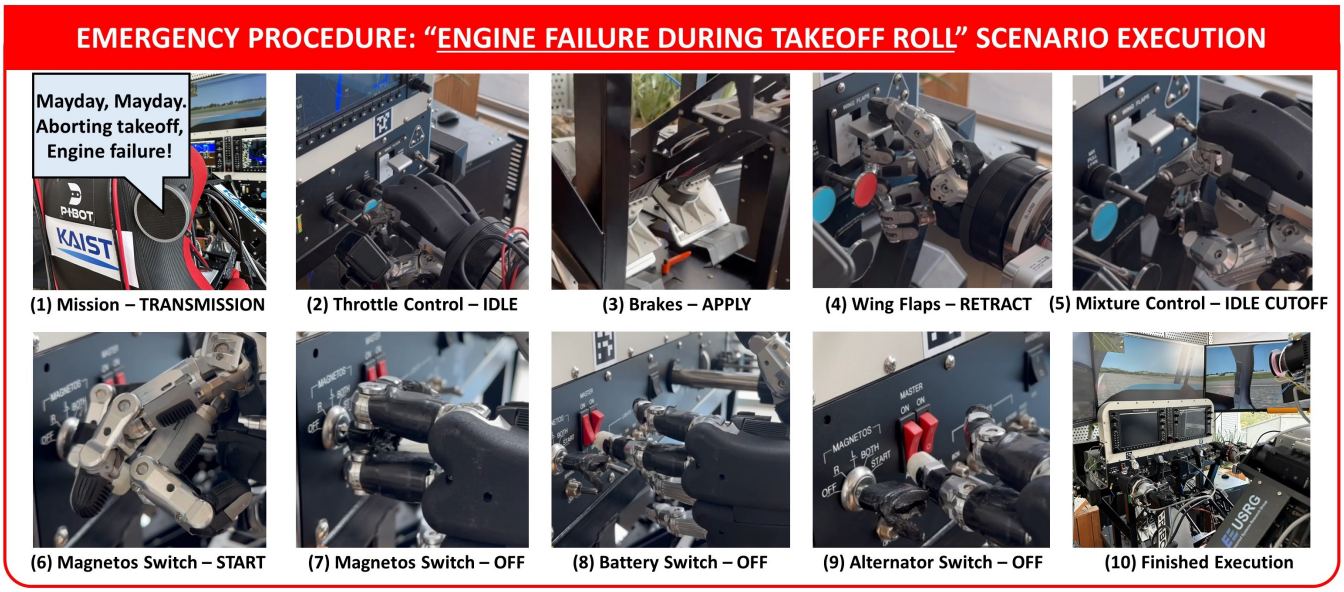

Fig. 5. Execution by PIBOT based on the final GTP generated from the emergency procedure "Engine Failure during Takeoff Roll" in Flight Manual POH2: Cessna-172S. This demonstrates the successful completion of all tasks corresponding to the situation, resulting in the aircraft with the engine off.

switch' and 'Fuel tank selector' for 'BATTERY_SWITCH' and 'FUEL_PUMP_SWITCH', respectively. Moreover, some flight tasks specify 'Throttle' and 'Brakes' as 'as required' instead of a specific quantity. This poses a challenge because the LLM cannot acquire real-time flight status data, leading to ambiguities and the decision to output 'as required' in *P*. The results for POH-3 were below expectations due to the inability to comprehensively extract tasks composed of multiple pages, as previously identified as an issue in ER.

Conditional Statement using P-C generated effective results across all POHs. However, it was observed that the LLM failed to recognize 'Rudder - maintain direction' as 'RUDDER_SLIDE', resulting in the generation of null tuples. In contrast, misinterpretations mentioned in GTP did not lead to the creation of null tuples so that transmitted incorrect task plans to PIBOT. This emphasizes the need for improved entity recognition and interpretation within the system to ensure the accuracy of task execution directives.

Real Execution by PIBOT was successfully performed most tasks according to the final GTP. Fig. 5 demonstrates PIBOT's perfectly execution of flight tasks during an 'Engine Failure during Takeoff Roll' emergency, based on the POH-2. This showcases the system's capability to immediately and

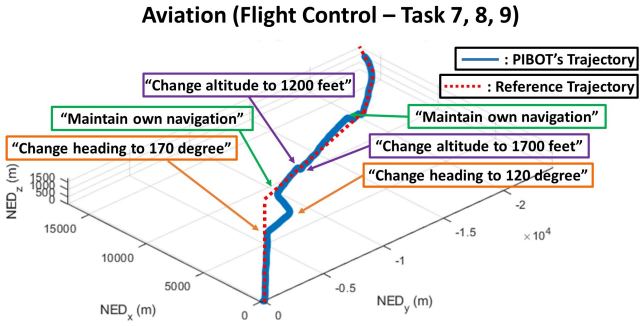

Fig. 6. The flight trajectory (blue line) is compared with the reference trajectory (red dashed line). Commands corresponding to tasks 7 (orange), 8 (purple), and 9 (green) are conveyed to PIBOT during flight operations.

safely shutdown the aircraft's engine, adhering to manual-guided procedures in response to emergencies. However, manipulating various controls with PIBOT's dual arms in the aircraft's cramped and complex environment proved to be a significant challenge. Arm planning occasionally failed to reach the target position safely and accurately. Furthermore, PIBOT proceeded to the next tasks, ignoring the problematic

TABLE V
INFERENCE AND EXECUTION TIMES FOR EACH TASK BASED ON POH-2

POH-2: Cessna-172S		
Task	**Inference Time (s)**	**Execution Time (s)**
1	58.56	74.76
2	17.34	47.70
3	25.58	67.04
4	22.82	50.76
5	18.64	26.65
6	36.26	55.44
7∼9 (avg.)	9.34	2.14

one when errors occurred in the final GTP. These instances were considered failures in our analysis.

The results for flight control tasks show PIBOT's flawless aircraft control as per assigned tasks, highlighted in Fig. 6. Post-takeoff, task-specific flight commands were relayed via microphone, confirming real-time dynamic task execution capabilities, including heading adjustments, altitude changes, and return-to-waypoint execution. Table V presents the final tuple generation and real execution times. These results vary depending on the POH and can significantly differ based on the number of flight tasks, indicating a strong dependence on the POH configuration. However, it is anticipated that inference times will be reduced as the performance of the GPT engine is upgraded in the future, allowing for even faster response times. Consequently, PIBOT accurately adjusted the flight path through precise control even in complex flight conditions, demonstrating its real-time task execution capabilities. This also underlines its adaptability to varied flight situations, showcasing robust flexibility and precision.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed our approach of using LLM to enable humanoid pilots to fly an airplane following a set of rules and procedures provided in various documents, including POHs. The LLM-PIBOT system has demonstrated successful task planning and execution by leveraging POHs for automated inference of flight procedures. We designed LLM prompts to improve flight task planning performance. These prompts categorize flight procedures and supply essential piloting knowledge, facilitating the comprehension of POHs. This system enables PIBOT to respond to commands in real-time during flight, adhere to POHs, and facilitate autonomous flying capabilities.

It is acknowledged that our proposed framework has some limitations in extracting flight tasks spread over many pages of the manuals. To address this, we intend to introduce a feedback loop where the LLM can identify and correct inaccuracies in embedding and task planning autonomously. Also, we intend to refine the system's performance with enriched prompt information and strengthen adaptability across different aircraft manuals. Furthermore, we plan to enable PIBOT to perform fully autonomous flights using vision-language models, navigate to emergency landing sites, and engage in self-decision making by recognizing the surrounding environment.

REFERENCES

[1] H. Song, H. Shin, H. You, J. Hong, and D. H. Shim, "Toward autonomous aircraft piloting by a humanoid robot: Hardware and control algorithm design," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 398–403.

[2] W. X. Z. et al., "A survey of large language models," 2023.

[3] Y. Liu, H. He, T. Han, X. Zhang, and M. L. et al., "Understanding llms: A comprehensive overview from training to inference," 2024.

[4] P. Liu and W. e. a. Yuan, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, vol. 55, no. 9, jan 2023.

[5] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[6] S. Arora, A. Narayan, M. F. Chen, L. Orr, N. Guha, K. Bhatia, I. Chami, F. Sala, and C. Ré, "Ask me anything: A simple strategy for prompting language models," 2022.

[7] T. Brown, B. Mann, N. Ryder, and S. et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[8] K. S. Phogat, C. Harsha, S. Dasaratha, S. Ramakrishna, and S. A. Puranam, "Zero-shot question answering over financial documents using large language models," 2023.

[9] M. Ahn, A. Brohan, and N. e. a. Brown, "Do as i can, not as i say: Grounding language in robotic affordances," 04 2022.

[10] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500.

[11] I. Singh, V. Blukis, and A. e. a. Mousavian, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530.

[12] L. Bärmann, R. Kartmann, F. Peller-Konrad, A. Waibel, and T. Asfour, "Incremental learning of humanoid robot behavior from natural interaction and large language models," 09 2023.

[13] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2086–2092.

[14] OpenAI, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[15] OpenAI., "Gpt-4 technical report," 2023.

[16] J. Devlin and M. W. C. et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 06 2019, pp. 4171–4186.

[17] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," 2023.

[18] T. W. et al., "Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts," 2022.

[19] J. L. Espejel and E. H. E. et al., "Gpt-3.5, gpt-4, or bard? evaluating llms reasoning ability in zero-shot setting and performance boosting through prompts," 2023.

[20] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: from natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, p. 1345–1365, Nov. 2023.

[21] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with llms as translators and checkers," 2023.

[22] L. Sun, D. K. Jha, C. Hori, S. Jain, R. Corcodel, X. Zhu, M. Tomizuka, and D. Romeres, "Interactive planning using large language models for partially observable robotics tasks," 2023.

[23] A. Radford and J. W. K. et al., "Robust speech recognition via large-scale weak supervision," 2022.

[24] P. D. Silva, A. Theeraphol, H. Tang, and K. Pipatsrisawat, "Voice builder: A tool for building text-to-speech voices," in *11th edition of the Language Resources and Evaluation Conference (LREC), 7-12 May 2018*, Miyazaki, Japan, 2018.

[25] OpenAI, "Openai embeddings," OpenAI Documentation, 2024, accessed: 2024-02-05. [Online]. Available: https://platform.openai.com/docs/guides/embeddings