北京邮电大学 本科毕业设计（论文）中期进度报告

Project Mid-term Progress Report

学院 School	International School	**专业 Programme**	**e-Commerce Engineering with Law**		
姓 Family name	Hu	**名 First Name**	Yitong		
BUPT 学号 BUPT number	2020213350	**QM 学号 QM number**	200980434	**班级 Class**	2020215111
论文题目 Project Title	Design and development of a human-agent collaboration model for situation awareness in cockpit				

是否完成任务书中所定的中期目标？Targets met (as set in the Specification)?
YES.

已完成工作 Finished work:

1. Literature Review and Requirement Analysis

The advent of fully autonomous driving is fast approaching. However, until self-driving systems can adeptly manage a diversity of situational challenges, environmental variables, and unforeseen circumstances, the journey toward complete autonomy will be evolutionary, marked by the necessity for human oversight. Concurrently, as autonomous driving technology evolves, an array of new challenges arises. Presently, numerous Original Equipment Manufacturers (OEMs) are adopting Level 2+ or Level 3 autonomous driving capabilities that permit drivers to temporarily relinquish control of specific driving functions, thereby harmonizing vehicular performance with cost-effectiveness.

However, such systems may intermittently necessitate human re-engagement in vehicle operation, and conversely, drivers might require support from autonomous systems in particular scenarios. Achieving fluid communication and collaboration between the driver and the autonomous system is paramount at this level of autonomous driving to enhance driving experience and safety.

Consequently, the industry is in pursuit of a sophisticated, which is core topic of this project, the interactive system—one that fosters situational awareness through the amalgamation of insights from both the vehicle's interior and the external environment, acting as a reciprocal link between the autonomous system and the driver, and orchestrating actions to facilitate human-agent collaboration tasks in the context of awareness.

All in all, the demand for human-agent cooperation in smart cockpit is mainly reflected in improving the user experience and safety of driving:

1. Improve user experience (UX):
    1. When the driver attempts to change lanes, reverse or turn around, the corresponding image will be displayed or a voice/text reminder will be initiated according to the cockpit function

2. When the driver initiates active interaction, the corresponding functions of the cockpit are mobilized to meet user needs.

3. Generate suitable autonomous driving preference suggestions based on driver status/habits, for example: when the driver drinks water, give priority to a smooth driving strategy

2. Improve security:

1. When the driver's attention level is low or unable to respond to a crisis, a reminder action is generated or the autonomous driving system is recommended to take over control.

2. When the driver is not aware of potential threats around him, he proactively displays the source of the threat

3. When encountering complex road conditions that the autonomous driving system is not capable of handling, assess whether the driver is qualified to take over, and then ask the driver to take over the vehicle or remind the driver to pay more attention and prepare to take over.

2. System Architecture Planning

2.1 Introduction

This project introduces the HarmonyCockpit (HCockpit), a framework designed to incorporate cutting-edge multi-modal large models and orchestrate human-agent collaboration (HAC) in cockpit with transparency.

The HCockpit achieves situational awareness by integrating insights into both the internal and external environments of the cockpit and generates natural language level tasks in context-aware to facilitate HAC. Subsequently, HCockpit translates tasks into actions adapting to predefined cockpit functions, thereby guiding both humans and agents towards coordinated action. Metaphorically speaking, within a cockpit setting that includes a human driver and an autonomous driving system (agent), the HCockpit serves as the orchestrating force between human and agent.

To assess the efficacy of the HCockpit and garner deeper understandings, this project developed HarmonyCopilot (HCopilot) as an instance of the HCockpit framework by leveraging state-of-the-art multi-modal large models and incorporating advanced smart cockpit models.

2.2 Preliminaries

This project defines actions of the environment and agents as follows. The global environment is defined as consisting of both the external environment outside the ego vehicle and the internal cockpit environment. Within the global environment, there are three types of agents: the autonomous driving system (AI driver), the human driver, and the HCopilot. The first two are considered independent homomorphic agents, meaning they have the same observation space (the external environment) and action space (vehicle control); differing from the first two, the HCopilot: 1. has global observation, 2. its output actions control cockpit devices. The

core of HCopilot is driven by language models, and its strategy is interpretable and auditable to ensure driving safety.

2.3 System Architecture Design

Input: Comprehensive situational awareness towards the global environment

HCopilot needs to continuously observe the global environment (outside + inside the cockpit) to achieve situational awareness (SA). For the internal cockpit environment, this project only considers the main driver inside the cockpit. HCopilot receives multimodal data inputs including driver's natural language commands, physiological and control data. It utilizes visual-language large models to extract semantic information of the scene and the driver's intention, then saves it as a structured state vector for observation of the human driver.

For the external environment, this project considers that HCopilot should obtain observations of the external environment through the AI driver, and therefore selects a visual-language large model fine-tuned in driving scenarios as simulator for the AI driver. This simulator provides interpretable environmental semantic information and projections of future states and events according to the vehicle's external environment. Similarly, HCopilot saves the aforementioned information as a state vector for observation of the external environment.

It should be noted that HCopilot does not observe another agent—the state of the AI driver, because when the human driver is in control of the vehicle, there is no need to consider the AI driver's strategy or intention; conversely, when the AI driver controls the vehicle, the human driver's intentions or preferences become crucial.

Decision: orchestrate HAC tasks in context-aware

HCopilot combines comprehensive SA of the global environment and context, serving as a messenger and coordinator for the driver and AI driver, orchestrating actions to facilitate HAC, e.g., controlling relevant hardware inside the cockpit to send alerts or information to the driver and providing control suggestions or takeover requests to the AI driver according to the driver's situation. In this process, context awareness is essential. Inspired by the work of Google RT-2, this project designed a memory module for HCockpit, enabling HCopilot to provide more comprehensive task orchestrations based on historical SA.

It is worth noting that there is still a considerable gap between generating HAC tasks and controlling specific cockpit devices because it involves details of the cockpit device control interface, which is beyond the scope of this project's research. HCockpit focuses on the generation and orchestration of more general, high-level HAC tasks, so it designed a simple device control interface for evaluation and demonstration. HCopilot is an instance that has adopted the aforementioned cockpit model.

3. Integrating Machine Learning Model

The HCopilot system incorporates advanced machine learning models to handle various tasks essential for enhancing human-agent collaboration (HAC) in a smart cockpit environment. This section elaborates on the integration of specific models like Pathways Language and Image model (PaLI-X), Pathways Language model Embodied (PaLM-E), GPT-4 variants, and OpenAI Codex for different functionalities within the HCopilot framework. However, integrating ML models is still undergoing and only a part of codes will be demonstrated.

3.1 Observation and Semantic Interpretation

Model: PaLI-X and PaLM-E

Functionality: These models are crucial for interpreting the multimodal data collected from both the internal and external environments of the cockpit. PaLI-X, with its prowess in language and image understanding, processes the visual inputs and natural language communications from the driver to understand the current situation inside and outside the cockpit. PaLM-E, being an embodied model, further enhances the system's capability to predict and propose actions based on the interpreted data, adding a layer of "understanding" to the system's decision-making process.

Implementation:

```
from pali_x import PaLIX
from palm_e import PaLME

# Initialize models
pali_x_model = PaLIX()
palm_e_model = PaLME()

def observe_and_interpret(internal_data, external_data):
    # Observing Internal Environment
    internal_state_vector = pali_x_model.process_internal_data(internal_
data)

    # Observing External Environment through PaLIX
    external_state_vector = pali_x_model.process_external_data(external_
data)

    # Using PaLME for action recommendation based on observed states
    action_suggestions = palm_e_model.recommend_actions(internal_state_v
ector, external_state_vector)

    return action_suggestions
```

3.2 Situational Awareness through Multi-Prompt-Tuning GPT-4 Variants

Model: GPT-4-V (Multiple instances with prompt-tuning for specific observational tasks)

Functionality: Customized GPT-4-V instances, each tuned with specific prompts to handle distinct situational awareness tasks. These instances analyze the state vectors provided by PaLI-X to generate natural language tasks that are context-aware, aiding in precise HAC task orchestration.

Implementation:

```
from gpt4v_instance import GPT4V

# Initialize GPT-4-V Instances for Different Observational Contexts
gpt4v_driving_conditions = GPT4V(prompt="Analyze driving conditions:")
gpt4v_driver_status = GPT4V(prompt="Evaluate driver's state:")

def generate_context_aware_tasks(internal_state_vector, external_state_v
ector):
    # Situational Analysis for Driving Conditions
    driving_condition_tasks = gpt4v_driving_conditions.generate_tasks(ex
ternal_state_vector)

    # Driver's Status Evaluation
    driver_status_tasks = gpt4v_driver_status.generate_tasks(internal_st
ate_vector)

    combined_tasks = {"driving_conditions": driving_condition_tasks, "dr
iver_status": driver_status_tasks}
    return combined_tasks
```

3.3 Translating HAC Tasks to Cockpit Actions

Model: OpenAI Codex

Functionality: Utilizes OpenAI Codex to interpret natural language HAC tasks generated by the GPT-4 variants and translate them into specific, executable commands for the cockpit's hardware and software interface, ensuring the HCockpit's integration with the vehicle's operational systems is seamless and efficient.

Implementation:

```
from openai_codex import OpenAICodex

codex = OpenAICodex()

def execute_cockpit_tasks(tasks):
    for task in tasks:
        # Translate HAC tasks to cockpit function API calls
        code_command = codex.translate_to_code(task)
        exec(code_command)  # Execute the translated command in a safe a
nd controlled environment
```

```
# Example Cockpit Functions API (Pseudocode)
def adjust_seat(position):
    # API Call to Adjust the Seat to Desired Position
    pass

def display_alert(message):
    # API Call to Display an Alert on the Cockpit's Screen
    pass
```

尚需完成的任务 Work to do:

For the HCockpit project, iterative developing and testing are critical steps in developing a robust system ready for real-world applications. Conducting experiments using simulators like GTA V can offer valuable insights into the system's performance under various scenarios, which closely mimic real-life conditions without the associated risks and costs.

1. Conduct Experiments to Evaluate the System

Using Simulators: Choose simulators like GTA V for its realistic urban environments, sophisticated physics engine, and dynamic weather conditions. These features provide a diverse range of driving scenarios, including pedestrian interactions, traffic congestion, and emergency situations, ideal for testing the HCockpit system.

Integration: Develop an integration layer that allows the HCockpit to communicate with the simulator. This layer translates the simulator's sensory data into inputs the HCockpit can understand and vice versa.

Evaluation Metrics: Define key performance indicators (KPIs) such as response time to external events, accuracy in executing given tasks, ease of human-agent collaboration, and system reliability under different conditions.

2. Develop a Demo

Scenario Planning: Create a variety of scenarios within the simulator that highlight the HCockpit's capabilities, such as emergency takeover, assisting human to turn around.

Interactivity: Ensure the demo allows for real-time interaction, giving users the ability to issue commands or change parameters and observe the system's response.

Visualization: Use visualization tools to clearly demonstrate the HCockpit's situational awareness, decision-making process, and action execution. This could include visual overlays on the simulator's output showing detected objects, planned paths, and decision points.

存在问题 **Problems:**

1. **Real-Time Inference and Deployment Cost:** Large model inference suffers from latency issues, and deploying such models locally incurs significant costs due to the need for powerful computing resources.
2. **Sampling Rate for Human and External Environment Data:** Determining the appropriate sampling rate for human data (e.g., physiological and behavioral indicators) and external environmental data is crucial for maintaining system responsiveness and accuracy without overwhelming the system with excessive data.
3. **Integration with Existing Vehicle Systems:** Integrating the HCockpit system with existing vehicle systems could pose compatibility and operational challenges, as vehicles have diverse hardware and software configurations.
4. **User Acceptance and Trust:** Convincing drivers to trust and accept the AI's decision-making, especially in critical and emergency scenarios, remains a considerable challenge.

拟采取的办法 **Solutions:**

1. **Model Optimization:**
   - Invest in optimizing the AI models for efficiency without significantly compromising accuracy, using techniques such as quantization, pruning, and knowledge distillation.
2. **Adaptive Sampling Rate:**
   - Implement an adaptive sampling algorithm that adjusts the rate based on the current driving scenario's complexity. For instance, increase the sampling rate in high-risk situations such as urban driving or bad weather conditions, and decrease it during steady highway cruising.
   - Use active learning strategies to identify and prioritize the most informative data points, optimizing the trade-off between system performance and computational load.
3. **Modular Integration Framework:**
   - CodeX is designed to be integrated for generating actions to control cockpit according to the predefined cockpit profile.
   - The seamless connection of HAC tasks to cockpit equipment control requires a unified and standardized interface, which requires the joint efforts of colleagues in the industry.
4. **Iterative Trust-Building:**

- Deploy the system initially in less critical functions, allowing users to experience and understand the technology in low-risk environments.
- Provide detailed feedback and explanations for the AI's decisions, especially when those decisions override or suggest against the driver's actions. That's one of the main reason LLM is used as core in HCockpit.

论文结构 Structure of the final report: (Chapter headings and section sub headings)

Table of Contents