DuckieGuard

Chang-Yan Li[1], Chong-Sin Huang[2], Hao-Yun Chen[2], Shih-Jie Chang[2], Yi-Lun Wu[2]

I. INTRODUCTION & MOTIVATION

With peoples rising awareness of their own safety, various monitoring machine have been applied to our surroundings. People expect these machine to prevent potential crimes from happening and thus guarantee their safety. Indeed, it works, but its not enough. Lack of the ability to move makes these machine predictable, while lack of the ability to detect danger makes it only useful after events take place. To solve these problems, we decide to develop this application DuckieGuard, which is able to move, detect dangers and send warning messages to owners.

As for moving ability, our DuckieGuard will patrol between two checking points that would have been set previously by users. When meeting the checking points, DuckieGuard will stop and check whether there is anything suspicious. In this way, we can make our DuckieGuard less predictable and more flexible with changeable checking points set by users.

When it comes to danger detection, we plan to enable our DuckieGuard to recognize dangerous things using deep learning. It is when located at checking points that DuckieGuard will implement this function. When detecting something suspicious, DuckieGuard will send warning messages to the users cellphone. In this way, users can get instant information of coming danger and thus able to avoid it.

To sum up, DuckieGuard will be especially distinctive when it comes to its flexibility and instantaneity. Users can set checking points by themselves and receive instant warning messages. In this way, we hope this application can provide users with more protection and a profounder sense of safety.

II. SYSTEM ARCHITECTURE & EQUIPMENTS

A. SYSTEM ARCHITECTURE

The system block diagram is showned in Fig. 1. Guard_node is the core controller in this system. It subscribes to object_recognition_node for recognition results. When any unusual object is detected, the guard_node publishs messages to lane_controller_node and notification_node which control duckiebot and deliver notifications respectively; then all notifications are sent to our http server. User agent can request for notifications from the server. After user receives notifications, he can control the duckiebot to do some helpful things via mobile device.

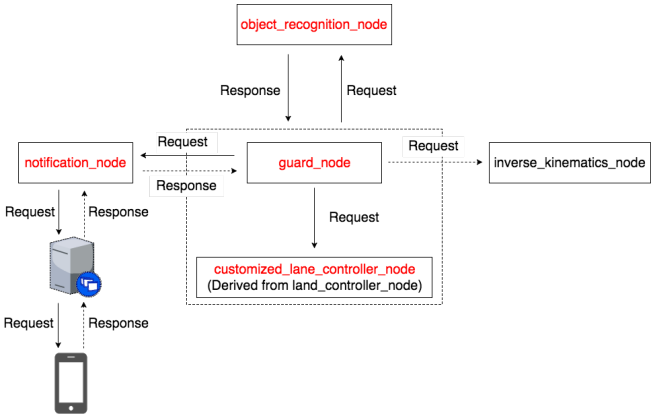

Fig. 1. System block diagram. Red text are the nodes which should be implemented. Dashed lines ara the optional architecture.

Core techniques of each node:

- Guard_node: Flow control.
- Object_recognition_node: OpenCV or TensorFlow.
- Lane_controller_node: Lane following.
- Inverse_kinematics_node: Use buildin node.
- Notification_node: HTTP request or WebSocket, thread handling.

Server side techniques:

- Django
- HTTP request, WebSocket

User agent techniques:

- Android
- HTTP request, WebSocket

B. EQUIPMENTS

The system doesn't need extra hardware on duckiebot(Fig. 2), but need a computer to run the server, and a mobile device to receive notification.

III. SPECIFIC AIMS

I am responsible for the design of the Guard_node. Including Flow control(Using Finite-state machine) and the center control api – consolidate all of the data from other node.

- Flow control – Finite-state machine
- Center control api

IV. APPROACH

In order to finish my work,I will use the Finite-state machine theory. The guard_node will keep request and get

*This work was supported by the Robotics Master Program in National Chiao Tung University, Taiwan

[1]Chang-Yan Li is with National Chiao Tung University, Taiwan. avneger2236@gmail.com

[2]Chong-Sin Huang, Hao-Yun Chen, Shih-Jie Chang, Yi-Lun Wu b.d.researcher@ieee.org

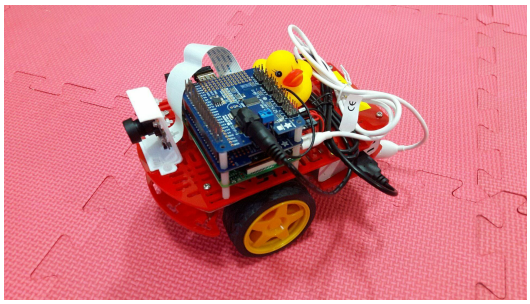

Fig. 2. Duckiebot.

the response of object_recongnition_node. If the duckieguard find the intruder, request to notification_node in order to let the user know. Besides, if user wants to control the duckieguard by himself/herself, guard_node will response what duckieguard see and accept the request of user. Last but not least, request Lane_controller_node by the result of all of the imformations. In order to do my work better, I will need to know preliminary knowledge of other nodes.

V. SCHEDULE AND TEAM COLLABORATION

In the first two week, I will finish the first version of the Finite-state machine, and then start to design the center control api. That means other members nead to finish the first version of the node. In my opinion, it will be better to finish the first version of center control api in one month, so that we have sufficient time to debug. After the attemption for the first month, I will finish a complete (but not perfect enough) version of the center control api. In the last period before the dead line, I will keep revise the center control api to make it perfect. Besides, if there are some new function of duckieguard, I will finish it during the period of the time.