Extending Automotive Vision System by Unmanned Aerial Vehicles

Demetrius Johnson*, Olivia Pellegrini*, Ryan Sauer*, Jonathan Schall*, Khairul Mottakin, Zheng Song
Computer and Information Science Department, University of Michigan at Dearborn

***Abstract*—Unmanned aerial vehicles (UAVs) provide bird's-eye view video, which can significantly extend the vision systems of ground vehicles. However, processing the video on the drone itself may drain the drone's battery rapidly, while sending the video to the ground vehicle for analysis may suffer from higher delay. In this poster, we introduce our ongoing project idea about processing the time-sensitive regions of the video on the drone and transmitting the processed results along with the time-insensitive regions to the car for decision-making and further analysis. To evaluate our idea, we build an experimental system using toy cars and programmable drones. We implement an image-based object detection application which can be deployed on our experimental drone-car system. As a future work direction, we will test with different workload distributions between the drone and the car and report the resulting latency and energy consumption.**

***Index Terms*—Autonomous unmanned aerial vehicles, image processing, object detection**

I. Introduction

With the rise in popularity of Advanced Driver Assistance Systems (ADAS), it is critical to sense the ever-changing surrounding environment of a vehicle in an accurate, wide-range, and timely manner [1]. Due to the restricted view of the car's camera, oftentimes the ADAS system may not have enough time to respond to an accident [2]. Many automotive companies are considering adding unmanned aerial vehicles (UAV) to fly with the ground vehicles, which provide additional sensing information to assist ADAS [3]. In particular, the bird's-eye video view by a UAV can be analyzed (processed) in real-time for highly accurate object recognition and event detection.

However, processing the video on the drone itself may drain the drone's battery rapidly. The battery capacity cannot be expanded efficiently since higher capacity comes with increased weight, making processing the high-resolution video on the drone itself unfeasible. On the other hand, transmitting the video to the ground vehicle for analysis will cause higher delay, given that the maximum WiFi transmission speed of the commercially popular DJI drone is 15 MBps, while the video rate is 25MBps [4].

In this poster, we introduce our ongoing research project about optimizing the video processing procedure for the drone-car collaboration system. To achieve the right balance between energy consumption and latency, we propose dividing video frames captured from the drone into time-sensitive and time-insensitive regions. This categorization depends on their effect on the car's decision-making algorithm. The time-sensitive regions of the video will be processed on the drone and the results transmitted to the car for decision making; the time-insensitive regions will be immediately transmitted to the car for further processing.

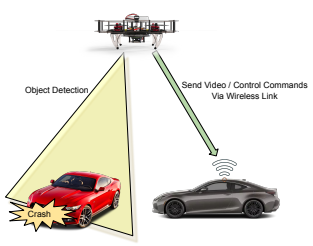
To evaluate our idea, we build an experimental system using toy cars (PiCar-X) and programmable drones (Clover 4.2). We implement an image-based object detection application using the OpenCV-Python library for our experimental drone-car system. The drone can choose to process some data locally or transmit the rest to the car for processing. We will test with different workload distributions between the drone and the car and report the resulting latency and energy consumption, which include both processing and transmission.

II. Experimental System Design and Implementation

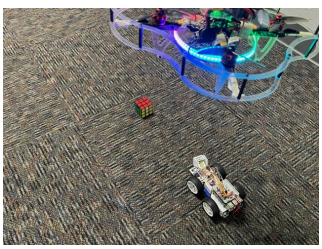
In this section, we introduce our system design, setup, and workflow.

A. System Design

The overall system design is demonstrated by Fig. 1a. Our experimental system consists of a programmable drone and a toy car. To fit the real-life scenario, we assume that the car has significantly more computational resources and battery capacity. The drone has a camera that can capture an overhead view in front of the car. Both the drone and car have computing power which can be used for processing the captured video. The car works as a WiFi access point, and the drone is connected to the car via WiFi. The drone can either transmit the raw video to the car, allowing the car to make the decision to stop, or it can process the video locally and send a command to the car to stop.



(a) System Design



(b) System Setup

Fig. 1: Experimental System Setup

The * authors are undergraduate students. They made equal contributions to this poster and they are listed in alphabetical order.

B. System Setup

Fig. 1b shows our real system setup. We choose to build our system using Clover 4.2 and PiCar-X. The Clover 4.2 is a programmable quad-copter controlled by a COEX Pix flight controller with a Raspberry-Pi-4 as the computer [5]. The Raspberry-Pi comes with a preconfigured image containing the software required to fly the drone. The drone is also equipped with a camera that we will use to obtain our overhead video footage of the car and the surrounding objects. The PiCar-X is a self-driving miniature robot car that is controlled by a Raspberry-Pi [6]. The car is equipped with a 2-axis camera, an ultrasonic sensor, and a line tracking system. The ultrasonic sensor will be used to prevent the vehicle from driving into obstacles in case our object detection fails.

C. Computer Vision-Based Event Detection

We place a Rubik's cube in front of the car and use the red side of the cube to simulate a car crash event. For our initial prototype, both the car and the drone will go in a straight line. The car will be programmed to move forward and will stop if a car crash event is detected by the drone. We develop an OpenCV-based python program for image processing, video analysis, and object detection. That python program can be executed on both the drone and the car, as they both run the Raspbian operating system.

Fig. 2: Demonstration of Event Detection

Figure 2 demonstrates how our CV (computer vision) program works. Each frame from the drone camera is processed in the drone's Raspberry-Pi. Then, the frame is converted from RGB (color values) to HSV (gray-scale values). To view only the red objects, a mask of the frame is created containing only the color red. Since red exists at both ends of the color spectrum, two masks were created to obtain red at both ends and were merged into a single mask. If there are any red contours in the frame, a box around the object can be drawn and a signal can be sent to the car via a TCP socket connection.

III. Future Work

Below are all of the improvements we plan on making in the following months:

A. Complex Computer Vision

Detecting the red side of a Rubik's cube is overly simple and is far from useful in real-world scenarios. To improve our model, we will develop a CV program to detect other vehicles and real-world events. We also want the drone to use CV to track and follow the car automatically.

B. Thorough Experimentation

To collect data on the drone battery life and communication delay, we will set up an automated test environment where we can continuously run the drone and car object detection experiment until the drone's battery life is depleted. We will test different setups of various workload distributions between the drone and the car and report the resulting latency and energy consumption, which include both processing and transmission.

C. Optimizing Drone Battery and Latency

We plan to minimize the latency while still maximizing the drone's battery life. To achieve this balance, we plan to divide video frames captured from the drone into time-sensitive regions and time-insensitive regions depending on their effect on the decision-making algorithm. The time-sensitive regions will be processed directly on the drone while the time-insensitive regions will be offloaded onto a computer-mounted car in order to preserve the battery life of the drone.

IV. Conclusion

UAVs can be a useful extension to ground vehicle vision systems. This poster introduces our idea for balancing the energy consumption and latency of video processing on the drone. We design and implement an experimental system of Raspberry Pi-based cars and drones, and develop a computer vision program to simulate event detection. We plan to further explore how the workload distribution on the drone-car system can be optimized toward the right balance of the energy consumption and processing delay.

References

[1] M.-g. Cho, "A study on the obstacle recognition for autonomous driving rc car using lidar and thermal infrared camera," in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2019, pp. 544–546.

[2] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2020.

[3] "Ford is patenting drone docking on cars," https://www.topgear.com/car-news/future-tech/ford-patenting-drone-docking-cars, accessed: 2023-3-24.

[4] "Consumer drones comparison," https://www.dji.com/products/comparison-consumer-drones, accessed: 2023-3-24.

[5] "Coex clover," https://clover.coex.tech/en/, accessed: 2023-3-19.

[6] "Welcome-to-picar-x's-documentation!" https://docs.sunfounder.com/projects/picar-x/en/latest/, accessed: 2023-3-19.