Fleye on the Car: Big Data meets the Internet Of Things

[Extended Abstract]

Soliman Nasser
University of Haifa

Andew Barry
MIT

Marek Doniec
MIT

Guy Peled
University of Haifa

Guy Rosman
MIT

Daniela Rus
MIT

Mikhail Volkov
MIT

Dan Feldman
University of Haifa.

ABSTRACT

Vehicle-based vision algorithms, such as the collision alert systems [4], are able to interpret a scene in real-time and provide drivers with immediate feedback. However, such technologies are based on cameras *on* the car, limited to the vicinity of the car, severely limiting their potential. They cannot find empty parking slots, bypass traffic jams, or warn about dangers outside the car's immediate surrounding. An intelligent driving system augmented with additional sensors and network inputs may significantly reduce the number of accidents, improve traffic congestion, and care for the safety and quality of people's lives .

We propose an *open-code* system, called *Fleye*, that consists of an autonomous drone (nano quadrotor) that carries a radio camera and flies few meters *in front* and *above* the car. The streaming video is transmitted in real time from the quadcopter to Amazon's EC2 cloud together with information about the driver, the drone, and the car's state. The output is then transmitted to the "smart glasses" of the driver. The control of the drone, as well as the sensor data collection from the driver, is done by low cost (<30$) mini-computer. Most computation is done in the cloud, allowing straightforward integration of multiple vehicle behaviour and additional sensors, as well as greater computational capability.

Keywords

Quadrotors, collision alert system, internet of things, video streaming

1. INTRODUCTION

Autonomous and robotic cars hold a great promise to help human life by both reducing accident rate, and releaving us of the need to drive. A major component in such cars is based on Advanced Driver Assistance Systems (ADAS) [4]. These provide collision prevention and offer a wide range of driver safety solutions combining computer vision, planning/AI, and control, with databases for local information, and a user interface.

However, current systems are based on cameras that are positioned on the car itself, limiting their field of view to that of the car's surrounding. In order to explore the next generation of such systems, we extend these in two dimensions: by incorporating data from detached sensors and cloud-based data, and by streaming the data directly to the driver's sensory input.

In order to easily allow technology developers to explore human-computer interface, and system design, and in order to make this system available to the the makers and robotics communities, we propose a system of this kind that is based solely on off-the-shelf, cost effective components.

2. SYSTEM OVERVIEW

Our system, Fleye, provides the driver a real-time streaming video from the view of a hovering quadcopter above the car, and combines additional sensors. Video and sensor data is first sent to the cloud to demonstrate further processing and incorporation with additional sources. The output video from the cloud is then projected to the smart glasses of the driver. The driver can also control the quadcopter using a dictionary of voice commands. We chose a quadcopter that is very low-cost, small, safe, and legally provided by popular toy stores. This, in turn, required us to augment the system with componenets to remote-control the quad.

Applications of such a system include: scouting traffic (for example, to avoid traffic jams), taking landscape pictures, detecting obstacles on the road ahead, and tracing empty parking spaces in the street or parking lots. Such a system may also save the driver's life if, for example, it detects a crossing car around the corner (out of the drivers's view) that went through a red light and is about to hit the driver. Our main contribution is to build an open platform for the Fleye system, that is based on low-cost Internet of Things (IoT) hardware, and algorithms that run on the cloud. The cost of each item in the system, except for the smart glasses, is roughly \$40.

The open software that we developed for our system can be freely uploaded from our web site (Will be added in the final version). The software includes: (a) A novel tracking algorithm, for localizing the quadcopter using regular web-cameras. (b) PID controller for the quadrotor. (c) quadrotor PPM encoding code on the Arduino. (d) Video stabilization OpenCV code. The video output is uploaded via http streamingto the smart glasses. (e) Communication code for sending the data from the wearable board to the cloud in real-time. (f) Code for adding more voice recognition commands to our system, based on the Sphinx [3] system.

Components The hardware of our system consists of the following off-the-shelf products: (a) Controllers: Intel's Galileo, Arduino UNO R3. (b) Quadrotor: Walkera's Ladybird, w/o sensors or transmitters, only a receiver. (c) Walkera DEVO 7E quadrotoc transmitter. Comes in the same package of the quadcopter. (d) Video transmission: TX5805: analog radio low-weight FPV (first pilot view) camera that is mounted on the quadcopter and sends radio 5.8GHz video signal. Shipped together with the Ladybird FPV version. RC5805: 5.8Ghz Video receiver. (e) Diamond VC500: analog to USB video grabber[1]. (f) Amazon's web services: for running computer vision algorithms on the EC2 cloud. (g) Cameras (Optitrack Flex 3 or Logitech C920 Webcam): for tracking the quadcopter. (h) VUZIX m100: smart glasses for the driver[2].

Hardware Setup The setup of our system is sketched in the poster. **Camera on the quadcopter.** The analog camera, carried by the quadcopter, is connected to a video-transmitter unit which transmits the video to the video-receiver via 5.8Ghz radio signal. The video-receiver is connected to an Analog-to-Digial converter (video capture device) which outputs a digital video to the IOT-board via a USB connection. The video is uploaded to the Internet via the computation unit to the cloud (Amazon EC2).

Cameras on the car. We use a pair of cameras that are mounted on front of the car; see Fig. ??. In our test both Optitrack's Flex 3 tracker, and our own tracking with Logitech C920 webcam. We used a simple launch pole in front of the car to assure the quad is in the cameras' field of view at take-off.

In the car there is the Galileo board and the smart glasses that the driver wears. The Galileo collects information from the driver and the car, such as temperature and speed of vehicle, as well as the video stream from the video capture device. The glasses download the streaming video and presents it to the driver.

Localization and Tracking Tracking Algorithm. We implemented particle filtering algorithm in pose-space (6DOF), adding colored markers to reduce mis-identifications, and track the quadrotor in a RGB input image from a regular web-camera.

Control Algorithm. The control of the quad is in the form of a PPM signal, formed on the arduino.

Given the current vehicle position $\mathrm{x} \in \mathbb{R}^3$ and desired vehicle position $\bar{\mathrm{x}} \in \mathbb{R}^3$ from the tracking sub-system, we compute the desired roll u_ψ, pitch u_θ, yaw u_ϕ, and throttle u_t control outputs using four separate PID loops.
The final throttle output is computed by adding combining A gravity compensation value u_0 is added to the throttle output, finally scaling the thrust according to the pitch and roll of the quadcopter. The yaw control output u_ϕ can be set to 0 if yaw control is not desired or necessary.

Finally, all outputs $u_\theta, u_\psi, u_\phi, u_t$ are capped to remain within the physically possible values.

3. CLOUD PROCESSING

The streaming video data from the camera on the hovering quadcopter, the cameras on the car, the glasses, and the sensors on the Galileo board are transmitted to Amazon's EC2 cloud for high performance computation. The result is a processed video image, possibly with additional markers and text, that is uploaded in real time to an http address. The glasses project this http content to the driver.

As a sample application, we provide an algorithm and its implementation that runs on the cloud and stabilizes the input video from the tilting quadcopter. The algorithm also gets sensors data, such as temperature, from the Galileo board and add it to the output video. We give more details in this section.

Due to various reasons like: winds, hovering accuracy and more - the video captured from the quadcopter's camera is unstable and shaky. Therefore, instead of directly displaying the video on the smart glasses, we run video stabilization code and present the output video stream to the driver for a better experience.

4. EXPERIMENTS

In order to test the effect of high latency in low-compute devices on quad control stability, we added an artificial delay and measured the average error from a desired target position, as shown in the poster. The minimum update rate was roughly 10 iterations per second. Longer delay in the loop caused the quadcopter control to be unstable. When the delay decreased too much (around 90 iterations per seconds), the error actually gets larger, due to PID calibration paremeters.

5. REFERENCES

[1] http://www.amazon.com/easycap-audio-video-capture-adapter/dp/b0019sssmy.

[2] http://www.vuzix.com/consumer/products_m100/.

[3] K.-F. Lee. *Automatic Speech Recognition: The Development of the Sphinx Recognition System*, volume 62. Springer, 1989.

[4] E. Raphael, R. Kiefer, P. Reisman, and G. Hayon. Development of a camera-based forward collision alert system. *SAE*, 2011.