A Testing Platform for On-drone Computation

Conference Paper · October 2015
DOI: 10.1109/ICCD.2015.7357188

5 authors, including:

Wang Zhou
Northwestern University
21 PUBLICATIONS **23** CITATIONS
SEE PROFILE

Oki Gunawan
IBM
136 PUBLICATIONS **8,574** CITATIONS
SEE PROFILE

Theodore van Kessel
IBM
30 PUBLICATIONS **134** CITATIONS
SEE PROFILE

Hendrik F. Hamann
IBM
136 PUBLICATIONS **4,860** CITATIONS
SEE PROFILE

A Testing Platform for On-drone Computation

Wang Zhou
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois 60208
Email: wangzhou@u.northwestern.edu

Dhruv Nair
Department of Mechanical Engineering
Columbia University
New York City, New York 10027
Email: dhruv.nair@gmail.com

Oki Gunawan, Theodore van Kessel and Hendrik F. Hamann
IBM T.J. Watson Research Center
Yorktown Heights, New York 10598
Email:{ogunawa,tvk,hendrikh}@us.ibm.com

***Abstract*—This paper describes the development of a test bed for an on-drone computation system, in which the drone plays the game of ping-pong competitively (YCCD: The Yorktown Cognitive Competition Drone). Unlike other drone systems and demonstrators YCCD will be completely autonomous with no external support from cameras, servers, GPS etc. YCCD will have ultra-low power computation capabilities including on-drone real-time processing for vision and localization (non-GPS based). Architectural design and processing algorithms of the system are discussed in detail.**

I. Introduction

Unmanned aerial vehicles (UAV) have been extensively studied in areas of automation, surveillance, transportation, and exploration. Quadcopters are of special interest due to their agility and maneuverability. They have been developed to juggle balls solo [1] and cooperatively [2]. It has also been shown that such drones can grasp and transport objects, and assemble architectures autonomously [3], [4]. While autonomous quadcopters leverage GPS technology, indoor ones have to rely on other means besides GPS. In these situations, the positioning and navigation has relied on external cameras and motion tracking systems with off-drone computation, which assist the drone to analyze sensor data, compute trajectory and control the flight paths [5]–[7]. Alternatively, Shen *et al.* [8] implemented scanning laser range sensors and an Iterative Closest Point algorithm to provide position estimations.

In this paper, we discuss the development of a test bed for an on-drone computation system – the Yorktown Cognitive Competition Drone (YCCD). YCCD is designed to be completely autonomous with no external support from motion tracking systems, servers, and GPS. All computation is done with on-board low-power "companion computers". The local positioning system based on stereo imaging is also integrated on the drone. Our ultimate goal is to demonstrate the operation of this platform through an autonomously played game of ping-pong at a reasonably competitive level. The development of this platform is motivated by two underlying objectives: (a) the synthesis of a challenging UAV-centered, real-time image processing application workflow that can be used to optimize future on-UAV embedded processors for power-performance efficiency; and, (b) the creation of a test bed that can be used to study the efficacy of actual power-performance optimized, fault-tolerant embedded processor systems.

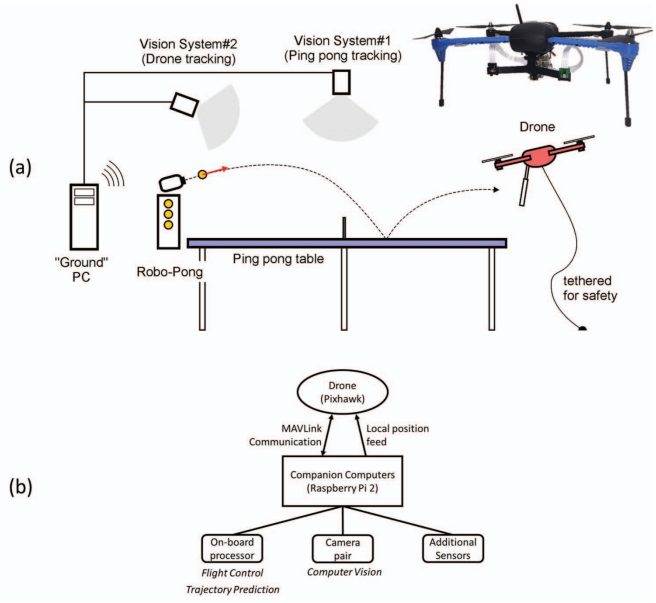

Fig. 1. (a) Development setup for YCCD. A robo-pong machine is used to serve balls. Two off-board vision systems are only used as alternative positioning system in the initial phase to speed up the process. The inset depicts the actual drone infrastructure. (b) Schematics of the YCCD architecture.

II. System Architecture and Development

The system for YCCD is illustrated in Fig. 1. The drone hovers on one side of the table (and is tethered to the ground for safety). A robo-pong machine serves balls on the other side. In the initial phase of this project, two off-board vision systems built from Microsoft Kinects and a PC running MATLAB are used as alternative positioning systems to track the position of the drone as well as the ping-pong ball.

To achieve the functionalities we are proposing, our YCCD consists of three parts: a quadcopter, companion computers, and on-board local positioning system. The mainframe of the YCCD architecture is depicted in Fig. 1b, and its infrastructure is shown as the inset of Fig. 1a.

A. Quadcopter Feasibility study

Among all the commercially available drones, we choose 3DR IRIS+ from 3DRobotics as the development platform based on the criteria below:

978-1-4673-7166-7/15/$31.00 ©2015 IEEE

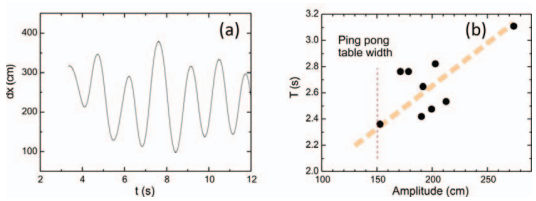

Fig. 2. Drone kinematics analysis. (a) Extracted lateral position vs time. (b) Amplitude-dependent oscillation period of the drone. The time needed for the drone to go over the width of a ping-pong table is estimated as $T = 2.3$ s.

- *Open-source*. Both the built-in pilot system and external controlling libraries are open-source projects, and the source codes are available on github. This enables customizing the control algorithms and the hardware to develop an autonomous drone system.
- *Controllability*. IRIS+ utilizes the general Micro Air Vehicle Link (MAVLink) communication protocol to transmit data and commands to/from Ground Control Stations (GCS).
- *Payload*. The moderate size of the drone allows it to carry up to 400 grams of payload, which is sufficient for the companion computers, a ping-pong racket and an on-board local positioning system.
- *Agility*. The drone is agile enough to respond fast as described below.

We first evaluated the feasibility of the drone playing ping-pong by analyzing the drone's kinematics. It is important that the drone can move sufficiently fast to catch up with the ping-pong ball in a game. For this we flew the IRIS+ manually with the remote controller from the left to the right and from the back to the front while recording a video. We tracked the position and eventually extracted the motion of the drone. Figure 2a shows the extracted lateral position of the drone during the flight. From the position data, we calculated the oscillation period. Unlike a harmonic oscillator, the period depends on the oscillation amplitude (Fig. 2b). We consider the case where the drone moves within the width of a ping-pong table $W = 1.5$ m, and estimate the period $T = 2.3$ s. In a ping-pong game, the drone would nominally stay at the center of the table and move left/right, therefore the response time would be within $T/4 = 0.6$ s. The response time is small compared to the ping-pong traveling time across the table which was measured to be $t = 1.1$ s during our test for a normal and slow speed game. Therefore we concluded that it is feasible for the IRIS+ to play a ping-pong game.

B. Companion Computers

YCCD is designed to fly autonomously, with control programs running on companion computers which are carried by the drone. Due to the power and payload limitations, the companion computers need to be low-power and light-weight. It is also advantageous that the companion computers have adequate computation capability for other tasks.

In this initial stage, a Raspberry Pi (RPi) 2 board was selected as the companion computer for YCCD. It is a low-cost, low-power, single-board linux computer, featured with a quad-core ARM Cortex-A7 CPU and a VideoCore IV dual-core GPU. It supports programming in Python which is the main programming language we used for the control algorithms. The computer weights only 45 grams which is within our payload constraint. It provides USB and UART ports for serial communication and data transmission, as well as an external sensor reader.

To set up the RPi board as the companion computer for the drone, we connected the UART serial port to the Telemetry 2 port on Pixhawk which is normally used for receiving data from the remote controller. The port is then configured to use MAVLink as its communication protocol to receive and send data between the drone and the companion computer. The MAVProxy python module from the DroneKit library is used to establish the communication over serial port. The RPi is powered by the drone battery via a DC-to-DC converter to step down the voltage to 5 V which is required by the board. The power consumption of each board is 4 W, less than 3% of the output power of the drone battery (5100 mAh lithium polymer 3S battery, output voltage $V_{cc} = 12$ V, maximum flight time is about 20 minutes), so the low-power RPi boards have negligible impact on the flight time.

As indicated in Fig. 1b, the RPi boards will be responsible for flight control, trajectory prediction, computer vision with two RPi cameras mounted, as well as reading and processing additional sensor information such as sonar sensors. Preliminary tests suggest that the computational complexity of the multi-task algorithms are within the capacity of RPi 2 boards.

C. Local Positioning System

The main challenge of indoor navigation is to determine the position of the drone in 3D space (x, y, z) and feed the drone its real-time location. For this we developed a Local Positioning System (LPS) [5]–[7] that works as a virtual GPS that feeds the data to the drone. We developed two approaches:

(1) Off-board LPS system: For initial phase of our development we developed a LPS system to track the 3D position of the drone relative to the ping-pong table. We used a Microsoft Kinect system and a PC running MATLAB as an off-board vision system. The Kinect system provides two images: a standard RGB image and the depth image of the scene. We performed camera calibration on the RGB camera to obtain the intrinsic camera parameters and we set the center of the ping-pong table as the origin, that determines the extrinsic camera parameters [9], [10]. We attached a larger ball (70 mm diameter) on the drone to perform tracking. By combining the camera normal position vector of the ball in the RGB image and the depth information we were able to identify the 3D world coordinate of the drone. Note that we also had to calibrate the transformation matrix (scale, rotation and translation) from the Kinect RGB image to its depth image.

(2) On-board LPS system: We used two on-board Raspberry Pi cameras to perform stereoscopic imaging. The payload of the vision system (including two RPi boards, cameras and 3D printed mount as shown in the inset of Fig. 1a) is

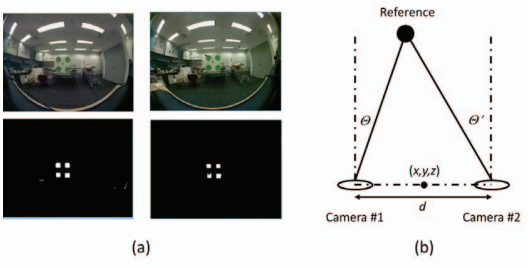

Fig. 3. Stereo imaging to determine the position of the drone in a room with reference stickers. (a) View from left and right camera. The bottom row is image output after thresholded for the color green. (b) Trigonometry algorithm to determine the 3D position of the drone (x, y, z) which is defined as the center point between two cameras.

approximately 250 grams which is well within the 400 gram recommended limit for the drone. In order to determine the drones position in the room, we use four reference points (four green stickers separated by $a = 450$ mm in a 2×2 grid in Fig. 3a). The image is first thresholded for the color green, and the contours of interest are extracted. Next we determine the centroid for each contour, with each camera. The angles Θ and Θ' can be determined from the pixel distance of the centroid from the camera center. With known distance d between two cameras, we are then able to determine the distance of these points from the camera through stereo imaging and trigonometry, which is illustrated in Fig. 3b, and therefore estimate the 3D position of the drone (x, y, z) in the world coordinates.

In the final version, the drone position (x, y, z) coordinate is estimated by on-board vision system and directly fed to the drone after transforming the 3D coordinate as a virtual GPS location (lat,lon,alt). The virtual GPS location is then formatted into an NMEA standard message and sent from RPi to the Pixhawk via a serial port. The drone is customized to take the virtual GPS string as a "real" GPS module which provides continuous localization. By using synthetic GPS feed, the drone is GPS locked even at indoor environment, and we can take advantage of the piloting system on Pixhawk which is optimized with GPS guidance and avoid writing control programs from scratch.

III. Pingpong Tracking and Trajectory Prediction

In order to be able to play ping-pong, the drone has to be capable of "seeing" where the ball is and "knowing" where the ball will be in a later time. In this section we discuss our algorithms to track the motion of ping-pong and predict its trajectory.

A. Pingpong Tracking Based on Computer Vision

Similar to the Local Positioning System, we developed two techniques for the ping-pong tracking:

(1) Off-board vision system: Similar to off-board drone LPS system in Section II-C, we employ another Microsoft Kinect camera mounted on top of the ping-pong table to track the 3D position of the moving ping-pong.

(2) On-board vision system: Here we utilized the same pair of on-board cameras to track movement of the ping-pong ball. The process to track the ping-pong ball relies on an extended Mean Shift and CAMshift algorithm [11]. These algorithms work by minimizing the distance between two probability density functions that are represented by a reference and candidate histogram. The distance is both shape and size invariant, therefore it can be used regardless of how near or far it may be from the drone. These methods have the added benefit of being computationally quick [12], [13], and this is imperative for our low power vision system.

An input reference image of the ball, I_o and an initial region of interest, I_{ROI} are first established. We convert I_{ROI} into HSV color space, and work exclusively with the hue channel. The histogram of the ROI, our object model, q is approximated by a histogram of m bins, $q = \sum_{o=1}^{m} q_o = \sum_{o=1}^{m} \sum_{i=1}^{n} \delta[c(x_i) - o]$, with q_o being probability of pixel belonging to the o^{th} bin, $c(x_i)$ being the function that associates pixel values to bin numbers, and n being the number of pixels in the region. Then, I_{ROI} histogram is then normalized $p_o = \min\left(\frac{255}{\max(q)}, 255\right)$ so that the generated probability density image in within the [0,255] range.

Rapidly moving objects create motion blur, which can cause the Mean Shift and CAMshift algorithm to fail. To account for this, we have employed an expanded local search method to verify whether our candidate is the object to be tracked. The first step is to calculate the Bhattacharyya similarity between the reference image histogram and the candidate histogram $\rho(p, q) = \sum_{o=1}^{m} \sqrt{p_o q_o}$. If the similarity coefficient is above a certain threshold value, we simply return the identified candidate; otherwise, the search window is expanded to twice its size, and split into four parts. We iterate over all four windows, and the sub window with the maximum similarity coefficient above the threshold value is selected as the candidate. Finally we use a Kalman filter to correct the position of the ball, and anticipate its position in the next frame. In this way we can account for instances where we might lose the ball.

B. Physical Modeling and Trajectory Prediction

For a successful gameplay, the drone has to respond fast enough to return the ball. For this purpose, the drone should be able to estimate the whole trajectory of the ping-pong ball based on some small initial fraction of the ping-pong motion images, therefore a detailed physical modeling is needed for trajectory prediction. Here we developed a simple kinematics model of the ping-pong ball, and also set up a simple wind-tunnel to obtain some important experimental parameters. In this first model we ignore the spin of the ball.

As shown in Fig. 4, the ball kinematics consists of two of physical processes: projectile motion and collision. There are three parameters that govern the motion of the ping-pong ball as illustrated in Fig. 4, the initial velocity vector v_0, the drag

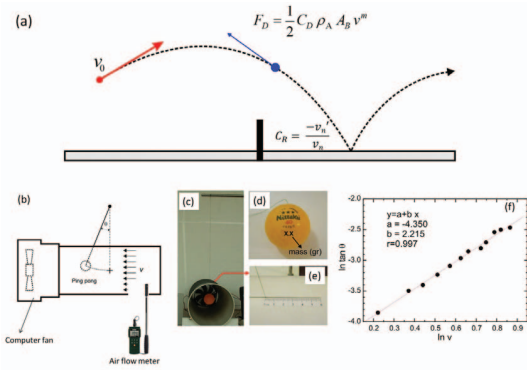

Fig. 4. (a) Physical model of a ping-pong ball trajectory. (b) Wind tunnel experiment to verify the air drag model. (c) Wind tunnel with computer fan and ping-pong in place. (d) The ping-pong ball. (e) A ruler is placed on the tunnel to measure the lateral displacement dx of the string, and then the deflection angle θ can be calculated by $\tan\theta = dx/h$, where h is the height from the anchor point to the top of the tunnel. (f) Linear fitting to extract the drag coefficient and the velocity exponent.

force coefficient C_D and the coefficient of restitution of the collision with the table C_R.

When the ball is moving through the air it experiences an aerodynamics drag force due to air viscosity given as $F_D = -\frac{1}{2} C_D \rho_A A_B v^m$, where C_D is the drag coefficient, $\rho_A = 1.2$ kg/m^3 is the air density, A_B is the cross section area, v is the velocity of the ball, and $m = 2$ is the velocity exponent. We verify the model with a wind tunnel experiment as illustrated in Fig. 4b. We use a computer fan to suck air out of a tunnel to create a laminar air flow whose velocity is measured by an air flow meter. A ping-pong is hanged in the tunnel by a string, and is pulled aside by the drag force. At steady state, the deflection angle θ of the string depends on the drag force or air velocity: $\tan\theta = F_D/m_B g = C_D \rho_A A_B v^m / 2 m_B g$. We run the computer fan at various frequencies to provide different flow velocity, and measure the corresponding deflection angle θ . By plotting $\ln\tan\theta$ vs $\ln v$, we extract the drag coefficient $C_{D,\text{exp}} = 0.40$ from the intercept and velocity exponent $m_{\text{exp}} = 2.2$ from the slope (Fig. 4f). These experimental data agree reasonably well with the theoretical values $C_D = 0.47$ and $m = 2.0$.

When the ball collides on the table, it loses part of its velocity due to inelastic collision. The restitution coefficient is given as $C_R = -\frac{v_n'}{v_n}$, where v_n (v_n') are the velocity before (after) the collision at normal to the table. Similar to drone kinematics characterization, we extracted the vertical position and velocity of the ball from a video of a ping-pong ball bouncing on the table, and by comparing v_n and v_n' during a collision, we got the average restitution coefficient $C_R = 0.89$.

While C_D and C_R are extracted, the initial velocity v_0 can be fed by tracking the ping-pong with computer vision. Further verification of this model to the actual ping-pong trajectories are still in progress.

IV. Conclusion

Here we have developed a test bed for an on-drone computation system (YCCD) and demonstrated its functionality by playing a ping-pong game. Compared to other test beds where GPS localization or external motion tracking systems and servers are required to provide location, compute trajectory and control the flight, our YCCD system is self-contained, energy efficient and simply structured. All the computation is realized by the companion computers which are mounted to the drone. We use a stereo imaging technique with two RPi cameras to provide position coordinates of the drone in 3D space which are then transformed to a virtual GPS location and fed to the drone, as well as to track the ping-pong ball. Microsoft Kinect systems have also been implemented as our initial drone positioning system and ping-pong trajectory tracking. Our new test bed opens up a new venue for the development of on-drone computing systems.

Acknowledgment

The real-time image processing and GPS-free navigational software development aspects of this project, as they relate to real-life military UAV applications, are sponsored in part by Defense Advanced Research Projects Agency, Microsystems Technology Office (MTO), under contract no. HR0011-13-C-0022. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This document is: Approved for Public Release, Distribution Unlimited. The authors would like to thank Pradip Bose, Alper Buyuktosunoglu, Karthik Swaminathan, Sharathchandra Pankanti and Chung-Ching Lin for enlightening discussion.

References

[1] M. Muller *et al.*, "Quadrocopter ball juggling," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5113–5120, 2011.
[2] R. Ritz *et al.*, "Cooperative quadrocopter ball throwing and catching," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4972–4978, 2012.
[3] D. Mellinger *et al.*, "Cooperative grasping and transport using multiple quadrotors," *Distributed autonomous robotic systems*, pp. 545–558, 2013.
[4] F. Augugliaro *et al.*, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, pp. 46–64, 2014.
[5] N. Michael *et al.*, "The grasp multiple micro-uav testbed," *IEEE Robotics and Automation Magazine*, vol. 17, pp. 56–65, 2010.
[6] S. Lupashin *et al.*, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, vol. 24, pp. 41–54, 2014.
[7] J. P. How *et al.*, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems*, vol. 28, pp. 51–64, 2008.
[8] S. Shen *et al.*, "Autonomous multi-floor indoor navigation with a computationally constrained mav," *IEEE International Conference on Robotics and Automation*, pp. 20–25, 2011.
[9] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 323–344, 1987.
[10] J.-Y. Bouguet, "Camera calibration toolbox for matlab."
[11] X. Chen *et al.*, "Real-time object tracking via camshift-based robust framework," *IEEE International Conference on Information Science and Technology*, 2012.
[12] T. Vojir *et al.*, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.
[13] A. Yilmaz *et al.*, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1531–1536, 2004.