An Autonomous, Visually-Guided, Counter-sUAS Aerial Vehicle with Net Countermeasure

Timothy K. Horiuchi[1,5,6], Marc Paras[2], Tim Cervi[2], Brent Oursler[2], Santiago Sanz[3], Joshua Gaus[2], Daniel McColl[3], Stephen Woller[1], Donald Woodbury[4], Imraan Faruque[3], and Huan Xu[3, 5].
[1]*Department of Electrical and Computer Engineering,* [2]*Department of Mechanical Engineering*
[3]*Department of Aerospace Engineering,* [4]*Division of Research*
[5]*Institute for Systems Research,* [6]*Neuroscience and Cognitive Science Program*
University of Maryland, College Park, MD 20742

To address the growing concern of small unmanned aircraft systems (sUAS) entering the airspace over a public event, building, or other protected spaces (whether for recreational purposes or with malicious intent), we are developing a counter-sUAS solution. This solution is based on an aerial intercept vehicle that autonomously pursues the target sUAS and launches a net to entangle the propellers and/or unbalance the sUAS to bring it to the ground. The intercept vehicle is a hexacopter that is guided by a visible-light camera and a single-pixel lidar mounted on a gimbal to track the target sUAS. A custom-built, lightweight, C02 cartridge-powered, net launcher mounted to the top of the vehicle fires when the intercept vehicle is oriented properly at the target and when it is in range. In this manuscript, we describe the problem constraints, system design, image processing chain, computational infrastructure, physical implementations, and various measurements of performance. We describe the current state of development and our experience with the challenge of designing effective countermeasures.

I. Introduction

WITH the current explosion in the availability and use of small unmanned aircraft systems (sUAS) around the world, there is a glaring lack of options available for anyone seeking to prevent a sUAS from entering and operating in a sensitive airspace. While the majority of incidents from sUAS have been and are likely to come from non-malicious actors or due to unintended operation, unauthorized aerial vehicles can create a hazardous situation. In the case of malicious aerial vehicles with protected communications or autonomous capability, a direct physical intervention is likely to be required. In all cases, however, a very short reaction time is needed. There is currently a great need for a low-cost, rapid-response, counter-UAS system for defending a small, defined airspace.

II. Design Constraints and System Organization

To address the defensive capability needed, we are developing a counter-UAS solution utilizing physical intervention to disable the target vehicle. This solution is based on an unmanned aerial intercept vehicle that autonomously pursues the target sUAS visually and launches a net to entangle the propellers and/or unbalance the sUAS to bring it to the ground. While communication with the ground is maintained for monitoring, the counter-sUAS system can complete its mission even in the presence of RF and GPS jamming.

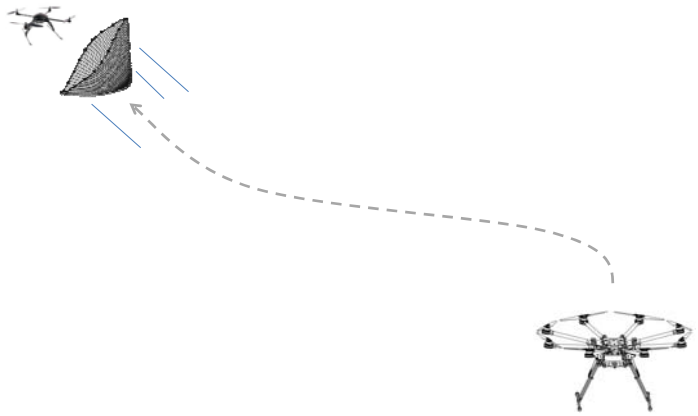

Figure 1. System Concept. *After a ground-based sensor system has detected an intruder sUAS and directed the intercept vehicle's sensors to acquire it, the system autonomously homes in on the target and launches an entanglement net.*

A. Design Constraints

There are two guiding themes for this project: low-cost and autonomy. Only commercial off-the-shelf or 3-D printable parts and open-source software (e.g., OpenCV [1]) have been integrated. Autonomy (after initial targeting) dictates that all sensors and computation must ride on the vehicle and that any interruption of communication with the ground does not affect the completion of the mission. A key component choice in our design is the use of the widely-adopted and low-cost Raspberry Pi 2.0 [2] computer. This quad-core processor running the Raspbian [3] (Linux) operating system allows the use of OpenCV computer vision software libraries along with the multiprocessor-capable Python (2.7.3) language.

B. Vision Algorithms

While most successful computer vision algorithms for tracking targets typically require significant computational resources due to the need to effectively segment the object from the ever-changing background via recognition methods, we have chosen to limit our counter-sUAS mission to scenarios where the target can be seen against the sky as the background (Figure 1). While this constrains our flight trajectory to remain below the altitude of the intruder sUAS, this is the natural flight path that would occur in most defensive situations and camouflages our approach from the target UAS. This constraint dramatically simplifies the visual tracking problem to one that is tractable at the high frame rates needed to home in on a moving target. Interestingly, this is the same strategy used by predatory visual insects such as dragonflies [9] and robber flies. Another key design component has been to maintain a flight data log (including camera images) that allows a post-hoc analysis of performance that aids in offline debugging.

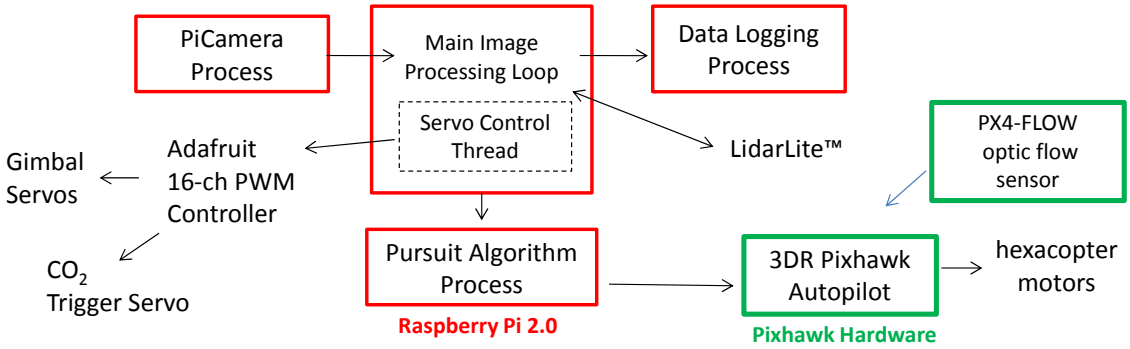

Figure 2. Software and Hardware Architecture. *Red boxes indicate software running on the Raspberry Pi 2.0 hardware. Green boxes indicate software running on the 3D Robotics Pixhawk Autopilot [4] hardware. The servo controller board [5] and LidarLite [6] sensor communicate using the I2C serial communication protocol, while the PiCamera [7] is a camera module*

specifically designed for the Raspberry Pi board. A second lidar is used for altitude control near the ground.

C. The "De-clouding" Filter

Although the basic strategy of an ascending intercept trajectory removes ground-based objects from the visual background, clouds can often interfere with visual tracking by either directly generating contrast blobs (i.e., false positives) or by providing enough background texture to prevent proper blob detection (false negatives). Fortunately, there are heuristics that can significantly improve the situation.

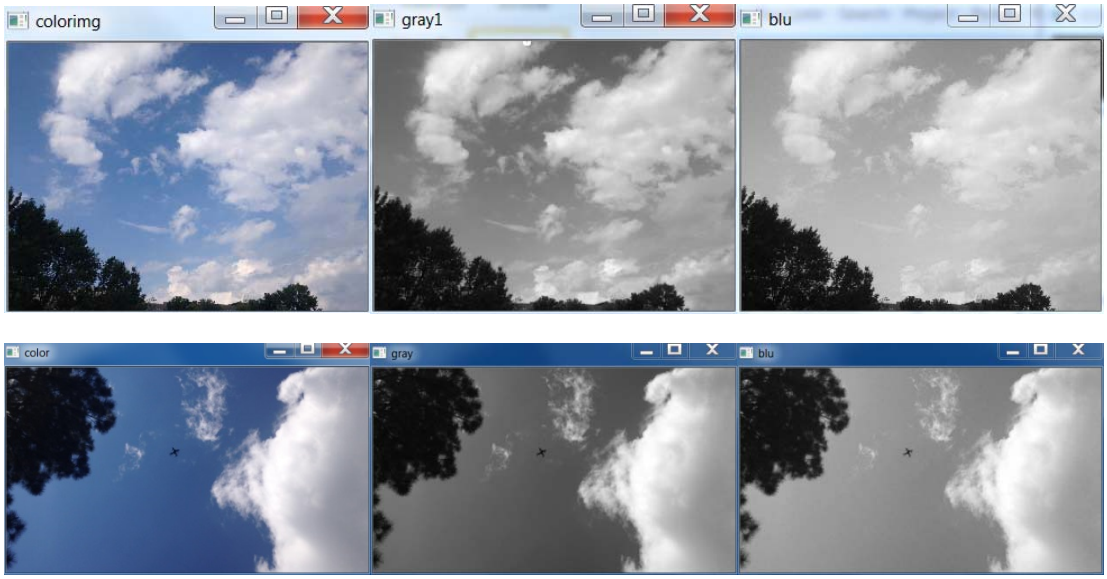

Figure 3. Reduction of the contrast of clouds by utilizing only the blue channel in an RGB image. *The left column shows the original color image, the middle column shows a conversion to grayscale and the right column shows the blue channel image.*

A common situation is seen in Figure 3 where the dark blue sky can create a high contrast edge with bright white clouds. We note that the blue sky will strongly excite the blue image channel (in the RGB color space) and because white strongly excites all three image channels, we can offset the brightness of white clouds by using *only* the blue image channel instead of converting the color image to grayscale. This is shown for two examples in Figure 3. While using only the blue channel can significantly reduce the background texture of the sky in most daytime situations, white clouds are often still brighter than the background blue. We can improve the situation further by reducing the intensity of pixels based on color. Since white contains significant intensity in the red channel whereas the blue does not, we can subtract an empirically-determined fraction (in this case, 0.4) of the red channel from the blue channel: $I(x,y) = I_B(x,y) - 0.4 \cdot I_R(x,y)$. Clearly, properly compensating for clouds in the background is much more complicated than a simple fixed-factor color-based intensity correction: the saturation level of the blue sky varies with the combination of viewing angle and sun angle (due to Rayleigh scattering), the intensity of white varies within a cloud, and there can be strongly differential illumination of the sky and clouds near sunset and sunrise. While we do not expect to completely "flatten" the intensity of the sky, these techniques can incrementally improve the conditions for tracking. In cases of cloudy, gray skies, no color-based correction is needed.

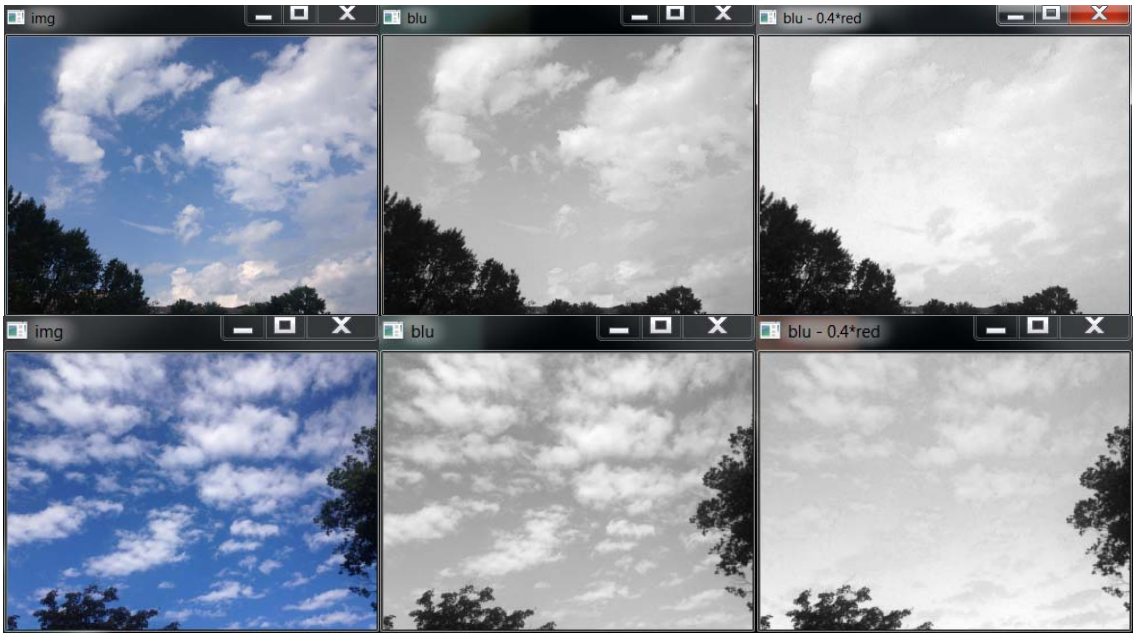

Figure 4. Further reduction of contrast for bright clouds. *Two example images (rows): Left column: original color image; middle column: blue channel alone; right column: blue – 0.4*red. Due to the overall reduction in intensity from this operation, an intensity normalization is also performed.*

Finally, a sigmoid function (e.g., the hyperbolic tangent, tanh(x)) can be applied to reduce contrast in the brighter (i.e., sky and clouds) and darker (i.e., ground) portions of the image. Selecting the 'threshold' intensity is, of course, critical and should be performed adaptively as lighting conditions change.

D. Vision-based "Isolated Blob" Detection

In this project, due to limited computational resources, we have elected to avoid explicit object recognition methods and will assume a relatively uniform visual background (i.e., the sky) to provide segmentation. Since the target object will vary dramatically in size with distance, we use a multi-scale image pyramid to find and track isolated blobs (i.e., flying objects in the sky; Figure 5). We apply a single 2-D difference-of-Gaussians (DOG) filter that favors dark objects against a bright sky. While the filter allows the tracking of bright objects against dark backgrounds, the peak values at the filter output will be lower. During the initial target search, all 5 spatial scales are used and full frame images are filtered. This occurs at approximately 4-5 frames per second.

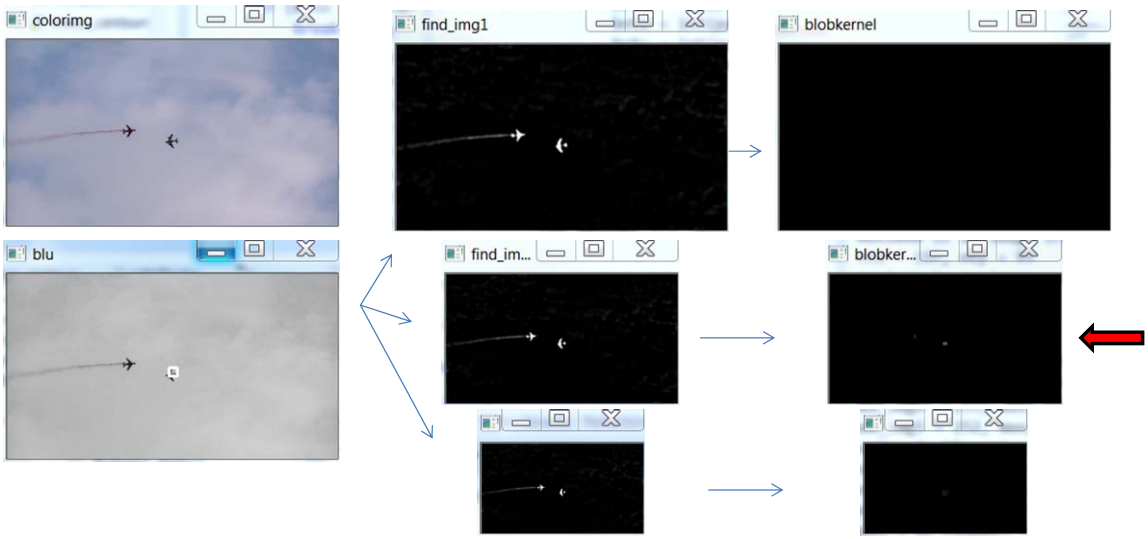

Figure 5. A multiscale image processing pyramid to detect isolated blobs. *Left column: original color image and the blue channel from the RGB image. Middle column: a fixed size DOG filter is applied at a 3 different scales during blob tracking. 5 scales are used initially when finding the target. Right column: a second filtering step identifies isolated blobs of texture and a max function across the image and across scales selects the location and scale of the best "blob".*

E. Tracking

Once a target has been identified from this initial process, a region of interest (ROI) is defined around the target and blob tracking is initiated. During tracking, the coordinates of the ROI for the next frame are computed using a prediction based on running estimates of position and velocity. During tracking, only the ROI is filtered, and only three spatial scales are analyzed: the scale at which the strongest signal was found in the previous frame and at one larger and one smaller scale, allowing tracking to occur smoothly across scale boundaries. Tracking can then be performed at 12-16 frames per second. If visual detection is lost, the model continues to predict its motion and persists in the visual analysis. After a timeout period without reacquisition, a full-frame search is initiated.

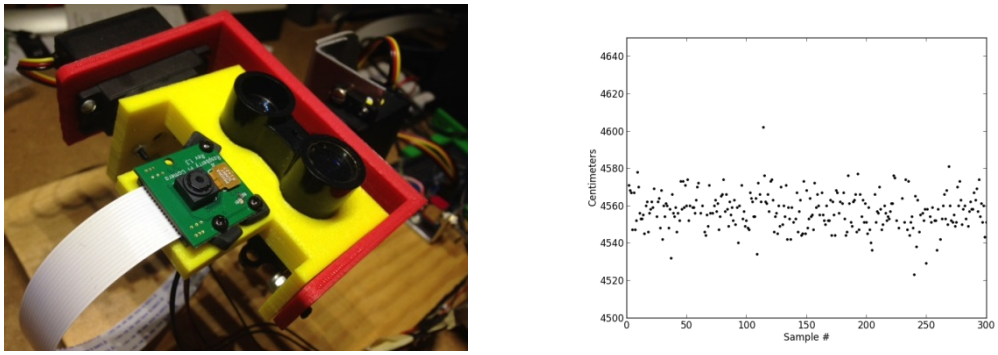

Figure 6. 2-D Gimbal and Lidar. *Left Panel: Photo of the Raspberry Pi camera and LidarLite™ sensor mounted together on a test gimbal. Mounting screws and rubber spacers allow the alignment to be adjusted. Additional alignment correction is performed in software. Right Panel: Lidar measurement of a flat white panel normal to the beam at 45.60 meters under warm, sunlit conditions. The standard deviation of the measurement was 9.7 centimeters or 0.21%. A similar measurement of a dark wooden panel at 7.19 meters exhibited a standard deviation of 1.35 cm or 0.19%. The transmitted IR laser beam was visualized using a CMOS imager without an IR-block filter and estimated to have a diameter of 4 cm when the LIDAR was placed at a distance of 720 cm.*

On the gimbal platform, the lidar beam was aligned with the center of the camera image (using the mounting screws) on a visual target at approximately 7 meters. After this mechanical alignment, software offsets were introduced to the tracking point to further refine the alignment. While the basic alignment between camera and lidar is good, at 10 meters, even 1.0 degree of divergence will produce an error of 17 centimeters, large enough to miss small aircraft.

-Δy	0	+Δy	
1/16	1/8	1/16	+Δx
1/8	1/4	1/8	0
1/16	1/8	1/16	-Δx

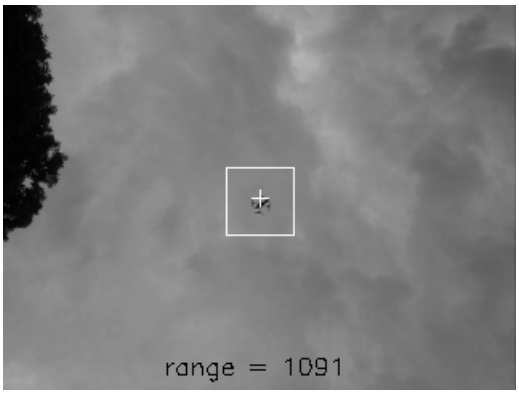

Figure 7. Visual Tracking and Lidar Ranging. *Left panel: Table of probabilities for introducing offsets (i.e., jitter) in targeting to increase the likelihood of obtaining a lidar reading, given alignment offsets. Right panel: Captured image of successful visual tracking and lidar ranging during ground-based tracking tests. The flying target is a Parrot AR Drone 2.0 with its outdoor hull. Range is given in centimeters.*

To deal with the residual misalignment after calibration, we introduce a discrete random jitter where an offset is added or subtracted to the x and y coordinates of the visually-derived tracking point with a probability given by the table in Figure 7 (left). Once a valid lidar reading is made, the random offset is held fixed until the lidar tracking is lost. The jittering resumes until the lidar is on target once again. Lidar measurements are more reliable at closer ranges due to the increased angular size of the target. In recent ground-based vision/lidar tracking experiments (see Figure 8), visual tracking is very reliable, particularly on days with clear skies. Lidar measurements are made when the gimbal has pointed the lidar directly at the target and thus measurements are most reliable when the target maintains a fixed velocity and the gimbal system is able to center the target in the camera's field of view. Currently, the lidar measurements are not used for flight guidance, and are only being used to trigger the net launcher and during autonomous, station-keeping, flight tests.

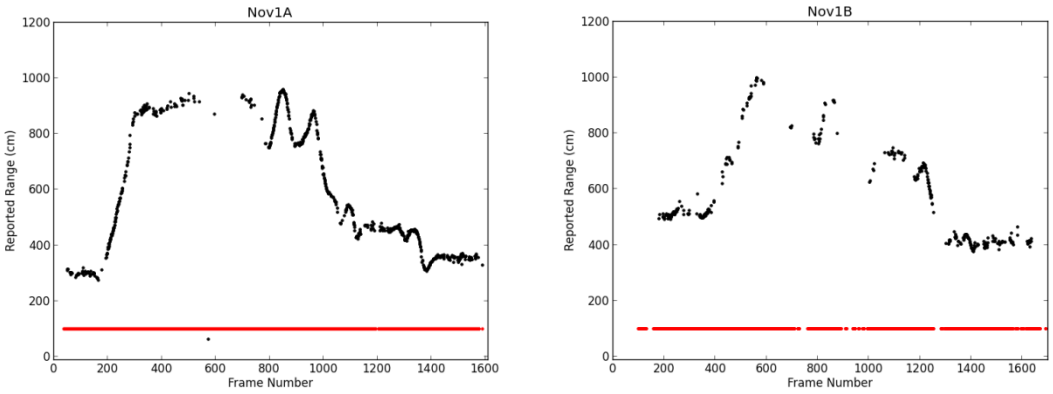

Figure 8. Lidar ranging of a maneuvering drone. *Bottom Row: Range data from two examples of tracking a Parrot AR Drone™ 2.0. Black dots indicate valid range measurements. Range measurements are only taken when a target is being tracked and the current ROI box includes the center of the image (i.e., lidar spot). Range measurement failures may occur if the laser misses the target (misalignment) or if there are communication errors (infrequent). Red dots at plotted at range 100 indicate frames where some target was being tracked. Although there are trees and clouds present, the target being tracked was always the aerial vehicle (except for the one point at frame 580 in the left panel, believed to be a falling leaf.) In the right panel, the target was successfully being tracked, but the ROI was often not centered due to a lag in gimbal tracking.*

F. Pursuit Algorithm

To guide the hexacopter to an intercept point, we have adopted a constant absolute target direction (CATD, or 'parallel navigation') strategy [8] inspired by the echolocating bat pursuing flying insect prey and by hunting dragonflies [9] and robber flies. By moving the vehicle to maintain a fixed absolute orientation vector (i.e., allocentric coordinates) connecting the pursuer and the evader, while continuing to ascend toward the target's altitude, the intercept vehicle continuously adjusts its orientation to achieve optimal time-of-capture performance, given the current vehicle speed. To implement CATD, the roll and pitch rates of the tracking gimbal are scaled and mirrored in the vehicle roll and pitch rates to change the vehicle's direction, holding the vector between interceptor and target relatively constant (see simulation results in Figure 9).

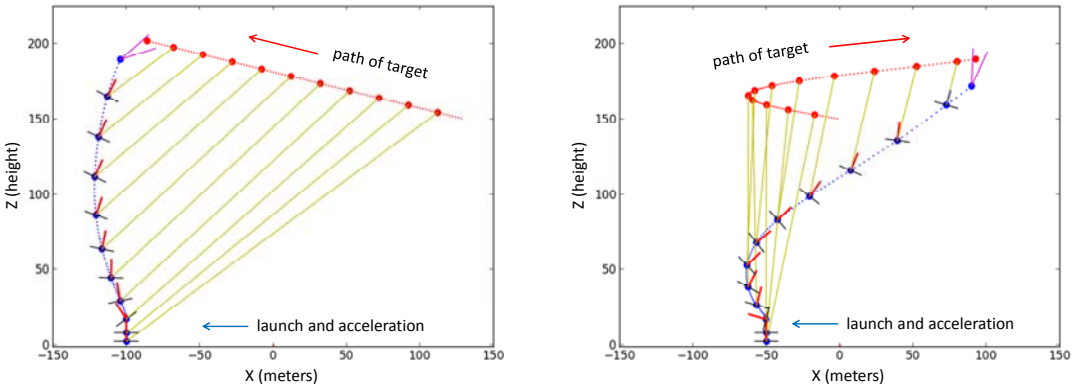

Figure 9. 2-D simulations of the 'constant-absolute target direction' pursuit algorithm driven by the visual sensor model. *Control inputs to the simulated interceptor vehicle are torque and thrust (in the direction of the vehicle orientation). Vehicle parameters were estimated. Initially, the vehicle is launched straight up and then tracking begins. Left: An approaching target is captured. Right: An evading target is captured. The steering algorithm uses only the time derivative of the angle between the interceptor and the target. The red line represents the direction of thrust.*

G. Net Launcher and Net Countermeasure

The net launcher is powered by commercially-available CO2 cartridges that are loaded into the bottom of the main launcher tube in front of a manually-armed spring. The mechanism is held in place by a servo motor trigger (right side of Figure 10, right panel). Upon release, the mouth of the cartridge is propelled toward a seal-puncturing pin at the top of the tube. The gases are then distributed into four launch tubes that are directed forwards. Inside the launch tubes are four weights that are attached to a square net on its four corners. The net is folded and loosely packed into the central cup. We have also been testing a small version of the net and net launcher in a Vicon [10] 3-D tracking system (Figures 11 and 12) to measure the dynamics and performance of the net.

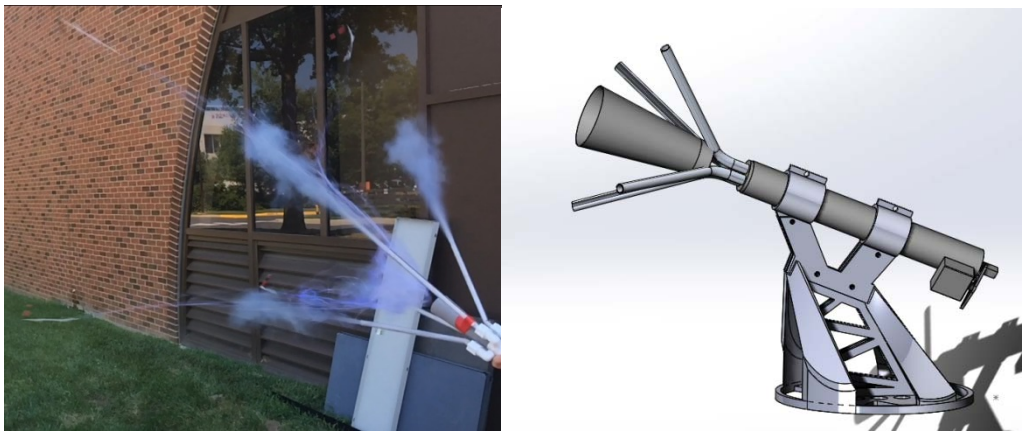

Figure 10. Net Launcher Design. *Left panel: A snapshot of a net launch. The left-side corner weights can be seen at the left edge of the photo. Right panel: CAD drawing of the net launcher. The 4 corner-weight launch tubes and the central net cup can be seen. The main tube contains the CO2 cartridge spring and seal-puncture pin. The spring is released by a servo motor controlled by the Raspberry Pi 2.0 computer. The launcher is constructed from four polycarbonate tubes, one central aluminum tube and 3D printed parts.*

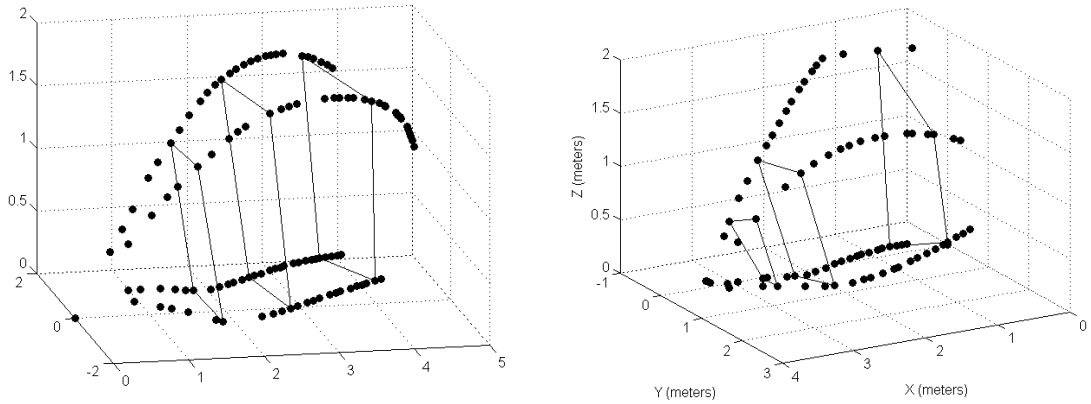

Figure 11. Net Trajectory *A smaller 2m x 2m net tested using a 4-camera Vicon 3-D tracking system dt=15.6 ms. This figure shows the starting position and 40 time points (624 ms of the trajectory) from the launch time. Right panel: 26 time points (406ms of the trajectory) from the launch time.*

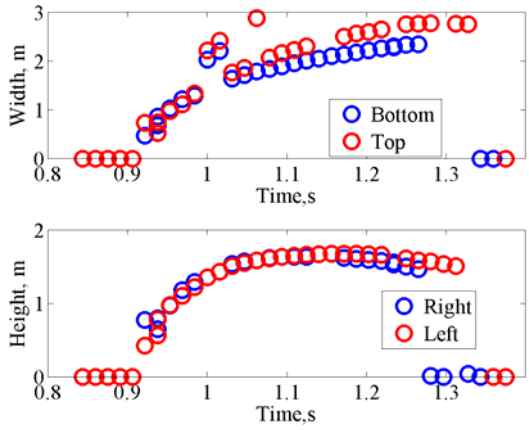

Figure 12. Net opening symmetry. *Example net launch, showing vertical and lateral net opening speeds are consistent. After 0.2 seconds, the net is 1.9 m x 1.9m square but continues to expand laterally to 2.9m*

In low altitude indoor testing on quadcopters with exposed propellers, we have consistently demonstrated that the propellers will rapidly draw in the net material until the propellers are forced to a halt due to entanglement.

H. Intercept Vehicles

We have constructed two vehicles: a primary intercept vehicle with the net launcher and a smaller, secondary intercept vehicle for pursuit algorithm development. The primary intercept vehicle is built upon the DJI F550 hexacopter [11] (Figure 13, left) using the 3DR Pixhawk Autopilot Controller. The motors are configured with 9 inch propellers. The vehicle has a payload capacity of ~1.5 kg and is currently measured at ~ 4kg fully loaded. The system uses a 5000 mAh 4s lithium-polymer battery and is estimated to have a flight time of 5 minutes. The secondary intercept vehicle is based on the FT Versacopter V2 [12] (280mm class; 0.8 kg) racing quadcopter with a microservo-based, gyro-stabilized gimbal and does not carry the lidar nor the net launcher (Figure 13, right).

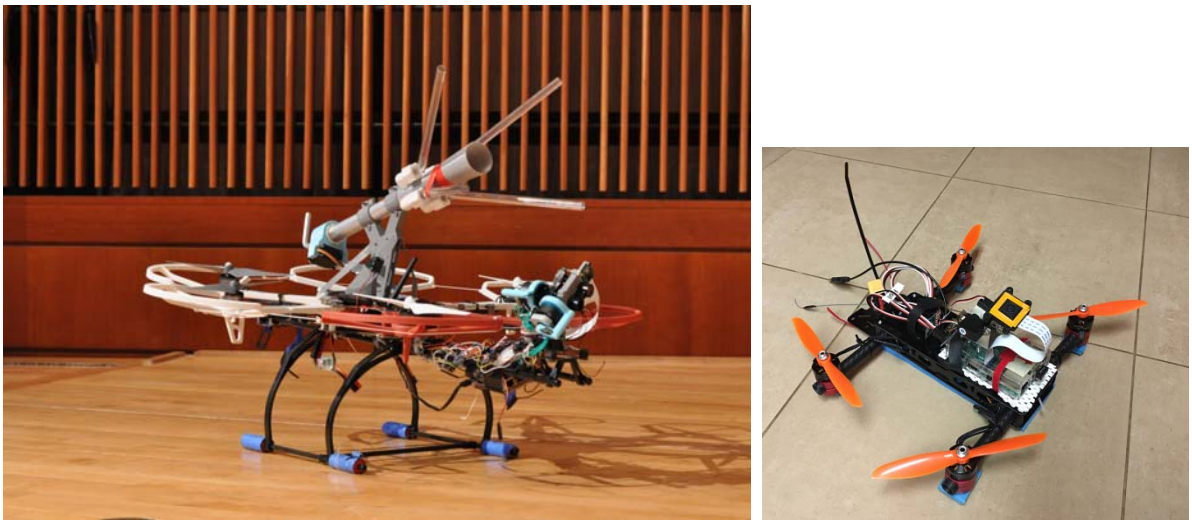

Figure 13. Two intercept vehicles. *Left: Photo of the primary intercept vehicle (hexacopter) without the net loaded. Right: Photo of the secondary vehicle used in algorithm development. The orange square is the gimbal-mounted camera.*

III. Testing

To date, we have demonstrated outdoor ground-based vision/lidar tracking of quadcopters and have demonstrated indoor autonomous station-keeping using vision and lidar. We have demonstrated the takedown of quadrotors in indoor flight using the net launcher with both intercept and target vehicles flown manually (see Figure 14). Delays in acquiring FAA permission to test and demonstrate the system outdoors have limited the team to indoor testing.

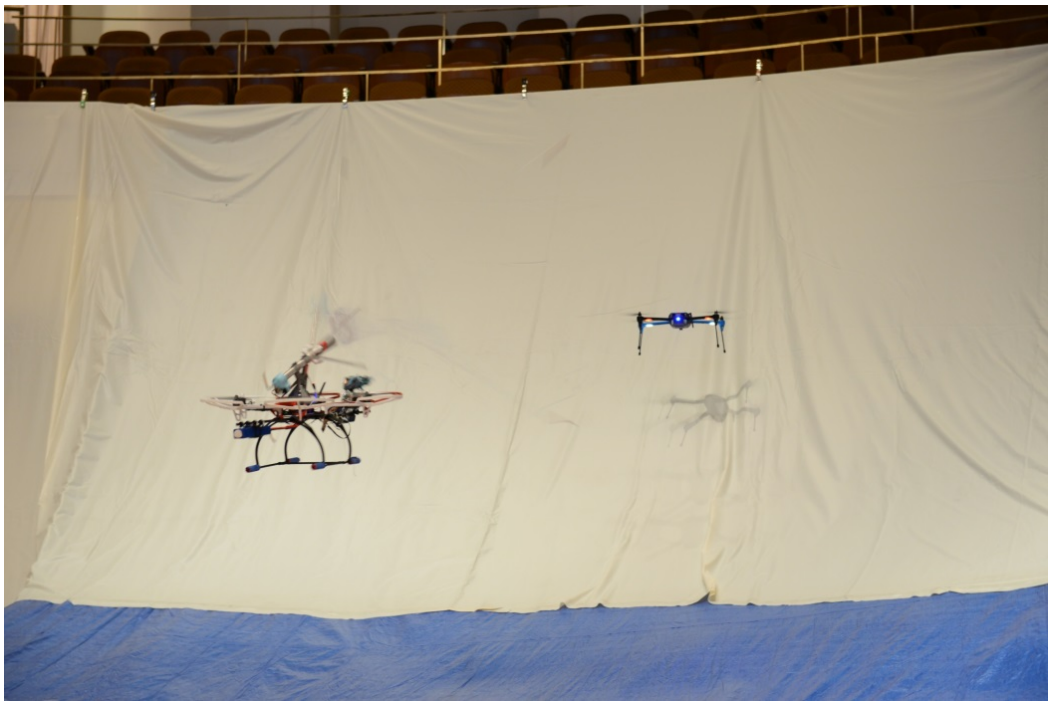

Figure 14. Manual Capture. *The counter-sUAS intercept vehicle (left) firing a net at a 3D Robotics Iris quadcopter in front of a large white sheet on stage at the University of Maryland Clarice Smith Performing Arts Center. The net was manually fired and the target quadcopter was entangled and fell to the ground.*

Acknowledgments

This work was supported in part by Unconventional Concepts Inc. (Mary Esther, FL) and by seed funding by the University of Maryland (College of Engineering). We thank the University of Maryland Clarice Smith Performing Arts Center for the use of one of their stages for our mid-project demonstration event. The photo in Figure 14 was taken by Ms. Jennifer Rooks.

References

[1]OpenCV ver. 2.4 – https://opencv.org or download from: http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-2.4.10.zip/download

[2]Raspberry Pi 2 Model B: visit https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[3]Raspbian Wheezy available via download from: https://www.raspberrypi.org/downloads/ or visit https://www.raspbian.org/

[4]Pixhawk Autopilot (3D Robotics) http://store.3drobotics.com/t/pixhawk

[5]Adafruit 16-channel, 12-bit PWM/Servo Driver Board (PCA9685) – https://www.adafruit.com

[6]LidarLite Sensor – https://pulsedlight3d.com

[7]Raspberry Pi Camera Module (5 megapixel) - https://www.raspberrypi.org/products/camera-module/

[8]Ghose, K., Horiuchi, Krishnaprasad, P. S., and Moss, C., (2006) "Echolocating Bats Use a Nearly Time-Optimal Strategy to Intercept Prey", PLoS Biology 4(5): e108, DOI: 10.1371/journal.pbio.0040108

[9]Oldberg, R. M., "Visual control of prey-capture flight in dragonflies", Curr Opin Neurobiol. (2011), DOI: 10.1016/j.conb.2011.11.015

[10]Vicon Motion Capture – https://www.vicon.com

[11]DJI F550 Flame Wheel – https://store.dji.com

[12]Flite Test Versacopter V2 – https://store.flitetest.com