Implementation of a Sensor Guided Flight Algorithm for Target Tracking by Small UAS

Gaemus E. Collins[a], Chris Stankevitz[a], and Jeffrey Liese[b]

[a] Toyon Research Corporation, 6800 Cortona Dr., Goleta, CA, 93117, USA;

[b] Department of Mathematics, California Polytechnic State University, San Luis Obispo, CA, 93407, USA.

ABSTRACT

Small fixed-wing UAS (SUAS) such as Raven and Unicorn have limited power, speed, and maneuverability. Their missions can be dramatically hindered by environmental conditions (wind, terrain), obstructions (buildings, trees) blocking clear line of sight to a target, and/or sensor hardware limitations (fixed stare, limited gimbal motion, lack of zoom). Toyon's Sensor Guided Flight (SGF) algorithm was designed to account for SUAS hardware shortcomings and enable long-term tracking of maneuvering targets by maintaining persistent eyes-on-target. SGF was successfully tested in simulation with high-fidelity UAS, sensor, and environment models, but real-world flight testing with 60 Unicorn UAS revealed surprising second order challenges that were not highlighted by the simulations. This paper describes the SGF algorithm, our first round simulation results, our second order discoveries from flight testing, and subsequent improvements that were made to the algorithm.

Keywords: Unmanned Air Vehicles, Search, Tracking, Sensor Management

1. INTRODUCTION

The seemingly easy problem of following a moving target to maintain constant surveillance turns out to be somewhat challenging for very small fixed-wing UAS. These UAS typically fly at low speeds and low altitudes, and are dramatically hindered by wind and environmental conditions. The range of permissible flight speeds for such platforms is often very narrow (e.g. minimum flight speed $\approx$ 15 m/s, maximum flight speed $\approx$ 20 m/s), so the UAS may have to perform "S"-turns or loops to stay nearby the target.

Sensors onboard small UAS may have additional limitations. The sensor may be fixed (e.g. side-look only), or the sensor gimbal may have a limited range of motion (e.g., biased to one side of the aircraft); these limitations are highlighted in Figure 1. Zoom capability may also be missing from these systems.

In addition, obstructions such as buildings, trees, or the UAS wings and landing gear may occasionally block the UAS's clear line of sight to the target. These challenges are summarized in Figure 2

"Sensor Guided Flight (SGF)" is understood as the ability for a UAS's sensing system, primarily an imaging system, to automatically request a platform position and attitude that maximizes sensor performance. SGF includes the ability to monitor viewing conditions for sensor tasks, assess whether the sensor system parameters and platform position and attitude are optimal for those viewing conditions, and compute preferred parameters and platform state for best quality imagery for those viewing conditions. This is a particularly important capability for small UAS to mitigate shortcomings of the platform and sensor such as narrow flight speed envelope, limited gimbal range, and lack of zoom.

2. OVERVIEW OF THE SENSOR GUIDED FLIGHT ALGORITHM

Toyon has developed an SGF algorithm[1,2] that optimizes sensor-to-target viewing geometry over possible future actions of the UAS and sensor. Our algorithm uses a joint UAS routing and sensor optimization technique that couples the platform and sensor control.

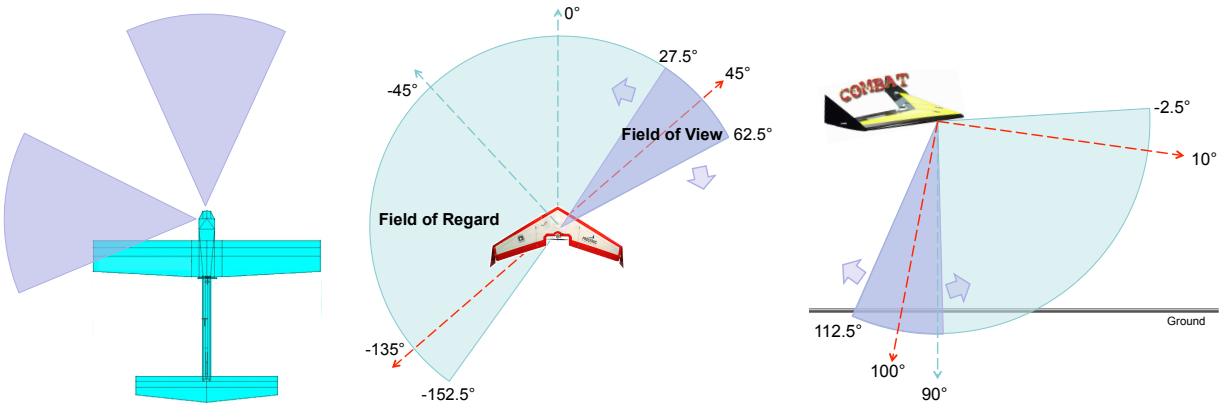


Figure 1. LEFT: Raven RQ-11 UAS has a fixed front-look and fixed side-look sensor, but no sensor gimbaling. MIDDLE and RIGHT: Unicorn UAS has a gimbaled sensor with limited range, and biased to the left.

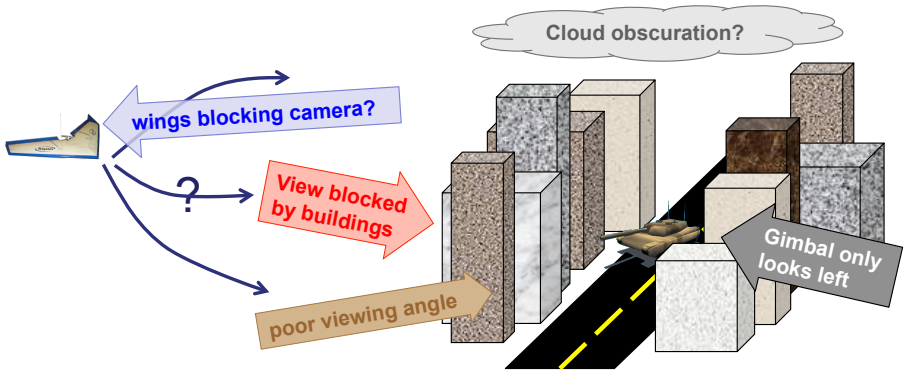


Figure 2. Small fixed-wing UAS face many ISR challenges

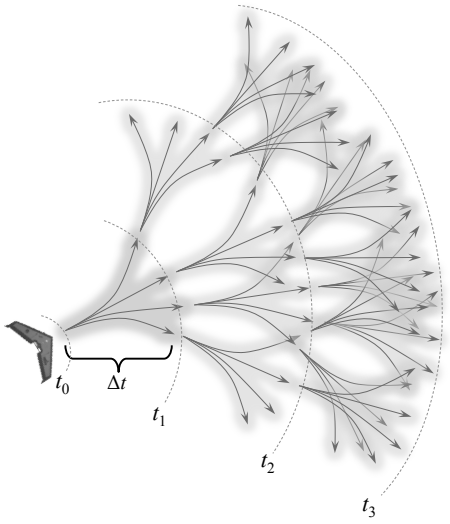


Figure 3. The SGF algorithm will consider several dynamical models for the platform over each time step Δt. Each dynamical model falls within the platform capabilities, as measured by our 6-DOF UAS model. The algorithm then considers what the sensors can detect along each path. This process is repeated for several steps into the future. The optimal sequence of maneuvers will be selected.

First, the algorithm employs a 6-DOF UAS flight model to generate highly accurate candidate UAS paths for several future time horizons. These paths provide a variety of future routing options while adhering to the platform dynamics and accounting for wind(Figure 3).

Next, the algorithm models the sensor coverage along each candidate path. This accounts for

- the roll, pitch, and yaw of the aircraft,
- the sensor gimbal dynamics and constraints,
- the sensor (instantaneous) field of view (we are not currently modeling zoomable sensors, but this is planned for future development).

To minimize complexity, this modeling is performed at discrete "sensing points" along each candidate path – typically taken at 1-second increments, but this parameter is adjustable.

For UAS with fixed sensors, candidate looks are generated from each sensor at each sensing point. For UAS with gimbaled sensors, the algorithm will perform a second optimization at each sensing point to compute the optimal sensor parameters (i.e. gimbal orientation). For target tracking applications, we assume that the optimal gimbal orientation is the one pointing the sensor directly at the target position estimate (as provided by an external target tracker, GPS, or other intelligence source). For search applications, the sensor parameter optimization can be more complex. A collection of candidate tasks at each sensing point can be generated by partitioning the azimuth and elevation ranges of the sensor Field of Regard (FOR), or by targeting specific points on the ground (like the track locations).

Once candidate future control actions (paths and sensor parameters) have been generated, the SGF algorithm will compute an expected reward or score for each control sequence. Our score function was designed to measure the quality of the imagery collected for ISR. It is primarily based on the estimated resolution of ground targets (number of pixels on target) in the imagery, but includes additional components. This is discussed in more detail in Section 3.

In the final step of the algorithm, the control sequence for the UAS path and sensor is computed as an aggregate score over several time steps, $[t_0, t_N]$. The score for the path+sensor control sequence yielding the highest expected score will be selected as the optimal sequence. In the style of model predictive control, the control actions corresponding to only the first step $[t_0, t_1]$ of the optimal sequence will be selected for execution. A new optimal control sequence for the remaining time steps $[t_1, t_N]$ will be re-computed as the UAS nears t_1 and collects new information.

3. ALGORITHM IMPLEMENTATION AND TESTING

The SGF algorithm was coded in C++ within a larger Toyon software package called CDAM. CDAM enabled simulation testing of SGF using a 6-DOF aircraft model, simulated sensors, and a virtual environment. We set up a simulated mission in Camp Roberts, CA, in which one UAS is attempting to follow and "bird-dog" a moving target to maintain persistent surveillance. We used an aircraft model for the Unicorn UAS (see Section A), with SGF generating waypoint commands that were sent to Aviones, a Unicorn flight emulator.[3] The UAS state from Aviones was used to update the SGF Unicorn model. The target moved along a road network with random speeds. A simulated sensor co-located with the UAS sent measurements to a software tracker (see Section B) that would fuse the measurements into a target track state estimate. This track position was used by SGF to route the UAS and task the sensor.

Initial testing in simulation looked good. The UAS seemed to follow the target and image it consistently. We had not yet set up Measures of Effectiveness (MOEs) to calibrate the algorithm performance.

Further author information:
e-mail: gcollins@toyon.com, Telephone: 1 (805) 968-6787

3.1 Preparation for First Flight Test

In preparation for our first flight test of the SGF algorithm, we began testing the algorithm in a hardware-in-the-loop (HWIL) configuration. Our HWIL configuration enabled the SGF algorithm to send waypoint commands to an autopilot (AP) sitting on the desk in the office. The AP serial port was connected to the Aviones flight simulator, so the AP thought that it was flying a real UAS. This HWIL testing highlighted several key issues with our implementation of the SGF algorithm:

(1) Our software sometimes did not run fast enough to produce good performance in hardware;
(2) The various parameters controlling the performance of the SGF algorithm could not be adjusted "on-the-fly" (without re-starting the software).

These discoveries prompted us to conclude that the CDAM software package came with too much overhead for the SGF algorithm. While CDAM provided a valuable simulation environment, it was not well-suited to real-time hardware testing. We decided to write a new stand-alone software app for the SGF algorithm. The new app was written in C++ and streamlined for speed. It enabled on-the-fly modification of critical parameters like step size, number of steps to look ahead, how often to re-evaluate the target state, and various parameters affecting the SGF score function. A screenshot of the SGF app appears in Figure 4.

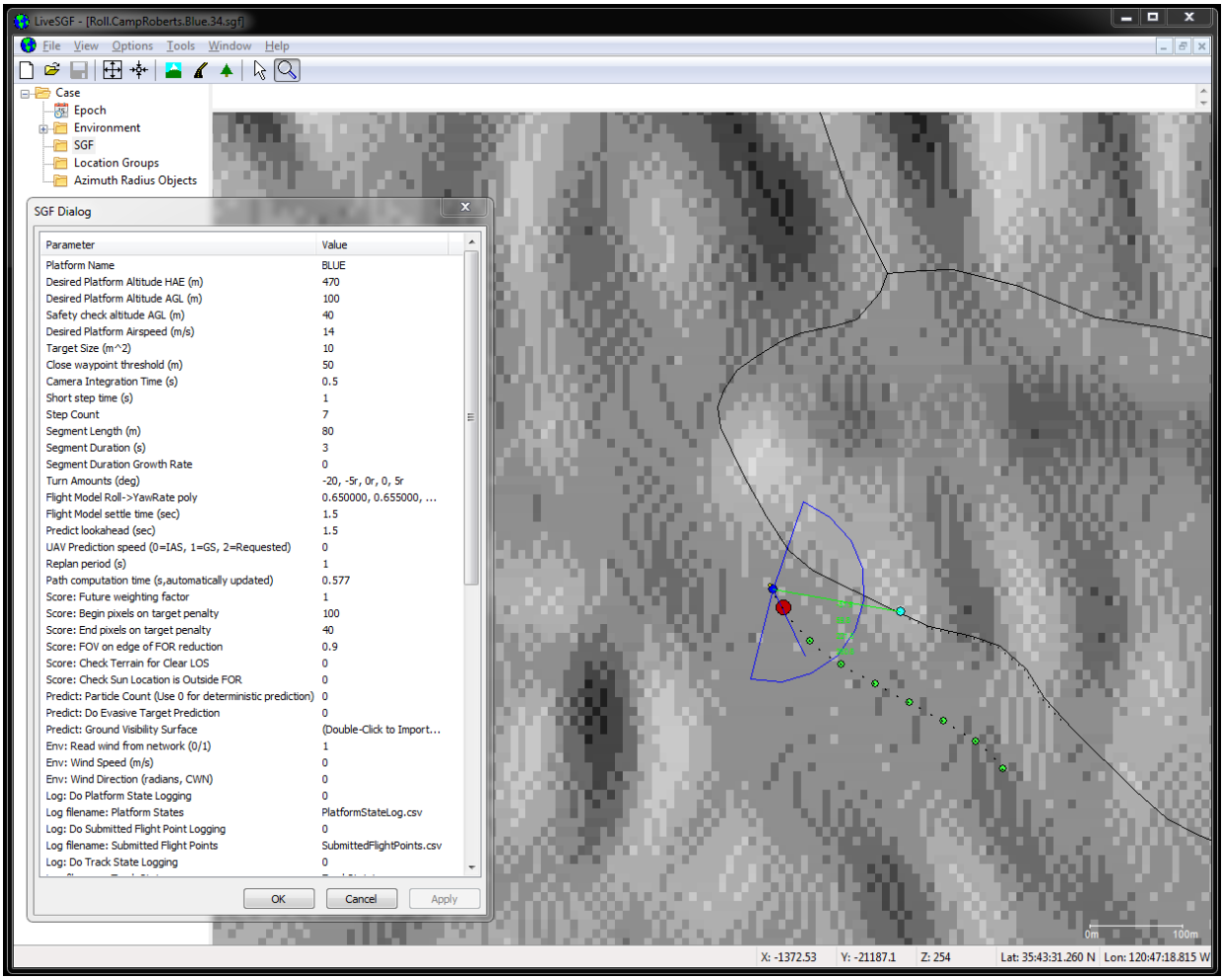


Figure 4. Toyon's SGF application showing parameter editing dialog box

3.2 Initial Flight Testing

With the new SGF app operational, we headed to Camp Roberts for preliminary flight testing of SGF. Flight testing of UAS control algorithms is a critical step in the development process, and always highlights features and issues with the algorithms that were not identified in development or simulation. We tested in light to moderate wind conditions (0 to 7 meters/sec wind speeds, including thermals and gusts), and moderate to high temperatures (50° F to 110° F).

In our first flights, the algorithm did a reasonable job of keeping the UAS close to the moving target, and keeping the video sensor pointed at the target. Preliminary results for a 20-minute flight test of the SGF algorithm are displayed in Figures 5–8. In this test the target motion was random along the road network, with the target typically moving 5–10 m/s. The UAS flight speed was constant at 14 m/s (airspeed).

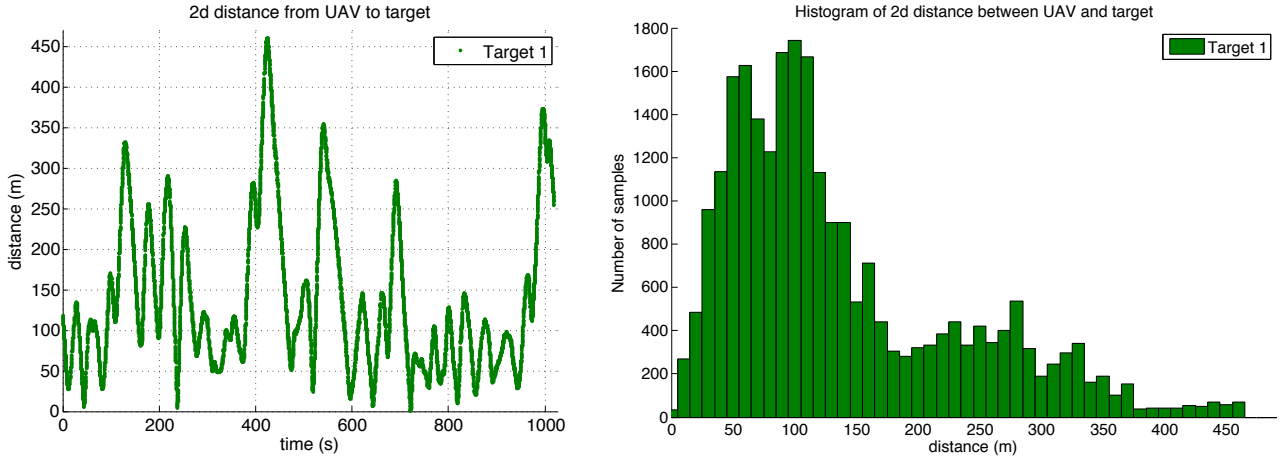


Figure 5. LEFT: 2D projected planar distance between UAS and target over 20 minutes. The ideal distance of 0m is impossible to maintain because the target is often moving slower than the UAS; but high quality imagery on the target can be obtained with inexpensive video cameras by keeping the 2D UAS-to-target distance as small as possible. RIGHT: Histogram of 2D projected planar distance between UAS and target over time shows that the algorithm mostly keeps the UAS within 200m of the target.

In this initial flight testing, reasonable paths for the UAS were chosen, but the imagery obtained by the UAS running SGF was not ideal. We identified several implementation issues and areas for improvement:

- The autopilot is sensitive to frequent changes in its waypoint queue, and will produce erratic and unpredictable behavior if waypoints in the queue are modified too frequently.
- Waypoint navigation for UAS control has inherent drawbacks that limit the performance envelope of any control algorithm.
- The objective function used by SGF to value candidate future actions does not always render the optimal viewing geometry on the target.
- The imagery often contained significant motion and jitter.
- The UAS was often too far away from the target to obtain reasonable resolution (# of pixels on target).
- Steep roll angles commanded by the autopilot waypoint controller often put UAS wings or landing gear in the camera Field of View (FOV), producing false detections in the image tracking software. A lack of reliable detections or an excessive number of false detections makes it nearly impossible to track a target.

3.3 Waypoint Navigation Drawbacks

3.3.1 Waypoint Queue Sensitivity

Our initial implementation of SGF produced a queue of waypoints and then uploaded them to the autopilot on the UAS. On each SGF re-plan event, a new waypoint queue was uploaded, overwriting the existing (previous) queue. Unfortunately the autopilot did not respond well to this implementation. When we would overwrite the

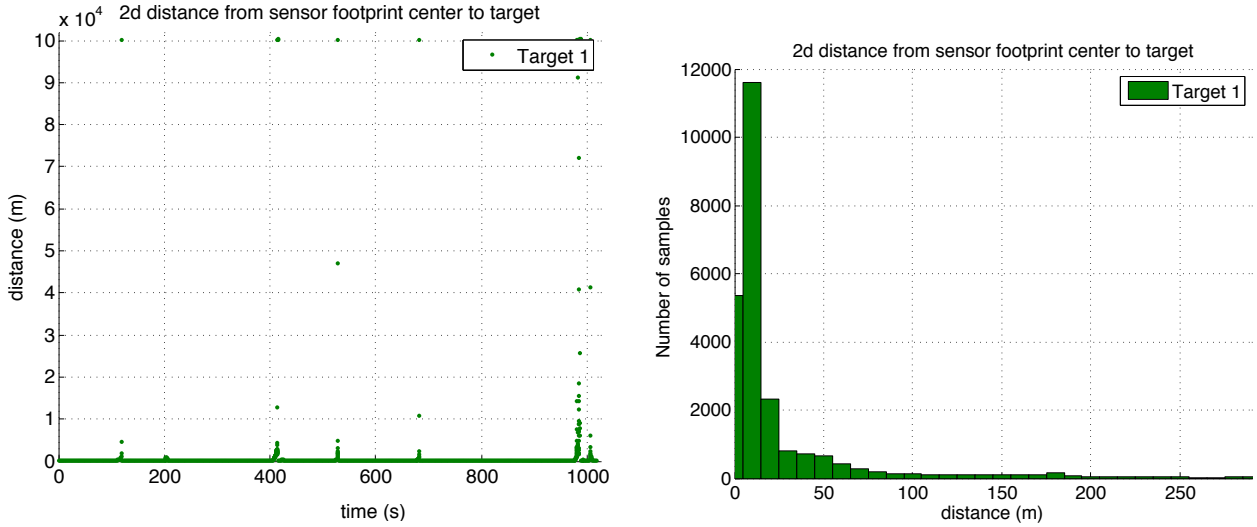


Figure 6. LEFT: 2D distance between sensor boresight (intersected with the Earth) and the target, plotted over time. This plot was generated from data showing the direction that the UAS AP was attempting to point the sensor, so it may be a bit optimistic. Note that the y-axis is $y * 10^4$. RIGHT: Histogram of 2D distance between sensor boresight and the target shows good performance of the sensor targeting portion of the SGF algorithm.

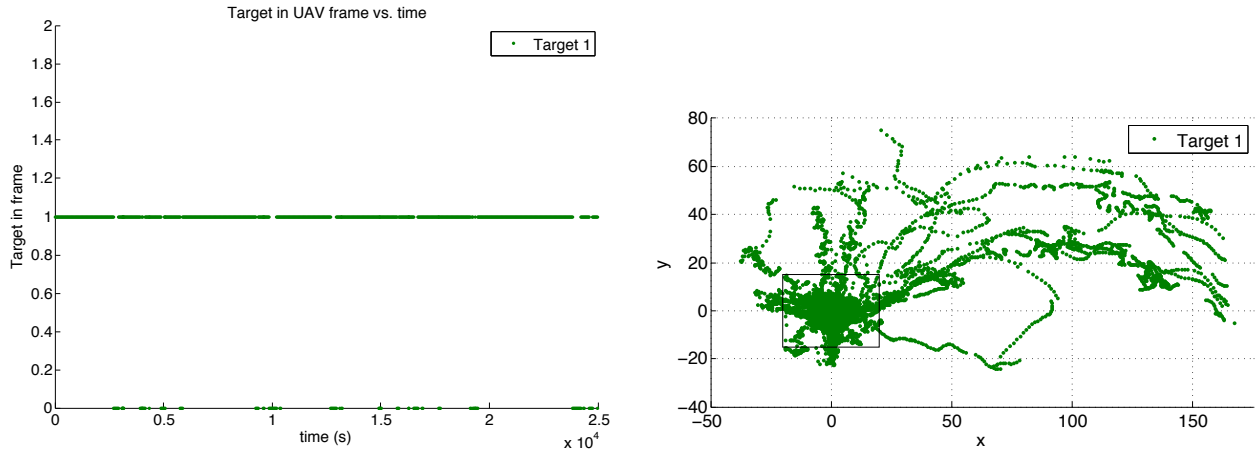


Figure 7. These figures show that the target stayed within the camera field of view (FOV) about 90% of the time. When the target strayed from the camera FOV, it was biased to the upper right corner of the camera view.

existing queue of waypoints, the plane would often make sharp turns in unpredictable directions before resetting to the uploaded waypoint path. This behavior prevented the SGF algorithm from reliably predicting future positions of the UAS, and deteriorated performance.

The waypoint queue sensitivity problem was solved by a modification to the SGF software. Instead of overwriting waypoint paths in the AP memory, we changed the SGF-UAS interface to only *append* waypoints to the end of the previously sent path. In particular, we discovered that new waypoints should be appended to the AP path no later than one waypoint in advance, as shown in Figure 9. Failing to adhere to this anticipation constraint would produce similar unpredictable AP behaviors as described in the previous paragraph.

3.3.2 Imprecise Control

Using waypoint routing, it is impossible to precisely control the roll or trajectory of the aircraft. The aircraft will get "close enough" to each waypoint to "check it off," and will continue to the next waypoint, as shown in Figure 10. If the aircraft can not get close enough to a waypoint, it will circle around to try it again. Both of these behaviors generate problems for SGF performance.

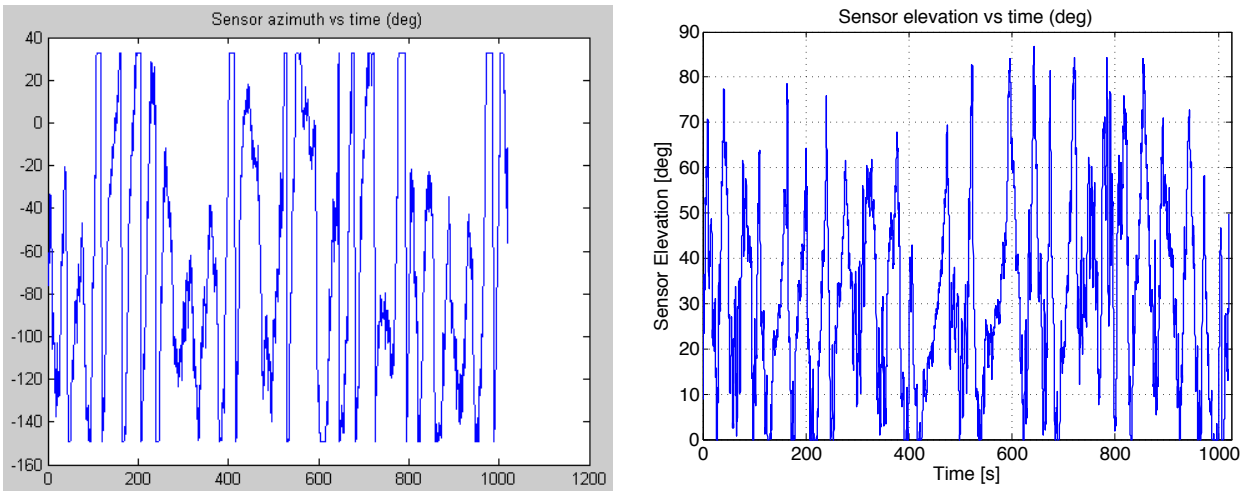


Figure 8. These figures show the gimbal azimuth and elevation commands required to keep the camera pointed at the target. For automatic image processing algorithms, we would like to see these numbers changing slowly (i.e., the first derivative of this function stays small in absolute magnitude) because fast gimbal movements can disrupt the foreground/background models of the image processor.

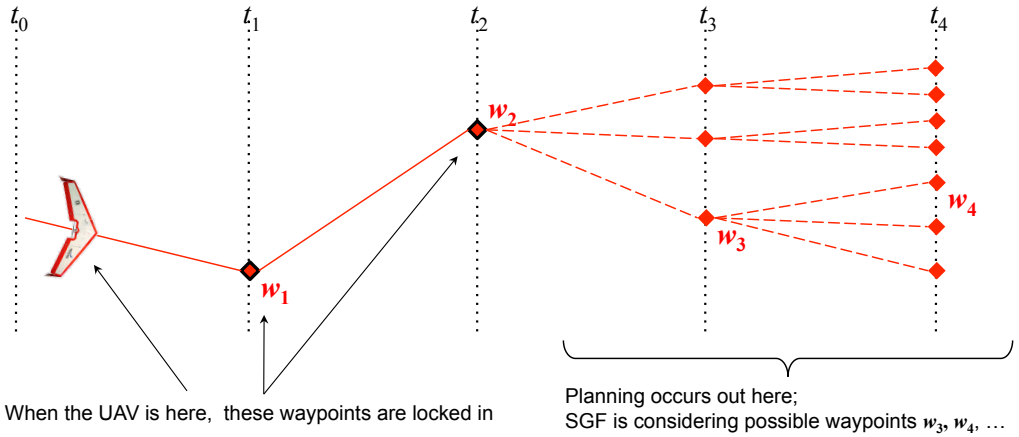


Figure 9. Waypoints must be appended to the UAS's path by SGF at least one waypoint in advance

3.3.3 Suboptimal Loiters

One particular situation that is limited by waypoint routing is viewing a stationary target. The optimal viewing geometry from a moving aircraft to a stationary target is achieved with the aircraft circling the target with a mild roll and at a distance that was close enough to see the target, as shown in Figure 11. With this geometry, the target should never leave the field of view.

However, when using waypoint navigation, SGF can't instruct the plane to fly a circular path – the best it can do is to send a polygonal path that encloses the target. The plane will successively perform sharp turns and then straighten out to fly around the polygon. While the plane is flying straight, we have no problem seeing the target, but while the plane is performing a steep bank, often times the target is blocked by the wings or landing gear of the UAS. The polygon can not be broken up into many short segments because the waypoints must be separated by a minimum distance (about 70 meters).

3.3.4 Wings and Landing Gear in the Sensor Field Of View

One of the most significant problems encountered by SGF during flight testing was the wings and landing gear appearing in the sensor FOV. Most commonly, the wings would appear when the plane was performing a steeply banked turn. These steep bank angles are commanded by the UAS autopilot during waypoint navigation; when

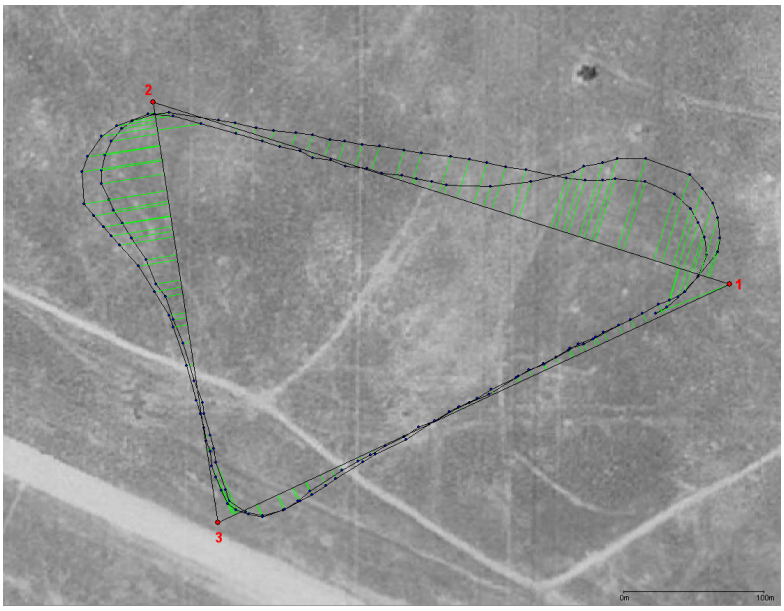


Figure 10. We measured the deviation between the exact waypoint path and the UAS flightpath. For 2 loops around the triangular flightpath shown above, the average deviation was 17.6 meters, with a maximum deviation of 67 meters.

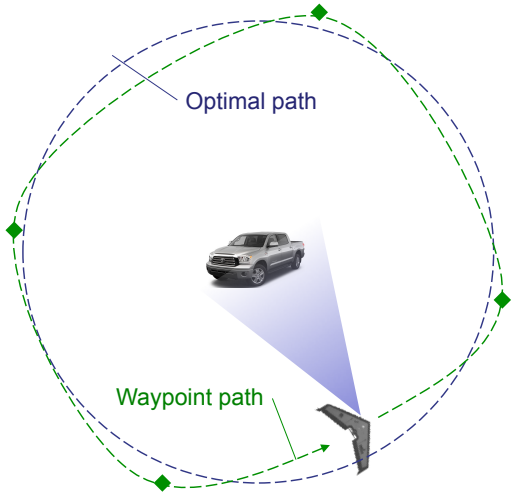
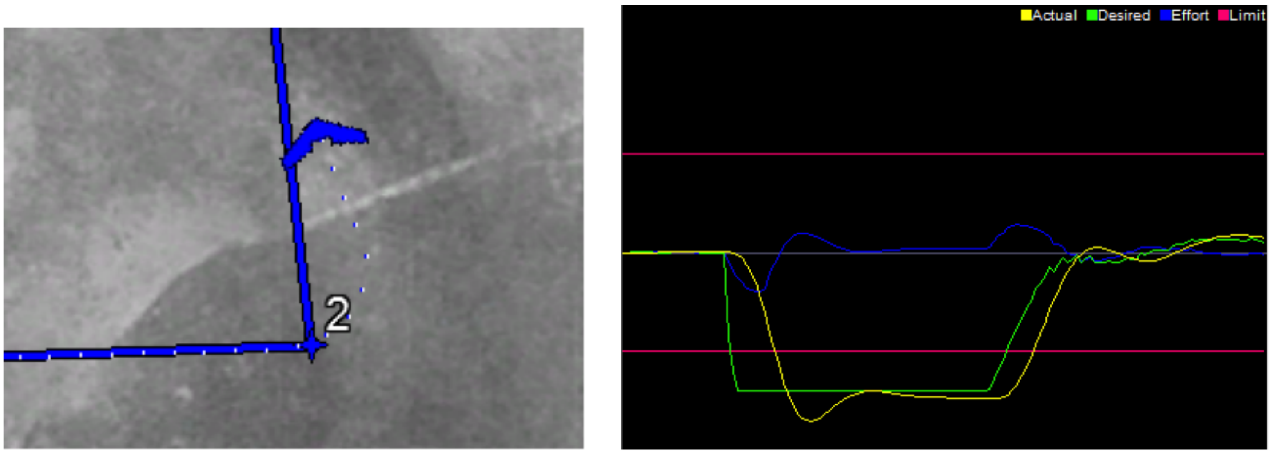


Figure 11. Loitering around a stationary target with waypoint navigation can not be done optimally

the autopilot reaches a particular waypoint and begins its progress toward the successive waypoint, it chooses a path (and bank angle) to reach the successive point as quickly as possible. Figure 12 is a screenshot of a sharp turn and steep bank angle ($\approx 30°$) commanded by the UAS autopilot waypoint controller. Figure 13 shows the resulting landing gear in the camera FOV.

3.4 Roll Control SGF

Based on the observations in Sections 3.2 and 3.3, we decided to add capability to the SGF-UAS interface to enable SGF to route the UAS using roll commands instead of waypoints. This new version of SGF was tested in simulation, and initial results looked very promising. Figure 14 shows the path and roll of a UAS which was originally at a roll of 0 degrees, then commanded to roll of -20 degrees for 3 seconds and then back to 0 degrees. Notice that this left hand turn was made without a steep bank.



(a) (b)

Figure 12. (a) The UAS waypoint path connecting waypoints 1, 2, and 3 creates a sharp turn at waypoint 2. (b) The roll plot shows the UAS banking with its maximum 30° roll angle to navigate toward waypoint 3 as quickly as possible.

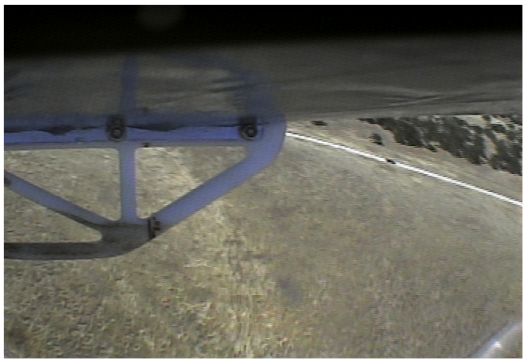


Figure 13. UAS wing and landing gear in camera FOV. The landing gear can be accidentally detected by automatic image processing software and confused with a real target in track.

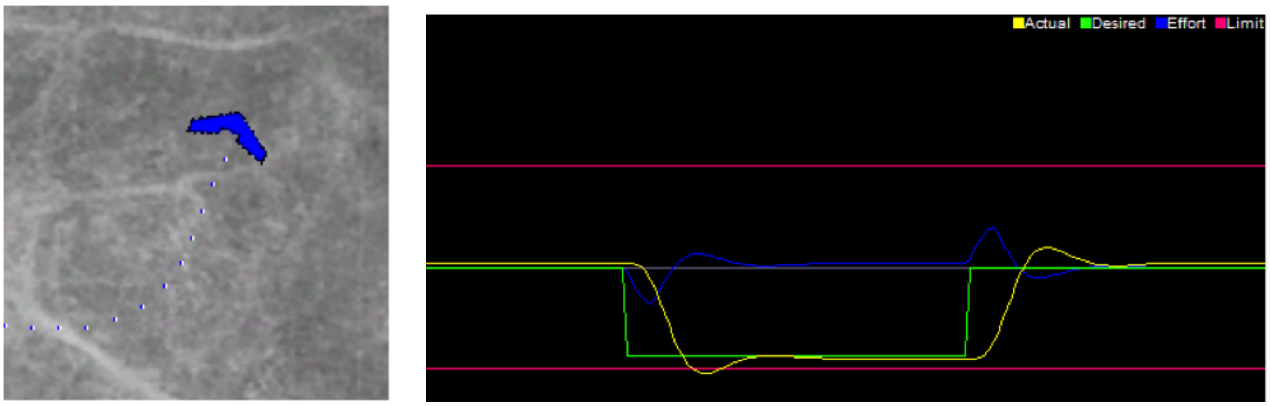


Figure 14. A -20° turn commanded by the SGF RCN version. These shallower roll angles will typically keep the imagery free of landing gear.

3.5 SGF Objective Function

The roll-control version of SGF solved many implementation problems, but subsequent flight tests showed that issues still remained. When examining the imagery that we obtained from various experiments running SGF, we noticed that even though SGF produced very reasonable paths for the UAS, the imagery obtained by the UAS running SGF was not excellent. The imagery often contained too much motion and jitter, and was often

shot from too far away from the target for our image tracking software to make accurate and reliable detections. Based on these observations, we put additional work into the SGF objective function, and adjusted parameters affecting SGF performance.

- The SGF routing objective function was modified to more realistically model environmental tracking conditions.
- To get better imagery with more pixels on target (POT), we purchased several new camera lenses which provide a higher level of zoom.
- We added the ability to simulate using the new lenses.

To determine the path SGF will choose to fly, the algorithm first creates candidate paths, then scores the viewing geometries along the path using an objective function. The purpose of the objective function is to produce a score $s(A)$ given particular viewing geometry A. For any geometries A and B, we desire that $s(A) > s(B)$ if the UAS will provide better ISR using the geometry from situation A than that of B.

3.5.1 Prior Objective Function

When our SGF scoring function was first created, a basic set of parameters were used to produce a direct effect on the score of a particular geometry:

- Estimated sensor-to-target distance (meters).
- Position of target in sensor FOV and FOR (degrees).

When planning a route for the UAS, SGF estimates the state of the UAS and target at various points along the route, and computes these parameters.

A target that is larger in the image will provide better resolution (more pixels on target), and a better detection threshold in our image processing software. The score function was therefore set up to minimize sensor-to-target distance.

Viewing a target in the center of the field of regard provides a cushion for maintaining the target in the FOV even if the target performs an unpredictable change of motion. Our score function returned higher values for sensor-to-target geometries that kept the target near the center of the FOV and FOR.

3.5.2 New Objective Function

After our testing in July 2010, it became clear that accurately quantifying the objective function would be a necessity to improve tracking performance. The software team consulted with several image plane video tracking (IPVT) experts who generated a new list of quantities that directly affect the ability to detect and track a target:

- Maximum frame-to-frame motion of background points for frame registration
- Minimum number of pixels (area) on target to generate a detection
- Minimum number of frames to generate a detection
- Imagery free of landing gear and wings in FOR
- Target speed in the image plane

This list was used to create a new SGF objective function. In addition to values for each quantity, the new score function includes the relative weighting of each factor.

The number of pixels on target was estimated from:

1. Camera Horizontal Field of View (degrees)
2. Camera Vertical Field of View (degrees)
3. Camera Horizontal Resolution (pixels)
4. Camera Vertical Resolution (pixels)
5. Target size (m^2)

The first two parameters are specific to a particular camera and lens. Parameters 3 and 4 are specific to the digital video encoding. Parameter 5 is simply the area of the target when viewed from above. If the estimated number of pixels on target (POT) in the image plane is less than a pre-specified threshold, the SGF score will be 0.

The new objective function made the SGF algorithm considerably more robust. The user now has the ability to change cameras, lenses, encoding technology and target size by simply letting the algorithm know which setup is being used. For example, if the experiment was using a zoom lens, running with high resolution imagery or tracking a very large target, the algorithm would natively understand that the UAS can operate at a further distance and still be able to produce quality detections. These changes have been implemented and tested in simulation and in flight.

3.5.3 Improved Flight Testing Results

With our revisions to the SGF algorithm and software, we went back to Camp Roberts in September 2010 for additional flight testing. We observed dramatic improvements in performance, and we were able to track targets for tens of minutes. Figures 15–17 show improved performance for all metrics presented in Section 3.2. Figure 18 shows a screen capture from SGF running live and tracking a moving truck.

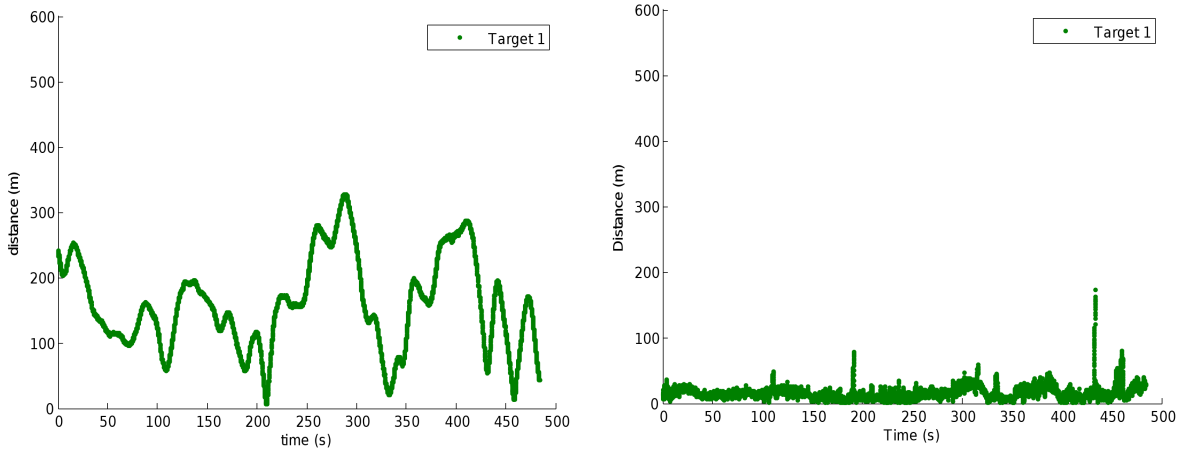


Figure 15. LEFT: 2D projected planar distance between UAS and target over 8 minutes (cf. Figure 5. Distance is lower, on average, than initial testing, and is changing more slowly. These results are not directly comparable with Figure 5 because the flight above used a zoom lens, so the UAS did not need to be as close tot he target. RIGHT: 2D distance between sensor boresight (intersected with the Earth) and the target, plotted over time, shows significant improvement over Figure 6.

4. CONCLUSIONS AND FUTURE WORK

We have shown that a properly implemented sensor guided flight algorithm can enable small UAS to maintain persistent surveillance on moving targets, despite their limited speed, maneuverability, and sensing capabilities. We were able to achieve 99% eyes-on-target using a 60" Unicorn UAS with a left-looking gimbal/sensor suite. However, excellent results were only achieved after careful implementation and several iterations of simulation and flight testing. This highlights the discrepancy between simulation and reality, even when the simulation environment uses high-fidelity models and attempts to capture real-world phenomena like wind.

Toyon is continuing to refine and improve our SGF algorithm. We have several more flight tests and demonstrations planned for 2011, and we hope to begin testing the algorithm with additional small UAS platforms, like Raven. In 2011 we will also begin work on a multi-agent cooperative version of SGF.

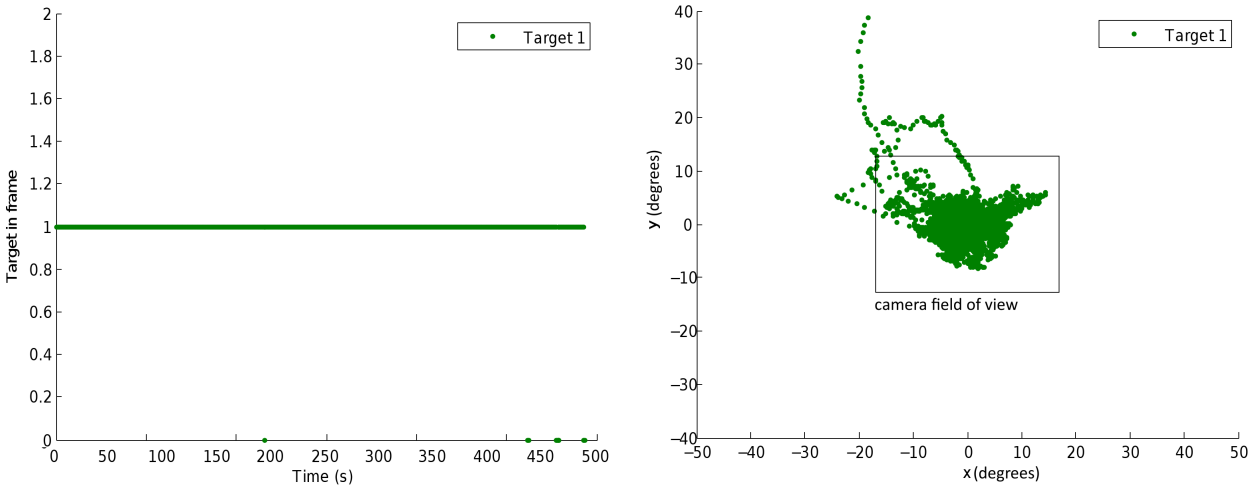


Figure 16. Target stayed within the camera field of view (FOV) 99% of the time (cf. Figure2-3).

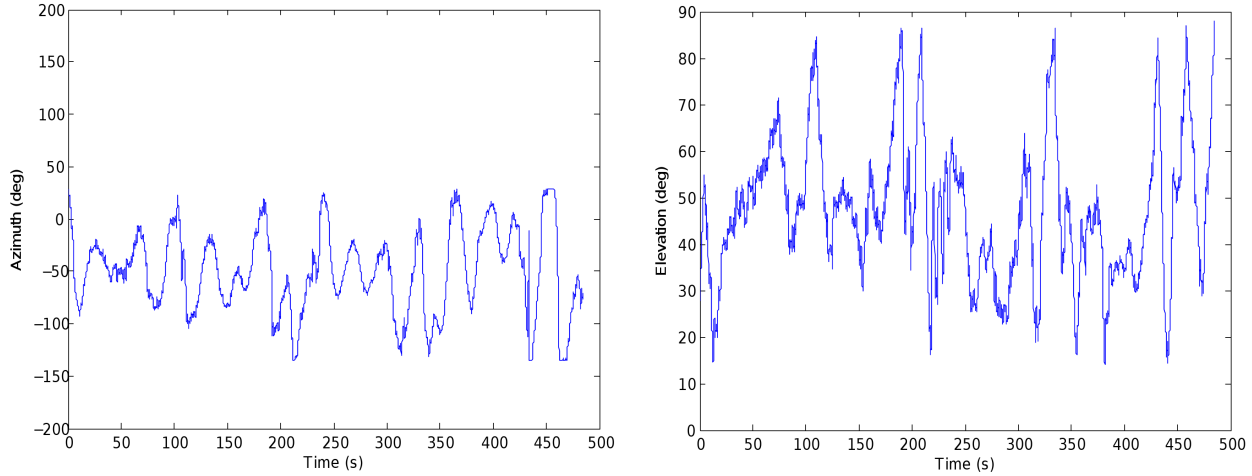


Figure 17. Gimbal azimuth and elevation commands required to keep the camera pointed at the target. These numbers are changing much more slowly than in Figure 8, improving the performance of our image processing software.

APPENDIX A. TOYON'S UAS TESTBED

Toyon maintains a testbed of small Unicorn UAS in a mobile UAS lab. The Unicorn is a hobby-type flying wing made from EPP foam. These battery-powered airframes fly at 13–18 m/s at altitudes of 70–700m, with sorties lasting about 45 minutes. Airframe avionics are controlled by the Procerus Kestrel^{TM} Autopilot, which also controls an on-board gimbaled video camera. Our algorithms can command the autopilot, camera, and gimbal via 900 MHz wireless link. We can automatically task the UAS with waypoints or roll commands, and we task the sensor with pointing commands at up to 5 Hz. Our image processing algorithms receive a digital MJPEG feed from the UAS, from which they can automatically detect and track ground targets. This UAS is ideal for testing image processing and control algorithms because it is rugged, easy to use, and provides a more challenging environment for the algorithms than most fielded DoD platforms. Toyon regularly flight tests our algorithms at the CIRPAS facility within Camp Roberts, CA.

APPENDIX B. TRACKER

SGF UAS feed video to Toyon's Image Plane Video Tracker (IPVT)[4] software. This image processing package automatically removes background motion and detects moving objects (such as our target vehicle) in the video sequence. In each processed frame, IPVT attempts to segment moving objects (foreground) from the stationary background to generate "detections". A sequence of frames containing one or more targets generates a sequence

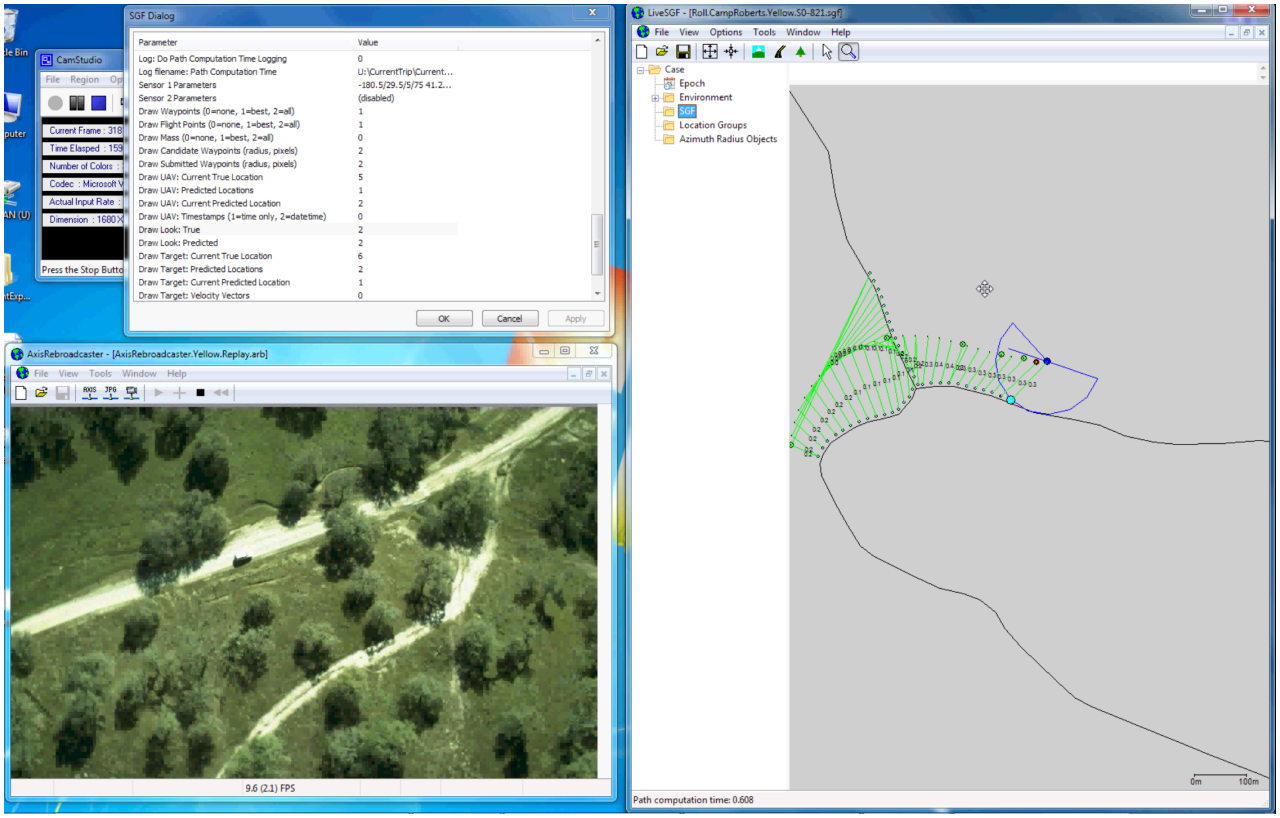


Figure 18. SGF tracking a truck in Camp Roberts, CA. Green lines show anticipated clear-line-of-sight to the target, and the small black numbers show the "score" of each SGF future state. As the vehicle approaches an intersection on the road, SGF's target prediction model splits the target state estimate along both road segments and attempts to route the UAS on a path that can image both estimates simultaneously.

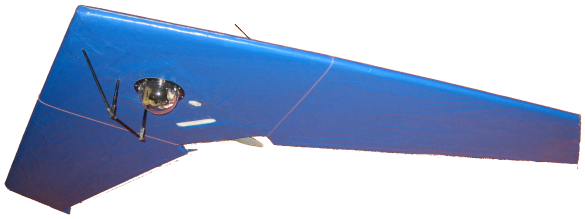


Figure 19. Unicorn UAS

of associated detections that are fused, using filtering techniques, into a target "track". The track is maintained by detections generated in the new video being processed. If the target leaves the sensor FOV for too long, the track will be dropped.

More information on Toyon's IPVT can be found on our website: `www.toyon.com`.

ACKNOWLEDGMENTS

Support for this work was provided by Army Aviation Applied Technology Directorate (AATD) Phase II SBIR contract W911W6-10-C-0001.

REFERENCES

1. G. E. Collins, "An automatic UAV search, intercept, and follow algorithm for persistent surveillance," in *Ground/Air Multi-Sensor Interoperability, Integration, and Networking for Persistent ISR*, M. A. Kolodny and T. Pham, eds., **7694**, SPIE, April 2010.
2. G. Collins, "Sensor guided flight for unmanned air vehicles," in *SIAM Conference on Control and its Applications (CT09)*, July 2009. (Presentation only).
3. Brigham Young University Human Centered Machine Intelligence (HCMI) and Multiple Agent Intelligent Coordination and Control (MAGICC) labs, "Aviones UAV flight simulator." `http://aviones.sourceforge.net/`.
4. A. P. Brown, K. J. Sullivan, and D. J. Miller, "Feature-aided multiple target tracking in the image plane," in *Proceedings of the SPIE Conference on Intelligent Computing: Theory and Applications IV*, **6229**, (Orlando, FL), April 2006.