Controlling UAVs with Sensor Input Spoofing Attacks

Drew Davidson
University of Wisconsin
davidson@cs.wisc.edu

Hao Wu
University of Wisconsin
hw@cs.wisc.edu

Robert Jellinek
University of Wisconsin
jellinek@cs.wisc.edu

Thomas Ristenpart
Cornell Tech
ristenpart@cornell.edu

Vikas Singh
University of Wisconsin
vsingh@biostat.wisc.edu

Abstract

There has been a recent surge in interest in autonomous robots and vehicles. From the Google self-driving car, to autonomous delivery robots, to hobbyist UAVs, there is a staggering variety of proposed deployments for autonomous vehicles. Ensuring that such vehicles can plan and execute routes safely is crucial.

The key insight of our paper is that the sensors that autonomous vehicles use to navigate represent a vector for adversarial control. With direct knowledge of how sensor algorithms operate, the adversary can manipulate the victim's environment to form an implicit control channel on the victim. We craft an attack based on this idea, which we call a *sensor input spoofing attack*.

We demonstrate a sensor input spoofing attack against the popular Lucas-Kanade method for optical flow sensing and characterize the ability of an attacker to trick optical flow via simulation. We also demonstrate the effectiveness of our optical flow sensor input spoofing attack against two consumer-grade UAVs, the AR.Drone 2.0 and the APM 2.5 ArduCopter. Finally, we introduce a method for defending against such an attack on optical-flow sensors, both using the RANSAC algorithm and a more robust weighted RANSAC algorithm to synthesize sensor outputs.

1 Introduction

In recent years, there has been a great deal of interest in autonomous vehicles. The vehicles themselves come in a wide variety of form factors suited to various purposes from driverless cars to minesweeping submarines to self-routing vacuum cleaners. Despite this variety, nearly all autonomous vehicles rely on sensors to navigate their environment and perform their given task. Even semi-autonomous vehicles, for which there is a human operator, often use sensors to contextualize operator commands or to react to unforeseen circumstances. For example, the Google Self-Driving Car uses LIDAR to detect proximity to obstacles and the Roomba automated vacuum cleaner uses downward-facing "cliff sensors" to avoid falling off ledges. As these examples illustrate, an autonomous vehicle's sensors are integral for ensuring that the craft can operate safely without doing harm to itself or its environment.

The primary contribution of our paper is to introduce the notion of a *sensor input spoofing attack*, in which an adversary exerts sustained, direct control over an autonomous vehicle (rather than introducing random noise or misclassifying a single image frame). Implicit control channels presented by sensor systems are essentially unavoidable, since the vehicle cannot completely ignore its sensor systems and still safely navigate. However, we argue that sensor systems can be made more robust to adversarial control. We consider several ways to mitigate sensor input spoofing attacks by hardening different parts of the sensor system's processing pipeline.

To make our discussion of sensor input spoofing attacks more concrete, we focus on a particular type of autonomous vehicle: unmanned aerial vehicles (UAVs). UAVs ares a good target for analysis because they are widely available, they are used in a variety of contexts, they rely heavily on sensors, and they are safety critical. We use the downward-facing *optical flow* system as a representative target sensor. These sensors are used for stabilizing the UAV in mid-flight.

We demonstrate that an attack on this sensor system gives an attacker complete control over the lateral movement of the UAV, even to the extent of causing collisions with obstacles at speeds capable of destroying the drone and damaging the obstacle. We characterize the requirements of mounting this attack and show that it is feasible in practice in both indoor and outdoor settings. We also propose and evaluate defenses against this instance of the attack.

The rest of our paper is structured as follows: In Section 2 we introduce UAV sensors systems for optical flow. In Section 3 we demonstrate sensor input spoofing attacks. In Section 4 we build a simulation framework and use it to characterize a sensor input spoofing attacker. In Section 5 we propose and evaluate defenses for the sensor system. We explore directions for future work in Section 6. Related work is discussed in Section 7. We conclude in Section 8.

2 Optical Flow for UAVs

UAVs use a wide array of sensors in order to fly safely: accelerometers, barometers, sonar, GPS receivers, cameras, and others. By necessity, the UAV must use its sensors to react to its environment in a predictable, timely way. Thus, the hardware sensors stream data to the UAV's navigation software, which uses a variety of algorithms to interpret this stream of sensor readings and trigger navigation commands. A particularly important sensor is the downward-facing optical flow camera, which is used to stabilize the UAV.

Optical flow is a good target for input spoofing: The sensor is a camera, so an unsophisticated adversary can interact with the sensor simply by obscuring the ground plane image detected by the camera. Furthermore, UAVs use optical flow readings to cancel out reported drift using an equal and opposite lateral movement. Thus, an adversary with control over optical flow can exercise an implicit control channel over the lateral movement of the victim.

Representative UAV Models: To gauge the effectiveness of optical flow sensor systems as deployed in practice, we tested a number of the most popular commercially available UAVs. We selected two representative UAVs to analyze: The *AR.Drone 2.0* is a popular, preassembled quadcopter UAV. While the AR.Drone line of UAVs is primarily used for hobbyists, a number have been purchased in an official capacity by police departments. The sensor system hardware uses a low-resolution 60 FPS vertical QVGA camera.

The *ArduCopter* is an open source UAV platform: the core model that we tested uses an APM 2.5 circuit board running the open-source ArduPilot core software, version 3.1.5. The optical flow sensor system deployed by the ArduCopter uses a special purpose optical flow camera, the ADNS-3080.

Optical Flow Details: The fundamental concept behind optical flow is that the camera hardware can detect if the UAV is drifting by comparing successive frames of the ground plane below. In essence, the sensor system will attempt to infer if the ground plane image has moved by a relative offset of $(\Delta x, \Delta y)$. If so, the system will assume that the ground plane is stationary and infer that it has drifted $(-\Delta x, -\Delta y)$. The UAV will attempt to move $(\Delta x, \Delta y)$, the same displacement reported by the sensor firmware, in order to compensate for this drift.

In practice, optical flow first requires a feature detection algorithm to identify regions of the ground plane image that are particularly amenable to tracking. These features can then be fed into the optical flow algorithm proper. The actual optical flow algorithm identifies the location of features in two successive ground plane image frames and uses these differences to compute an overall displacement for the image.

A classic configuration for optical flow sensors is to use the Shi-Tomasi [19] corner detection algorithm for detecting good candidate features, and the Lucas-Kanade [13] method of computing optical flow. In Shi-Tomasi, which is based on the Harris corner detector [9], features are selected if they represent pixel regions that are *corners*, defined as regions of the ground plane frame in which the first derivative of the image signal is large in both the x and y direction. Intuitively, this corresponds to sharp contrast between two neighboring pixels in both x and y, as opposed to *edges*, in which the first derivative of the image signal is in either the x or y direction. Shi-Tomasi is often used since it performs well while being efficient; more sophisticated feature detectors (SURF [2], SIFT [12]) are usually much slower (c.f., [15]).

Lucas-Kanade is the classic optical flow algorithm. It assumes that the difference between two consecutive frames is small and approximately constant within some neighborhood. This assumption is arranged to be met for UAV sensors by a combination of a sufficiently high camera framerate and sufficiently low resolution. For each corner pixel p returned by the Shi-Tomasi corner detector, define a local window of n neighboring pixels q_i for $i \in \{1, \ldots, n\}$. Then for each i let

$$I_x(q_i)V_x + I_y(q_i)V_y = -I_t(q_i), \tag{1}$$

where $I_x(\cdot), I_y(\cdot), I_t(\cdot)$ are partial derivatives of the image along x, y and time and V_x, V_y are the velocity or motion along x and y. Lucas-Kanade estimates V_x, V_y using a least squares method as the set of equations is typically overdetermined, and then outputs the (component-wise) average over the V_x, V_y solutions for each feature as the final motion $(\Delta x, \Delta y)$.

3 Real-World Attacks

In this section, we consider the high-level requirements that a sensor input spoofing attacker must meet in order to mount an attack in practice, and show that an optical flow attack on UAVs is realistic and cheap. In many cases, the attacker gains full control over the lateral motion of the UAV, even to drive the UAV to collide with obstacles.

3.1 Attacker Requirements

For a sensor input spoofing attack to work, an adversary must meet three requirements, which we call the *environment influence* requirement, the *plausible input* require-

Environment	Illuminance (lux)	ArduCopter			AR.Drone		
		Benign	Projector	Laser	Benign	Projector	Laser
Tile	200	Drift	Fail	Control	Drift	Fail	Control
Carpet	150	Drift	Fail	Control	Drift	Fail	Control
Concrete	138	Stable	Control	Control	Stable	Control	Control
Grass	438	Stable	Fail	Fail	Stable	Fail	Fail

Table 1: *Optical flow results on two reference UAVs. The "Benign" columns indicate whether or not the UAV is stable or drifts without adversarial influence. The other columns indicate whether the adversary can successfully control the movement of the UAV using a projector (Projector), or the grid pattern of lasers (Laser), respectively.*

ment, and the *meaningful response* requirement. Meeting the environment influence requirement means that the adversary can alter the physical phenomenon that the sensor system measures. For the case of the optical flow adversary, this means that the attacker must be able to alter the appearance of the ground plane that the optical flow camera captures. The strength of a sensor input spoofing adversary depends on the effort required to influence the environment. As an example where environmental influence may be hard to achieve, an adversary who attacks the magnetometer of a UAV would need to be able to project a powerful enough magnetic field to stand out against the earth's natural magnetic field. In the case of a flying UAV several meters in the air, a ground-based attacker would need powerful equipment to create a field strong enough to reach the UAV.

On the other hand, meeting the plausible input requirement means that the attacker can use their influence over the environment to induce a reading by the sensor system that will actually be used by the system as valid input. For example, if a sonar sensor only uses regular sonar pulses, then the adversary must ensure that they are adding or subtracting pulses in sync with the pulses that the original sensor emits. We note that the plausible input requirement is with respect to the sensor firmware: if the firmware does not employ reasonable sanity checks, then exceptional input readings may be considered plausible.

Finally, meeting the meaningful response requirement means that the adversary can induce a behavior on the UAV, representing the control that the adversary has over the device. For example, an adversary may be able to spoof inputs to an infrared thermometer onboard an autonomous vehicle, but they can only exert control over the vehicle in the unlikely event that the vehicle changes its behavior based on temperature.

A crucial insight for mounting a sensor input spoofing attack is that an adversary can leverage knowledge of how the sensor algorithm translates its environment readings into judgments for the control software of the autonomous vehicle. With this in mind, plausible input may correspond to a degenerate case of the algorithm. In principle, a weaker algorithm allows an adversary with more limited influence to induce a response.

3.2 Optical Flow Attackers

Consider the above framework in the context of an optical flow attack on a UAV. We discuss each of these requirements, and examine the capabilities of the attacker to meet each one:

Environment Influence: To meet this requirement, the adversary must be able to alter or obscure the ground plane, thus indirectly altering the pixel values reported by the optical flow camera. To meet the plausible input requirement, the attacker must ensure that the surface quality measurements of the optical flow system are above the threshold. A strong attacker could accomplish this by covering the ground plane with a sheet of material printed with a feature-rich pattern of their choice. However, we consider this degree of physical proximity to the victim to be overly onerous to the attacker: physically entering the ground plane region of the UAV is highly conspicuous, and the adversary would need to carry the obscuring material. Instead, the adversary can influence the ground plane image from a distance by projecting light onto the surface. This has two advantages: the adversary can mount the attack from a distance, and they can use an emitter that is smaller than the region that they need to spoof.

We designed two different methods of light-based optical flow sensor input spoofing: In the first method, the adversary constructs an image that contains a large number of corners, and uses a projector to shine the image onto the ground plane. In order to make the attack more reasonable, we used a AAXA Pico handheld projector, a battery operated projector that can load an image from a USB stick. This allows the adversary to easily carry and conceal the projector in a pocket.

In the second method, the adversary directs lasers onto the ground plane to induce features. While a single laser beam is only likely to induce a single feature, the attacker can split or filter the beam to project a pattern from the laser, or use an array of laser emitters. For our experiment, we use a single 5 mW laser fitted with a filter to project a regular grid pattern of dots onto the surface.

Plausible Input: In "loiter" mode, the purpose of the optical flow sensor system in a UAV is to detect a lateral drift displacement and compensate by instructing the vehicle to move in an equal and opposite displacement. The key insight in our attack is that the sensor system assumes that the ground plane image is stationary, so it interprets a feature's motion along a vector v as a drift along a vector $-v$. However, if the ground plane *is* moving, then the UAV will follow the motion – essentially tracking the moving ground plane. The sensor input spoofing attack can take advantage of this behavior by controlling and then moving features of the ground plane image.

Meaningful Response: The adversary can leverage knowledge of the optical flow algorithm to construct effective input to the sensor system. If the sensor system uses the combination Shi-Tomasi and Lucas Kanade method discussed in Section 2, then the attacker's goal is to project a sharp gradient onto the ground so that the feature detection algorithm will pick up the light as a corner. Since the Lucas-Kanade-based optical flow computes a final displacement based on the average displacement of each feature, the attacker needs to generate a large number of corners.

In practice, the attacker can simply sweep their projected light across the ground plane. If effective, gradients in the projected light source will be picked up as corners, and induce moving features. As described above, this will cause the UAV to follow the path of the light. The challenge of the attack is to overcome the influence of the "natural" features that are part of the unaltered ground plane. This is limited by two factors: the feature-richness of the benign scene upon which the attack projects their light, and the strength of the gradient that the attacker can project.

3.3 Attack Setting

To test the effectiveness of our attack methods in practice, we deployed each of our UAVs in a variety of realistic scenes. We selected scenes that naturally present different levels of challenge to the adversary: in a well-lit scene, the attacker will have trouble overcoming the ambient light to create stark gradients. Also, in a scene that naturally causes a high amount of variation in the ground pixels, the spoofed features are of less comparative value to the feature selection and are less likely to be used. We tested both of the reference UAVs introduced in Section 2 on each of our test scenes by deploying them in the scene, manually flying them to a height of 3 meters, and then initiating a "loiter" command to cause the UAV to maintain it's position in midair. For the laser attack, we swept the laser beam across the ground plane from a distance of 10 feet away. For the project attack, we swept the projector from the launch site of the UAV across the ground plane from a distance of 4 feet. The results of our experiments are shown in Table 1. The four different scenes are as follows:

Tile: is an enclosed lab space with a clean, white tile floor. Although the edges of the 9-inch tiles are visible to the human eye, neither the AR.Drone or the ArduCopter is equipped with a powerful enough camera to capture features from the tiles.

Carpet: is a large, carpeted hallway with a slight geometric pattern. We hypothesized that this geometric pattern would be enough for at least one of the UAVs to recognize features in the surface, but ultimately we saw no perceivable difference between tile and carpet.

Concrete: is an outdoor sidewalk environment. While individual slabs of concrete are a uniform color, the gaps between the slabs are large enough for feature detection. This means that both the AR.Drone and ArduCopter are capable of maintaining a stable lateral position over this surface using optical flow.

Grass: is an outdoor grassy field. This surface appears to represent a very favorable surface for optical flow, with a large number of corners for feature detection to find. Unlike all previous settings, the surface does not contain any regular pattern and regions of identical pixel values are small.

3.4 Discussion

Surprisingly, the sensor systems of both UAVs perform in a similar fashion: Both the carpet and tile do not have a high enough surface quality for optical flow and cause the UAV to drift. We also note that the settings tested span a range of interesting outcomes: the tile and carpet settings represent environments that are not amenable to the benign user's goal and amenable to the attacker's goal: in both settings the UAV will drift in the benign setting, but an attacker can reliably control the UAV using the laser (though not the projector). The concrete setting represents an environment in which a user can successfully use optical flow in the benign setting, but can be overpowered by the attacker. Finally, the grass setting demonstrates an ideal use case for optical flow, in which the highly varied surface benefits the user and mitigates the effect of the attack. The attack failed to exert meaningful control over the UAV with either the projector or laser.

Table 1 shows that the laser attack is effective across many different scenes in which the UAV is likely to be expected to work. The vulnerability of the UAV in indoor environments is particularly troubling, as obstacles abound. We note that the projector attack was only successful in the darkest lighting condition.

Although we mounted our laser attack from 10 feet away on foot, the attack is almost certainly possible from a far greater distance. The grid-pattern filter attached to our single laser causes the beam to become diffuse fairly quickly. However, when removing this filter and focusing the beam to a single point, we were able to induce a feature on the ground from any distance in which we could still maintain line of sight with the ground plane. An adversary with a large number of individual laser emitters could theoretically bundle them together in parallel and direct the entire bundle onto the ground plane. Thus, we believe that the maximum distance under which the laser attack can be mounted is likely to be limited not by the strength of the beam but by the adversary's ability to hit the ground plane. On the other hand, the projector attack emitted light that was simply too weak to induce features. Our original attempt to perform the attack started by standing 10 feet away from the launch site, but ultimately required the attacker to move to 4 feet from the UAV before we observed any meaningful response.

We also note that the adversary may judge the effectiveness of the attack differently depending on his or her goal. The adversary may simply want to lead the UAV away from its stationary position. Alternatively, by rapidly sweeping the light through the ground plane they could cause either UAV to rapidly accelerate in a dangerous way. For example, in the indoor environments it was trivial to cause the AR.Drone to accelerate towards a wall with enough force to damage its protective shield.

These results show that optical flow attacks are possible in real-world settings. Although our projector attack showed severe limitations, both in terms of the diversity of settings where it was effective and in the required to the target, the laser attack showed that optical flow sensor input spoofing attacks can be achieved with a commodity laser pointer. In order to more fully characterize the strength of this attack strategy, we turn our attention to exploring how the adversarial control effects optical flow algorithms.

4 Sensor Input Spoofing Simulation

To better study attacks and educate pursuit of countermeasures, we analyze the performance of optical flow algorithms in simulation. We focus in particular on the popular Lucas-Kanade motion detection algorithm, discussed in Section 2. Although the firmware of the optical flow sensors for both of our UAVs are closed source, the attacks that we mounted in Section 3 demonstrates behaviors closely associated with Lucas-Kanade. This fact, in addition to the popularity of Lucas-Kanade for fast optical flow and low-resolution requirements, make it a good candidate for baseline measurement.

Simulation of laser-based attack. We perform a simulation of several variants of the laser-based attack in which we alter the intensity of the attacker's laser and the density of the resulting grid. The simulation is done on videos captured on a Raspberry Pi camera, configured for a high frame rates and small field of view (approximately 1 square meter for the typical hovering height of a UAV), and which is akin to optical flow cameras built for UAVs. The grid pattern is projected onto the video via Povray [1]. We initiate the simulation with a 3×3 grid pattern. Each grid dot is a white spot with size around 0.5 centimeters falling off another 0.5 centimeters. While other configurations are possible, we feel this configuration is sufficient to characterize the behavior of typical optical flow for the purposes of sensor input spoofing. This allows identifying the light intensity an adversary needs to successfully alter the output of optical flow.

We assess the power of the adversary in the following way. The adversarial grid pattern is moved across the field of view in the minus x direction throughout the video. We then run optical flow on the video under benign and adversarial settings, and compare the accumulated motion in the x direction. The adversary pattern is in the view for 60 frames, where the laser grid sweeps through the entire width of the view. In this manner, we can assess how much optical flow is affected by the simulated lasers.

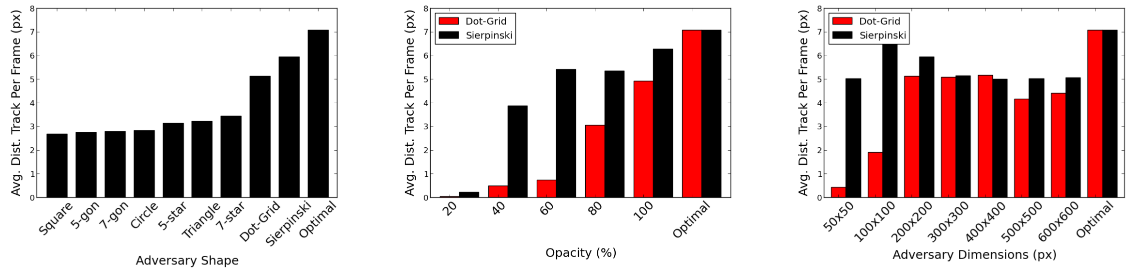
We vary two parameters: laser intensity and the space between grid points. The intensity of lights can be varied by the averaged increment of the pixel grayscale value, and the grid is varied in terms of centimeters betwen dots. The results are presented in Table 2, which gives the cumulative x direction motion detected in the 60 frame simulation. As expected, the power of the adversary increases with laser intensity. In contrast, grids that become overly dense can decrease efficacy, because the Shi-Tomasi corner detector (and most of other feature detectors) do not favor features within short distances. Thus an attacker would want to focus more on covering a larger portion of view, instead of concentrating on a narrow projected pattern.

In the worst case, the grid causes cumulative movement in the negative x direction of 45.1 centimeters. Given that the total movement of the adversarial grid is 1 meter, half of the adversarial movement is picked up by the optical flow algorithm.

	Grid spacing (m)			
Intensity	**0.015**	**0.012**	**0.01**	**0.008**
0.6	1.9	16.0	17.3	14.1
0.8	8.5	27.5	25.6	18.4
1.0	20.8	36.7	44.2	24.6
1.2	32.7	45.1	44.9	32.3

Table 2: Simulated cumulative optical flow in negative x direction (in centimeters) at various levels of laser intensity and distance between grid points.

Simulating other attacks. The discussion above suggests the pattern of pixels controlled by the adversary impacts the efficacy of manipulating the optical flow outputs. To determine the strongest adversarial patterns (in-



(a) Controlling for Shape (b) Controlling for Opacity (c) Controlling for Size

Figure 1: *Results of performing averaging over output motion pairs in v_{diff} to arrive at a final $(dx_{\text{out}}, dy_{\text{out}})$ pair. The closer the bar to the optimal, the stronger the effect the adversary had on the algorithm, indicating a strong adversary. Results are averaged over all four backgrounds for each bar.*

terchangeably, "adversaries"), we tested each adversary against four fixed simulated backgrounds that had varying patterns and correspondingly, varying feature densities. We treat the background as the "ground truth," i.e., what the algorithm should track in the ideal setting. The adversary wins if the algorithm ends up tracking the adversarial pattern instead of the background. Thus, if the background is stationary and the adversary is moving with $(\Delta x = 5, \Delta y = 5)$ pixels, the closer the algorithm's output is to $(5,5)$, the better the adversary is doing. If the algorithm returns a motion pair of $(0,0)$, we say the adversary is weak relative to the background.

Figure 1 gives results for a number of simulations that measure varying adversarial patterns, opacity of the adversarial pixels, and size of the adversarial pattern. All adversaries except in the size experiment were $200px \times 200px$, and the background image was $600px$ in all cases. All adversaries moved at a rate of $\Delta x = 5px, \Delta y = 5px$ so that if the algorithm tracked the adversary optimally, it would yield a per-frame optical flow of $\sqrt{5^2+5^2} = 7.07px/frame$. This is labeled as optimal in each graph. (All adversaries achieved this optimal performance against a monotone background, which is to be expected since such a background introduces no additional features that would perturb the average.) The other columns relay the average distance observed by the optical flow algorithm per frame, averaged over the four backgrounds mentioned above.

Figure 1a shows that while there is a slight ordering on the effectiveness of the various simple shapes (the more pointed concave star shapes being stronger than their convex counterparts), the internally feature-dense adversaries of the dot-grid and the Sierpinski fractal were the strongest shapes by far. We focus on these two adversarial patterns in measuring the effects of opacity and size. Interestingly, Figures 1b and 1c show that the Sierpinski adversary performs well almost immediately, even at small size ($50px \times 50px$) or low opacity (40%). On the other hand, while the Dot-grid is relatively weak at low opacity, and is only effective at dimensions of $200px \times 200px$ and greater relative to the $600px$ background.

The density of sharp corners present in the Sierpinski fractal would allow the adversary to introduce more features into the frame. In turn, this provides greater influence on the outcome of optical flow. This shows that, in theory, there exists better adversarial strategies than the laser grid we used before, but whether one can take advantage of this in practice remains an open question.

5 Defenses

In the previous sections we showed that optical flow can be adversarially manipulated both in practice against real UAVs, and in simulations. The latter particularly allowed measuring efficacy against a known optical flow algorithm, and confirmed the clear intuition that an adversary that can overwhelm the contribution of background features to the average flow can force unwanted movements. We therefore turn now to whether one can build more robust optical flow algorithms that make the adversary's job more difficult.

Using RANSAC. The Lucas-Kanade optical flow averages flow over all features detected. Such naïve averaging works well in the absence of an adversary, since it is fairly robust to random noise. That is, if perturbation is uniformly distributed, averaging will still perform well. However, in the adversarial setting, the introduced perturbation is 'structured' — it is easy to see that the mean will be sub-optimal because individual adversarial features may have outsized affect on the average.

Instead of taking a simple mean over the flow of the detected features, which favors the adversary to a degree proportional to the number of features he controls and the magnitude of those features, we propose applying the RANSAC algorithm [5] to obtain a final $(\Delta x_{\text{out}}, \Delta y_{\text{out}})$ pair. The algorithm works as follows. Let $v_{\text{diff}} = (dx_1, dy_1), (dx_2, dy_2)\ldots$ be the vector of motion for each feat as output by applying Shi-Tomasi to obtain features and then Lucas-Kanade to each of those features. Then RANSAC randomly samples k features and forms a hypothesis for each of them. The hypothesis h_j for vector (dx_j, dy_j) is the ground truth motion for vector (dx_j, dy_j). Then, we let all other features vote for each of these k hypotheses. For a feature motion (dx_i, dy_i) to vote for a hypothesis h_j, the two motion vectors need to be similar in an ℓ_1 norm sense (i.e., below a certain threshold).

$$|(dx_i - dx_j, dy_i - dy_j)|_1 < \text{threshold}. \quad (2)$$

In our application, this threshold could be as small as 2 or 3 pixels. If this criterion does not hold, the motion (dx_i, dy_i) refrains from voting for h_j. RANSAC performs many realizations of this process, and then picks the hypothesis with the highest vote to be the final hypothesis for the frame, and the corresponding motion to be the "ground truth" of the frame, i.e., the pair $(\Delta x_{\text{out}}, \Delta y_{\text{out}})$.

The main advantage of using RANSAC is that assuming the level of corruption is approximately known, the probability that we pick a good hypothesis (corresponding to the 'true' motion) at least once will approach one [5]. Let P be the probability that RANSAC picks a true background motion vector for the frame, p be the probability that a feature motion actually belongs to the background, k be the number of random samples (or realizations) drawn in the RANSAC process and n be the number of feature motions returned from Shi-Tomasi and Lucas-Kanade. Then, with a running time cost of $\mathcal{O}(nk)$, we have,

$$P > 1 - (1-p)^k. \quad (3)$$

Furthermore, RANSAC fails only if all the k samples hit the adversary. The running time will be slightly slower compared to averaging. In practice, k is set to a small constant (10 in our experiments) and is independent of the number of features.

The voting nature of RANSAC makes it more robust to noise than averaging. Especially given an adversary whose motion significantly differs from the background in both magnitude and direction, averaging tries to produce a motion vector that fits both the background and the adversary, while RANSAC will pick the majority vote, essentially choosing only the "inlier" features of the background. If the background wins the majority vote, the resulting motion vector is not compromised by the adversary at all.

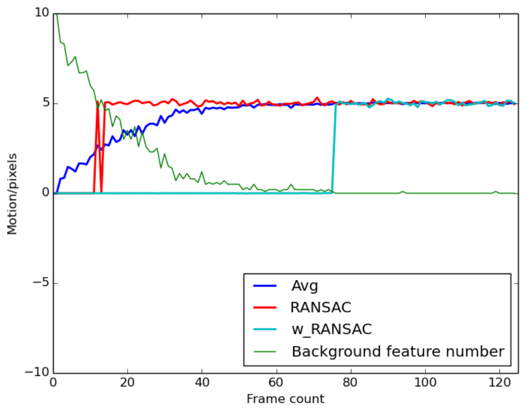
While promising, RANSAC will fail should the number of adversarial features dominate the background features. We thus consider further enhancements to help mitigate sensor input spoofing attacks in cases where an adversary can control a relatively large number of features.

Weighted RANSAC with momentum. We propose introducing a notion of "momentum" into optical flow in order to differentiate background from adversary. As we will see this can provide stable optical flow even in cases where the adversary controls more than 50% of the features.

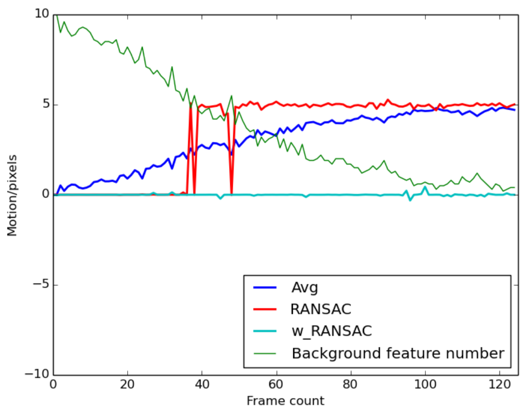
To do so we introduce context about which features consistently appeared in the sensor's field of view in earlier frames. Effectively, we favor features that exhibit "momentum", which discounts both arbitrary perturbation and adversarial manipulation. Suppose that for each feature we have a belief or weight associated with it that comes from the information in previous frames. In a weighted RANSAC process, instead of treating each motion vector uniformly, we sample from a distribution that is proportional to the weights of each feature that produces the motion. In voting, instead of counting each vector as one vote, we weigh the votes accordingly. In this manner, if we are able to build up weights for background features, this weighted version of RANSAC will stand out even in situations where the background does not control the majority of features, provided that the adversary is either a new arrival, or moves dynamically or unstably relative to the background.

We construct the weights in the following way: assume that the adversary enters the view gradually (alternatively, assume that in the first few frames after the drone and its optical flow sensor start, the benign background accounts more than 50% of the features). Initially each feature in the first frame is assigned a constant default weight (or the same "momentum"). Then for each frame, after the final motion $(\Delta x_{\text{out}}, \Delta y_{\text{out}})$ is computed, we upgrade or downgrade the weight of each feature. For a feature whose motion vector (dx_i, dy_i) is close to the motion $(\Delta x_{\text{out}}, \Delta y_{\text{out}})$, we upgrade the weights by a factor of $1 + r$ where r is a learning rate. Otherwise, we downgrade the weight of the feature by a factor of $\frac{1}{1+r}$. In our experiments we use $r = 0.1$, which proves to be sufficient to build a strong enough belief for background features. Other stages of the pipeline are similar to standard RANSAC. In each frame, we sample k features according to the weights and pick the one with the best weighted vote.

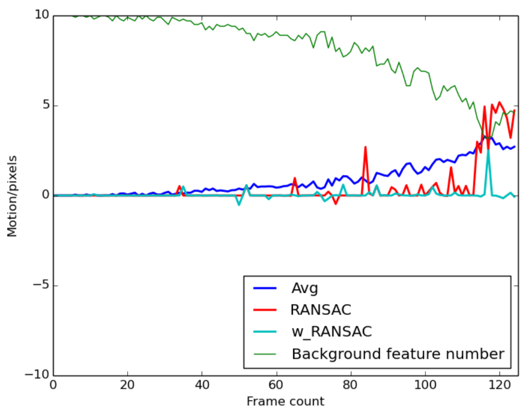
We performed a set of simulations over carpet, grass and asphalt backgrounds and used the full-screen Sierpinski fractal as adversary, as it was the strongest adversary from our previous experiments. The adversary enters the view from left at a constant speed of $dx = 5$ pixels per frame. The background stays static. Shi-Tomasi corner detection returns the best 100 features detected in the image frame. The carpet background loses almost all features when the adversary is only half-way into the view. The grass background is stronger, since it contains about 10 features when the adversary occupies the



(a) Carpet background.



(b) Grass background.



(c) Asphalt background.

Figure 2: *Motion detected from Sierpinski adversary on different backgrounds.*

full view. Meanwhile, the asphalt background is even stronger since more background features remain after the adversary enters the scene.

RANSAC performs well when the number of features controlled by the background is greater than 50, or $p > 0.5$ in Equation 3. The performance of RANSAC depends heavily on p as we can see in Figure 2a and Figure 2b. Whenever p falls below 0.5, RANSAC switches immediately towards the adversary. This is because the bound in Eq. 3 applies for adversary as well, meaning that it follows the adversary with high probability when $p < 0.5$. The weighted RANSAC algorithm outperforms both averaging and standard RANSAC in most cases. Even in Figure 2a where the features from background drop quickly, giving the algorithm little time to build up weights for background features, it still sticks to the background quite well until the adversary takes all the features, after which there is no more ground truth motion to follow so it switches to the adversarial features.

In summary, as long as there is a small period of time to build up the weights for the background features and there are at least some background features in view, weighted RANSAC is still able to pick up the background motion. This is evident, for example, in Figure 2b, where despite there being almost no background features at the end, weighted RANSAC still tracks those features.

Finally, we note that all three approaches—averaging, standard RANSAC, and weighted RANSAC—respond in real time to the movement of the background when only a background is present in the field of view and there is no adversary. However, while it is trivial to defeat averaging and standard RANSAC (the adversary must only control more than 50% of the features), weighted RANSAC is robust to adversarial inputs so long as the assumption holds that the algorithm was initialized with the true background in its view, and the adversary only appeared after a delay.

We finally evaluate weighted RANSAC on the set of simualted laser grid videos described previously in Section 4. Here we quantify the laser light by the increment of grayscale intensity each laser dot contributes. For example, value 50 on the horizontal axis denotes a three-by-three grid, where each of the laser dots in the grid contributes an increment of 50 in the total grayscale intensity of the image. In Figure 3, the dashed lines are the performance of Lucas-Kanade and the solid lines are representing weighted RANSAC. In all cases, weighted RANSAC outperforms Lucas-Kanade by a wide margin. The best adversary for Lucas-Kanade picks up about 45% of the adversary movement, while under the same adversary weighted RANSAC is affected by 29% only. The percentage is obtained via dividing the extra movement by the width of the frame (96) since the adversary moves across the whole frame. The strongest adversary for weighted RANSAC occurs at a different density, where weighted RANSAC is affected by 37% in the worst case.

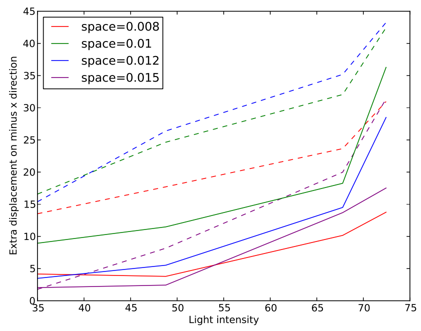


Figure 3: *Adversarially induced motion for Lucas-Kanade (dotted lines) versus weighted RANSAC (solid lines) on simulated laser grids with different spacings. The background is taken from a Raspberry Pi camera video. Lower is better.*

6 Future Work

Our evaluation shows that sensor input spoofing attacks constitute a real attack vector that must be addressed to secure autonomous vehicles, and our case study shows that such attacks can be exercised under realistic conditions. The previous section describes mitigation strategies to defend against sensor input spoofing attacks, but we believe that there is still much work to be done in the general realm of sensor input spoofing and in the specific case of hardened optical flow. In this section, we describe some of the more promising directions.

Additional sensor attacks: The optical flow attacks demonstrated in this paper are representative of a class of sensor input spoofing attacks. Future work could investigate similar attacks on other sensor systems. In particular, it is important to consider spoofing attacks on LIDAR and sonar, as both of these sensor systems are used to detect proximity to obstacles (UAVs commonly use a downward-facing sonar to detect proximity to the ground). While the attacks discussed in this paper intuitively attempt to make the autonomous vehicle detect an object not actually present, it may be possible through LIDAR or sonar interference to cause the autonomous vehicle to fail to see an object which actually is present.

Hardware-level robustness: Our experiments above show that an adversary's strength can be quantified by the number of features they can control. Thus, we may consider ways for the optical flow sensor to sense the features from the background more reliably. In particular, a higher resolution camera, while more expensive, produces a larger, crisper field of view, which is harder for the adversary to cover.

Optical flow parameter tuning: It is likely the case that our defenses could be tuned to particular use cases for UAVs. For example, both of our UAV models threshold the displacement that the optical flow sensor reported, allowing an attacker to quickly move their light source through the ground plane and cause the UAV to veer off sharply in feature-poor environments. Although we do not have a full view into the firmware of the UAVs, we believe that no thresholds are in place, or if so do not prevent highly implausible input which results in dangerous behavior on the part of the UAVs. From a purely statistical point of view, it may be possible to bound the displacement of the optical flow system in a number of ways such as maximum speed, perceived light gradient, etc.

Sensor fusion: Combining data from multiple distinct sensors, known as *sensor fusion* [3], significantly raises the difficulty of sensor input spoofing attacks. As an example, actual drift by a UAV will be detected by the accelerometer, while spoofed input motion will not. If the UAV used both sensors to determine drift, the adversary would have to interfere with both sensors simultaneously. This significantly raises the bar for the attack, since each sensor detects distinct physical phenomena. One specific direction for future work is to analyze the control software of the UAV to ensure that control channels are appropriately guarded by sensor fusion sanity checks.

7 Related Work

Fully-autonomous and semi-autonomous vehicles are subject to a large body of work. To our knowledge, this paper is the first work to broadly characterize sensor input spoofing attacks and in particular to describe such attacks against optical flow. In this section, we describe work most closely related to our own.

Robust optical flow: Drone navigation using optical flow has been studied before in [7, 4]. Their focus was on using optical flow for lateral stabilization and navigation in GPS-deprived environments.

Seegmiller *et al.* studied optical flow in the context of semi-autonomous vehicle navigation for planetary rovers [18]. They observe the need for robust optical flow algorithms, and propose the use of RANSAC to avoid miscalculating optical flow due to the movement of a rover's shadow within the field of view of its optical flow sensor. Seegmiller *et al.* did not consider adversarial manipulation of the scene, but only when faced with a specific and predictable bias introduced by the rover's shadow.

To the best of our knowledge, our work is the first to consider intentional manipulation of the optical flow environment, and is the first to study how such interference could be used to control a UAV rather than simply lose sensor fidelity due to noise.

Channel hijacking: Sensor input spoofing attacks leverage an implicit control channel. In constrast, channel hijacking is focused on hijacking the explicit channels that are intended to control an autonomous vehicle. For example, we found that the AR.Drone operates over an unencrypted tcp/ip WiFi channel. Consequently, we found that it was trivial for a 3rd-party with a WiFi-enabled device to deauthenticate the legitimate controller, and in the worst case wresting control over the WiFi channel to the UAV.

A representative project in the domain of channel hijacking is Skyjack [11]. Skyjack equips an autonomous vehicle with additional hardware and a WiFi stack. The vehicle can then be piloted within range of target UAVs, and used to hijack the unsecured WiFi channel of the target midflight. We consider such attacks to be orthogonal to sensor input spoofing attacks, because they leverage protocol weaknesses in designated channels. As such, The natural solution to channel hijacking is encrypt the channel (e.g., by encrypting the target UAV's WiFi channel) or to otherwise harden the protocol itself. In contrast, sensor input spoofing interferes with the detection of physical phenomena outside the control of protocol designers. Nevertheless, channel hijacking may be used in coordination with a sensor input spoofing attacks.

Signal jamming: A number of attacks on UAVs are based on sending noise into a sensor or receiver so that it can no longer report a signal [10, 14, 20]. We do not consider attacks on jamming attacks to be instances of sensor spoofing attacks, as our attacks are designed to spoof plausible data that change the behavior of the UAV.

Kune *et al.* describe attacks and defenses based on electromagnetic interference (EMI) [6]. In these attacks, electromagnetic signals are used to confuse the electronics in various sensors. Kune *et al.* demonstrate Ghost Talk, for using EMI to confuse acoustic sensors. The application of Ghost Talk that is closest to our own work is in its use of EMI to make acoustic sensors register dial tones that were never actually emitted. Unlike sensor spoofing attacks, which rely on exploit weaknesses in the sensing algorithm, Kune *et al.* exploit weaknesses in the electronics of the sensors. As a result, Kune *et al.* propose mitigation techniques such as shielding the electronics from the EM signal or using components that detect the EM signal. These strategies do not apply to sensor input spoofing attacks, since they do not rely on EMI. Ultimately, EMI shielding and algorithmic robustness are complementary and necessary defenses for securing electronic sensors.

Adversarial machine learning: A recent body of work constructs adversarial inputs to machine learning algorithms [16, 17, 21, 8]. The goal of this work is to modify an input such that the machine learning algorithm misclassifies it. In the worst case, the change may be too subtle to be recognized by a human, but powerful enough that the attacker can choose any classification they wish. While adversarial machine learning is similar in spirit to sensor input spoofing attacks, the techniques are different. Sensor input spoofing directly attacks the sensor algorithm, rather than a trained classifier. Thus, adversarial machine learning relies on "imperfections in the training phase" [16] whereas sensor input spoofing does not. Similarly defenses against adversarial machine learning focus on hardening the training phase of the algorithm [17] rather than direct changes to the algorithm and hardware, as we propose.

8 Conclusion

Autonomous vehicles are useful in a wide variety of settings, but the capabilities that they provide also offer unique attack vectors: in addition to finding the presence of traditional control channel vulnerabilities, we also found a new class of vulnerabilities, called *sensor input spoofing attacks* that exploit the environmental sensors of UAVs to open implicit control channels.

We perform a case study on optical flow sensors. We find that the algorithms currently employed by this sensor are vulnerable to a sensor input spoofing attack, which is exacerbated by control software that fails to validate readings from its optical flow sensor.

Finally, we explore defense-in-depth mitigation for the entire range of attacks that we found, including a novel algorithm that is robust to malicious input.

As autonomous vehicles become increasingly ubiquitous, the importance of discovering, characterizing, and mitigating new attack vectors will continue to grow. We believe that sensor input spoofing attacks are a step in this direction, and represent a promising direction for further work.

References

[1] Pov-ray - the persistence of vision raytracer by david k. buck, aaron a. collins.

[2] H. Bay, T. Tuytelaars, and L.V. Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[3] Richard R Brooks and Sundararaja S Iyengar. *Multi-sensor fusion: fundamentals and applications with software*. Prentice-Hall, Inc., 1998.

[4] M. Dille, B. Grocholsky, and S. Singh. Outdoor downward-facing optical flow odometry with commodity sensors. In *Field and Service Robotics*, pages 183–193. Springer, 2010.

[5] M.A. Fischer and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Comm. of the ACM*, pages 381–395, 1981.

[6] Denis Foo Kune, John Backes, Shane S. Clark, Daniel B. Kramer, Matthew R. Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost talk: Mitigating EMI signal injection attacks against analog sensors. In *Proceedings of the 34th Annual IEEE Symposium on Security and Privacy*, May 2013.

[7] N. Gageik, M. Strohmeier, and S. Montenegro. An autonomous UAV with an optical flow sensor for positioning and navigation. *INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS*, 10, 2013.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *Proceedings of the 2015 International Conference on Learning Representations*, 2015.

[9] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[10] A.Y. Javaid, W. Sun, V.K. Devabhaktuni, and M. Alam. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 585–590. IEEE, 2012.

[11] S. Kamkar. SkyJack. `http://www.samy.pl/skyjack`, December 2013. Accessed: 2014-05-16.

[12] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[13] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[14] T. Nighswander, B. Ledvina, J. Diamond, R. Brumley, and D. Brumley. Gps software attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 450–461, New York, NY, USA, 2012. ACM.

[15] N. Nourani-Vatani, P. Borges, and J. Roberts. A study of feature extraction algorithms for optical flow tracking. In *Australasian Conference on Robotics and Automation*, 2012.

[16] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 372–387, 2016.

[17] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE Symposium on Security and Privacy, S&P 2016, San Jose, California, May 23-25, 2016*, pages 372–387.

[18] N. Seegmiller and D. Wettergreen. Optical flow odometry with robustness to self-shadowing. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 613–618. IEEE, 2011.

[19] J. Shi and C. Tomasi. Good features to track. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, June 1994.

[20] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 881–896, Washington, D.C., 2015. USENIX Association.

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *Proceedings of the 2015 International Conference on Learning Representations*, 2014.