CPS Project: Security in AI-aided Autoland

Anshuman Suri
as9rw

***Abstract*—Developments in robust artificial intelligence for autonomous systems is finding its way to aviation. Aimed to enhance current autopilot systems and enable autonomous takeoff and landing, these systems bring with them a lot of potential security problems. This project covers some AI-driven approaches under development in the aviation industry, along with their potential security problems and how they could be alleviated. With a focus on autonomous AI-drive landing systems, this work presents a proof-of-concept as to how such systems can be vulnerable to input-based adversarial attacks.**

I. Introduction

The relevance of cyber-security and modelling for autonomous vehicles has gained increasing attention over the years, with projects like Waymo [1] using virtual simulations to achieve billions of miles worth of driving experience. The safety of automated vehicles is crucial to their long-term success, especially for aviation-based systems operating in hostile conditions: with temperatures as low as $-70°$F, tailwinds as high as 100mph, and turbulent seas the only surface below. Without the luxury of slowing down and pulling over, aircraft require redundancies to ensure passenger safety.

Points of failure in such a system grow with the increasing dependency of such aircraft on automated tools: under both unintended and adversarially induced environments [2], [3]. Aircraft deploy systems like ILS (Instrument Landing System) for landing and are, like any cyber-physical system, prone to error (a crash in 2001 is believed to have been because of an ILS-based error [4]). The presence of real-time air-traffic data via ADS-B opens up avenues for side-channel attacks as well [5]. With system failures potentially leading to multiple fatalities and hundreds of millions of dollars worth of losses, these systems need to be thoroughly tested and equipped with redundancies. AI-guided landing systems have also started to receive attention [6], with a recent test conducted by Airbus for automated take-off and landing [7].

II. Problem Definition

This work considers a threat model consists of an adversary that wishes to cause harm to an autonomous aircraft that uses a vision-based system with AI to perform landing. The adversary can generate input-adversarial images via adversarial patches that it can print and place on the visible path of an aircraft's descent. Concretely, the adversary wishes the cause a deviation in the aircraft's descent normal to the runway, leading to the aircraft ideally missing the landing strip in the worst case. To maximize its impact, the adversary wishes to cause deviations in only one direction, so that they may add up and cause maximum harm- potentially leading to a fatal crash.

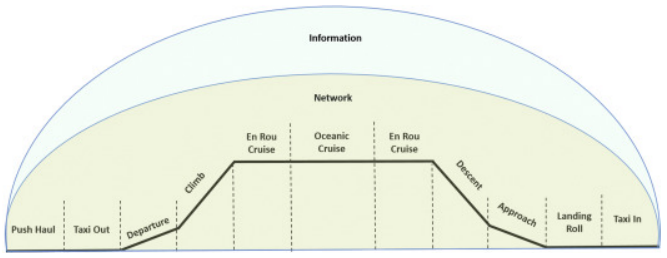


Fig. 1. Phases of flight in an aircraft.

III. Background

A typical flight consists of seven stages: takeoff, initial climb, climb, cruise, descent, final approach, landing (Figure 1). This project shall focus only on the approach phase of an aircraft's flight.

A. *Traditional Autopilot*

The Instrument Landing System (ILS) is a navigation system that uses radio signals to compute glideslopes for aircraft approach when landing, allowing automated descent until a height of 200 ft over ground level. At this height, termed the decision height, pilots must decide if visual cues are sufficient for a successful landing, or a missed approach manoeuvre is required. Category III ILS systems allow for this decision height to be as low as 100 ft [8].

This particular phase of landing is handled by the Autoland [9] part of autopilot. Only small course corrections can be tolerated at low heights, which means systems need to be highly accurate to be reliable. At low speeds near approach, the rudder loses effectiveness, so the pilot uses the steering to control, which is not connected to flight control computers.

Given the high-risk factor of aircraft systems, these vessels are usually fitted with two/three independent autopilot systems. These systems work independently, communicating their decisions and computations with each other to identify any potential system failures. Even for operations like landing, which is heavily controlled by the pilots, autopilot systems provide help only when at least two of three autopilot systems produce consistent values. This phase of landing is so critical that the overall failure probability must be less than one in a million.

B. *AI-driven autonomous systems*

Out of the 4000 airports that Airbus aircraft operate out of, fewer than 1000 have any kind of ILS, and fewer than 100 have the CAT III variant. Most landings are, like takeoff, almost fully controlled by pilots. This makes it nearly impossible to

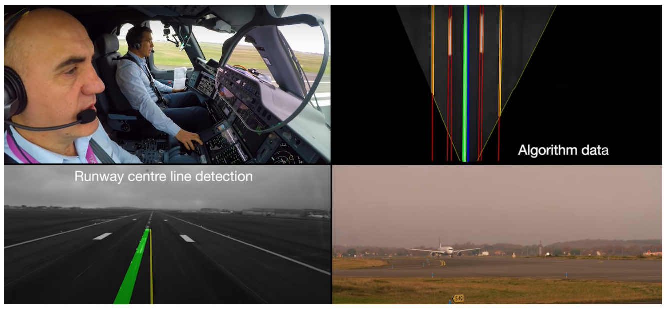


Fig. 2. Airbus Automatic Takeoff Landing (ATTOL) system, using AI to control the aircraft.

operate in condition with low visibility such as fog and heavy rain, where pilots may rely almost entirely on system readings until the last minute.

As a first step to entirely automate this process, these autopilot systems need a way to be able to see as pilots do - RGB cameras, along with IR sensors for robust vision under occlusions like snow and rain. We need object detection for drawing bounding boxes around runways and markings, and regression for estimating distance and glideslope values.

Glideslope deviation estimates are very challenging since even a 0.1 degree error is a lot when the entire glideslope range is only ± 0.7 degrees. The machine-learning driven system in development by Airbus (Figure 2) is aimed to help aircraft manage fully autonomous takeoff and landing. Similar systems aim to use classic computer-vision algorithms with cameras to enable landing on airstrips without ILS or proper supervision [10]. Using line and corner detection algorithms, these systems can be used by aircraft for alignment and distance/altitude calibration while approaching the runway.

IV. Potential Security Threats

A. Physical Access

Similar to modern automobiles, aeroplanes also use a CAN-bus architecture for communication between sensors, actuators, and onboard computers and controls [11]. Unfortunately, these systems also share the same kind of vulnerabilities: weak security protocols, virtually full control if components are compromised [12]. However, assuming physical access to aircraft is a much stronger assumption than cars. Aviation vehicles are never unattended and undergo several physical checks regularly. Thus, sneaking in such an attack would be nearly impossible.

B. Targeting Image Guidance Systems

A malicious adversary has two options to try and force a vision system in aircraft to be bypassed.

1) **Adversarial Stickers** are patterns that when printed and shown to a vision-based machine learning system, are highly likely to induce unintended behaviour [13]. These patterns generalize well across different environments, and thus could successfully fool the cameras onboard an aircraft. However, such an approach would require large stickers (since an aircraft is several miles away when it prepares to land) as well as physical access to the runway, which is possible only on small-scale airstrips.
2) **Infrared Lasers** could be pointed at both the RGB (normal) and IR cameras in such systems. Such lasers would effectively blind both sensors without affecting the pilots, thus not raising immediate suspicion. Additionally, these lasers are not visible to the human eye and would thus be hard to trace back to their origin.

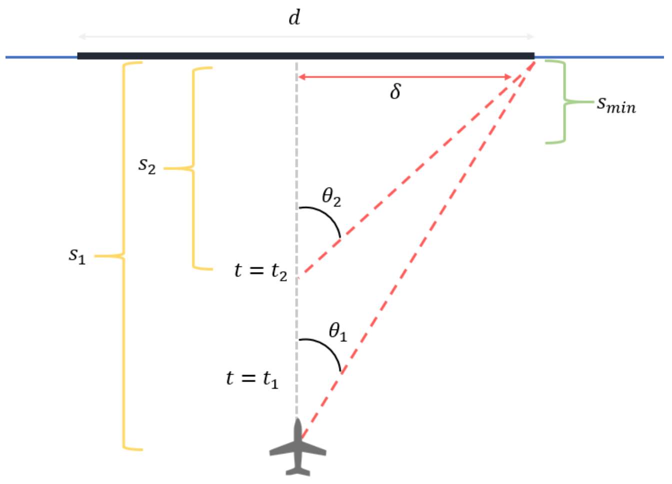


Fig. 3. Modeling the landing approach of an aircraft. Different angles of deviation are required at different distances from the landing strip.

A system like ATTOL would not be severely affected by such attacks. Even if their vision-guided systems are rendered useless, autopilot systems can always switch back to their normal mode (that is currently used) without hampering operations in any way. Even under adverse conditions like snow or heavy rain, the pilots can always make a turnaround and attempt flight on another runway, hoping to overcome the adversary's attempts.

A more sly and potentially dangerous reaction, thus, would happen with systems that rely on vision-based data for autonomous landing. Although system checks exist to limit the amount of change the system can cause within a given time frame, a carefully timed attack by the adversary could sway the aircraft when it is very close to its descent, leading to catastrophic consequences even if the system's response remains within permissible limits.

V. Setup

Assuming a simple model without any wind, the exact landing trajectory of an aircraft can be computed using its height, distance from the airstrip, and angle of approach (Figure 3). The threat model consists of an adversary that wants the aircraft to ultimately sway in one particular direction, away from the landing strip. The adversary considers success when the aircraft sways totally outside the landing strip, *i.e.* $\delta \geq \frac{d}{2}$.

A deviation at any given distance from the airport would lead to a proportionate deviation from the centre of the landing

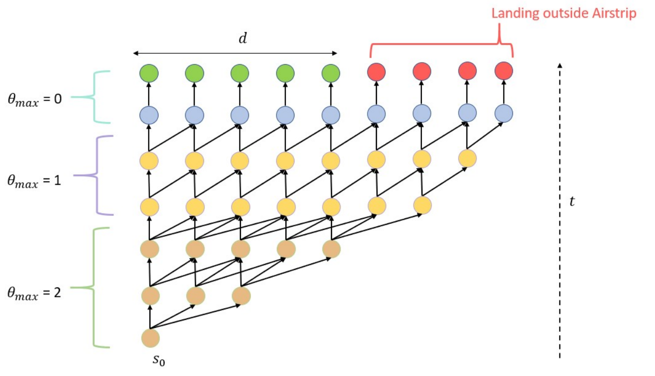


Fig. 4. Internal states while modeling deviations caused by the adversary, with limited angle deviations as the aircraft gets closer to landing. Sink nodes are used to identify success and failure nodes, which is then used to construct a DTMC.

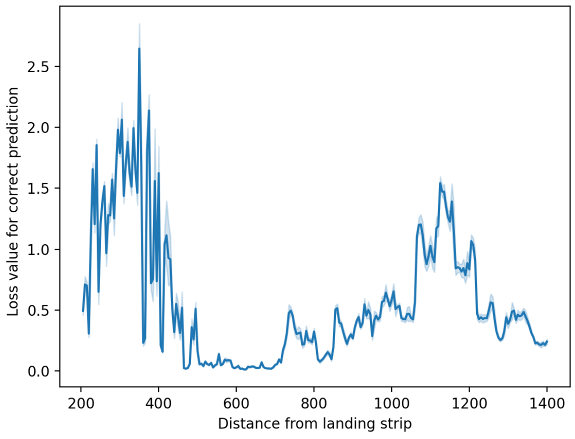


Fig. 5. Loss value of the classification system as a (direct) distance of the aircraft from the landing strip across 20 runs (mean in dark, deviation in light).

strip. Using an internal representation of states that keeps track of the total deviation so far, a graph of transitions can be constructed from the starting node (Figure 4). Then, by checking which nodes indeed satisfy the adversary's goal, auxiliary nodes can be constructed: all nodes that lead to success converge to one node, and the others to the second node. Using this DTMC, the following property verification can be used to compute the probability of adversary's success:

$$P = ?[F \ s = Z \ \& \ d = 1] \tag{1}$$

, where Z is the final node to which all nodes converge, and $d = 1$ is a sink node for all nodes that correspond to the adversary's success. To incorporate potential system checks in an aircraft that limit aircraft sway at a particular altitude, upper limits on deviation angles proportional to height are implemented in all simulations: the highest being $0.4°$ when furthest from the airport, and no deviation at all below a certain altitude. Consequently, the probability of deviation for a certain angle is quantized and set inversely proportional to the angle. So a probability of failure 0.75, with possible angles $1°, 2°, 3°$ is set as $0.75 * 0.54, 0.75 * 0.27, 0.75 * 0.19$ respectively. Additionally, a cut-off height can be incorporated into the implementation to simulate the role of a human that can take over manual control below a certain altitude. Beyond this height, no additional deviation is allowed by the adversary.

VI. Experiments

The project can be divided into three main phases: estimation of the adversary's strength with distance, running an actual flight landing simulation and using model verification software to estimate the probability of failure. Implementation is available at https://github.com/iamgroot42/cps_project

A. Adversary Simulation

To simulate the landing of an aircraft and how its view would warp the adversarial patch as seen by the aircraft, perspective transforms are applied on a single image of the landing strip perturbed with a patch. Since the adversary wants to optimize the patch to cause failure at all distances, the generation process must be differentiable. The adversary's goal is to find some patch x such that, for varying distances from the landing strip:

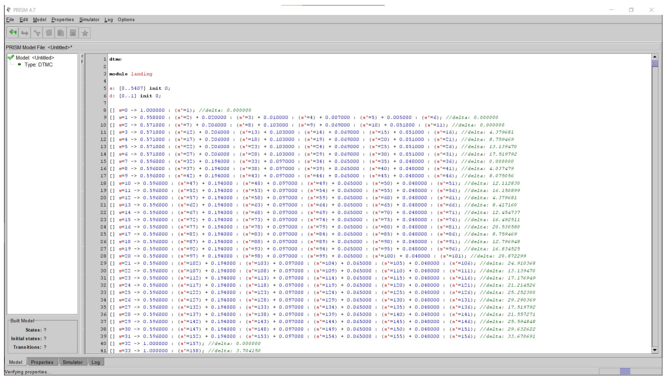
$$\sum_{i=1}^{n} l(A_2 \times (\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dist_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \times (R \times A_1)) \times x) \tag{2}$$

is maximized, where $l()$ is a loss function that increases when the victim sways off-course. A_1, A_2 here correspond to constant transform matrices for a homography. To add variations that might occur because of wind, I also modified some of the angles slightly randomly while generating flight paths. Since there are no open models or datasets available for landing strip detection, I used an image classification model for ImageNet [14], with the adversarial loss corresponding to the adversary causing a change in classification. I used PyTorch along with Kornia [15] to produce differentiable perspective transforms, along with a custom implementation for adversarial patch generation.

After generating the patch, the loss value is measured as a function of distance from the aircraft. Based on multiple runs of the generation process, a lookup-base function can be created to estimate the success of failure at any given distance. As visible in Figure 5, this function is not non-decreasing with distance: a behaviour that can perhaps be explained by how varying angles and scale could influence the adversary's chances of success.

B. Flight Landing Simulation

Instead of using complicated flight simulators to get landing paths, I used the OpenAP [16] toolkit to access flight path



```
dtmc

module landing

s: [0..5407] init 0;
d: [0..1] init 0;

[] s=0 -> 1.000000 : (s'=1); //delta: 0.000000
[] s=1 -> 0.958000 : (s'=2) + 0.020000 : (s'=3) + 0.010000 : (s'=4) + 0.007000 : (s'=5) + 0.005000 : (s'=6); //delta: 0.000000
[] s=2 -> 0.571000 : (s'=7) + 0.206000 : (s'=8) + 0.103000 : (s'=9) + 0.069000 : (s'=10) + 0.051000 : (s'=11); //delta: 0.000000
[] s=3 -> 0.571000 : (s'=12) + 0.206000 : (s'=13) + 0.103000 : (s'=14) + 0.069000 : (s'=15) + 0.051000 : (s'=16); //delta: 4.379681
[] s=4 -> 0.571000 : (s'=17) + 0.206000 : (s'=18) + 0.103000 : (s'=19) + 0.069000 : (s'=20) + 0.051000 : (s'=21); //delta: 8.759469
[] s=5 -> 0.571000 : (s'=22) + 0.206000 : (s'=23) + 0.103000 : (s'=24) + 0.069000 : (s'=25) + 0.051000 : (s'=26); //delta: 13.139470
[] s=6 -> 0.571000 : (s'=27) + 0.206000 : (s'=28) + 0.103000 : (s'=29) + 0.069000 : (s'=30) + 0.051000 : (s'=31); //delta: 17.519792
[] s=7 -> 0.596000 : (s'=32) + 0.194000 : (s'=33) + 0.097000 : (s'=34) + 0.065000 : (s'=35) + 0.048000 : (s'=36); //delta: 0.000000
[] s=8 -> 0.596000 : (s'=37) + 0.194000 : (s'=38) + 0.097000 : (s'=39) + 0.065000 : (s'=40) + 0.048000 : (s'=41); //delta: 4.037479
[] s=9 -> 0.596000 : (s'=42) + 0.194000 : (s'=43) + 0.097000 : (s'=44) + 0.065000 : (s'=45) + 0.048000 : (s'=46); //delta: 8.075056
[] s=10 -> 0.596000 : (s'=47) + 0.194000 : (s'=48) + 0.097000 : (s'=49) + 0.065000 : (s'=50) + 0.048000 : (s'=51); //delta: 12.112830
[] s=11 -> 0.596000 : (s'=52) + 0.194000 : (s'=53) + 0.097000 : (s'=54) + 0.065000 : (s'=55) + 0.048000 : (s'=56); //delta: 16.150899
[] s=12 -> 0.596000 : (s'=57) + 0.194000 : (s'=58) + 0.097000 : (s'=59) + 0.065000 : (s'=60) + 0.048000 : (s'=61); //delta: 4.379681
[] s=13 -> 0.596000 : (s'=62) + 0.194000 : (s'=63) + 0.097000 : (s'=64) + 0.065000 : (s'=65) + 0.048000 : (s'=66); //delta: 8.417160
[] s=14 -> 0.596000 : (s'=67) + 0.194000 : (s'=68) + 0.097000 : (s'=69) + 0.065000 : (s'=70) + 0.048000 : (s'=71); //delta: 12.454737
[] s=15 -> 0.596000 : (s'=72) + 0.194000 : (s'=73) + 0.097000 : (s'=74) + 0.065000 : (s'=75) + 0.048000 : (s'=76); //delta: 16.492511
[] s=16 -> 0.596000 : (s'=77) + 0.194000 : (s'=78) + 0.097000 : (s'=79) + 0.065000 : (s'=80) + 0.048000 : (s'=81); //delta: 20.530580
[] s=17 -> 0.596000 : (s'=82) + 0.194000 : (s'=83) + 0.097000 : (s'=84) + 0.065000 : (s'=85) + 0.048000 : (s'=86); //delta: 8.759469
[] s=18 -> 0.596000 : (s'=87) + 0.194000 : (s'=88) + 0.097000 : (s'=89) + 0.065000 : (s'=90) + 0.048000 : (s'=91); //delta: 12.796948
[] s=19 -> 0.596000 : (s'=92) + 0.194000 : (s'=93) + 0.097000 : (s'=94) + 0.065000 : (s'=95) + 0.048000 : (s'=96); //delta: 16.834525
[] s=20 -> 0.596000 : (s'=97) + 0.194000 : (s'=98) + 0.097000 : (s'=99) + 0.065000 : (s'=100) + 0.048000 : (s'=101); //delta: 20.872299
[] s=21 -> 0.596000 : (s'=102) + 0.194000 : (s'=103) + 0.097000 : (s'=104) + 0.065000 : (s'=105) + 0.048000 : (s'=106); //delta: 24.910368
[] s=22 -> 0.596000 : (s'=107) + 0.194000 : (s'=108) + 0.097000 : (s'=109) + 0.065000 : (s'=110) + 0.048000 : (s'=111); //delta: 13.139470
[] s=23 -> 0.596000 : (s'=112) + 0.194000 : (s'=113) + 0.097000 : (s'=114) + 0.065000 : (s'=115) + 0.048000 : (s'=116); //delta: 17.176949
[] s=24 -> 0.596000 : (s'=117) + 0.194000 : (s'=118) + 0.097000 : (s'=119) + 0.065000 : (s'=120) + 0.048000 : (s'=121); //delta: 21.214526
[] s=25 -> 0.596000 : (s'=122) + 0.194000 : (s'=123) + 0.097000 : (s'=124) + 0.065000 : (s'=125) + 0.048000 : (s'=126); //delta: 25.252300
[] s=26 -> 0.596000 : (s'=127) + 0.194000 : (s'=128) + 0.097000 : (s'=129) + 0.065000 : (s'=130) + 0.048000 : (s'=131); //delta: 29.290369
[] s=27 -> 0.596000 : (s'=132) + 0.194000 : (s'=133) + 0.097000 : (s'=134) + 0.065000 : (s'=135) + 0.048000 : (s'=136); //delta: 17.519792
[] s=28 -> 0.596000 : (s'=137) + 0.194000 : (s'=138) + 0.097000 : (s'=139) + 0.065000 : (s'=140) + 0.048000 : (s'=141); //delta: 21.557271
[] s=29 -> 0.596000 : (s'=142) + 0.194000 : (s'=143) + 0.097000 : (s'=144) + 0.065000 : (s'=145) + 0.048000 : (s'=146); //delta: 25.594848
[] s=30 -> 0.596000 : (s'=147) + 0.194000 : (s'=148) + 0.097000 : (s'=149) + 0.065000 : (s'=150) + 0.048000 : (s'=151); //delta: 29.632622
[] s=31 -> 0.596000 : (s'=152) + 0.194000 : (s'=153) + 0.097000 : (s'=154) + 0.065000 : (s'=155) + 0.048000 : (s'=156); //delta: 33.670691
[] s=32 -> 1.000000 : (s'=157); //delta: 0.000000
[] s=33 -> 1.000000 : (s'=158); //delta: 3.704150
```

Fig. 6. Example of some DTMC model code generated using the intermediate states in Python scripts, parsed and stored as a .pm file for PRISM.

models: simulations based on actual parameters from engine and plane schematics. To assess the generalization of our analyses across aircraft, flight paths are generated for multiple aircraft (Airbus A380, Airbus A320, Boeing 787) multiple times (three per simulation).

C. Model Verification

Once the simulation has run, the generated DTMC model can be passed to PRISM to compute the probability of its success (Figure 6). The generated DTMCs had around 2000-8000 nodes, which can probably be reduced by clubbing similar-valued states together. I performed two forms of analyses: studying the probability of an adversary succeeding in its task for varying values of δ, as well as cutoff heights.

Cut-off Height (m)	δ (m)	P(failure)
0	10	0.204 ± 0.038
0	15	0.095 ± 0.027
0	20	0.046 ± 0.012
10	20	0.020 ± 0.010
20	20	0.022 ± 0.021
50	20	0.043 ± 0.028
100	20	0.0257 ± 0.025
200	20	0.007 ± 0.013
400	20	0 ± 0

TABLE I
PROBABILITY OF ADVERSARY SUCCESS, VARYING WITH CUT-OFF HEIGHTS AND δ DEVIATIONS

D. Results

As visible in Table I, the probability of absolute failure (landing outside the landing strip) is quite significant, and simply unacceptable compared to the kind of guarantees required by aircraft systems (in the orders of $1e^{-}5$. In fact, with probability ~ 0.2, the adversary can cause a deviation of 10 metres off the centre line of the landing strip, which can have severe consequences. Thus, this threat is very much real when using autonomous systems.

Looking at the cut-off height, the trend is in line with the observer adversary success rates with varying distances. The trend is not strictly non-decreasing with height. A cut-off height of 50 metres can be more harmful than no cut-off height at all. Even a system that allows no deviations below a height of 200 metres $\sim$ 650 feet) has a 0.007 chance of landing outside the airstrip, which is not negligible. It is clear from these simulation results that even a limited adversary, without any physical access to the aircraft, can cause damage with a non-trivial probability. Only when the cut-off height is significantly high, which is best implemented via human supervision, does the probability disappear into negligible values.

VII. Challenges

Multiple challenges limit the scope of both theoretical and experimental analyses of autopilot systems.

There are no publicly available datasets for plane-view landing cameras. Even sophisticated systems currently in development use costly simulation software with texture shaders and processing to generate datasets to train their machine learning models [17]. Thus, running a precise proof-of-concept of potential adversarial attacks was impossible.

Most design blueprints for aircraft are excessively complicated, with control systems handling sensor data from hundreds of sensors concurrently, all while making adjustments in rudders, flaps, thrust based on external conditions like wind direction and speed. Modelling all of those systems, even for just the landing phases, required extensive background knowledge of aerodynamics as well as these systems, and is not in the scope of this project.

VIII. Conclusion

AI-aided solutions for commercial autonomous flight are in very early stages. Nonetheless, it is critical to think ahead and identify potential shortcomings in their implementation and integration. Via a controlled simulation, we can see how an input-based adversary can cause damage with a non-trivial probability of success. Thus, switching to autonomous vision-aided systems is a bad idea until the problem of adversarial examples for machine learning systems can be fully solved. Until then, these systems can be used as a potential replacement for ILS, along with proper human supervision.

References

[1] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[2] D. F. Alqushayri, "Cybersecurity vulnerability analysis and countermeasures of commercial aircraft avionic systems," *ERAU Scholarly Commons*, 2020.

[3] K. Sampigethaya and R. Poovendran, "Aviation cyber–physical systems: Foundations for future aircraft and air transport," *Proceedings of the IEEE*, vol. 101, no. 8, pp. 1834–1855, 2013.

[4] "Ils trap 'may have caused' korean boeing 767 to crash into guam hillside," https://www.flightglobal.com/ils-trap-may-have-caused-korean-boeing-767-to-crash-into-guam-hillside-/38768.article, accessed: 2021-04-28.

[5] "Tracking airplanes: how flightradar24 works," https://www.kaspersky.com/blog/tracking-airplanes-how-flightradar24-works/8389/, accessed: 2021-04-28.

[6] A. J. Moore, M. Schubert, C. Dolph, and G. Woodell, "Machine vision identification of airport runways with visible and infrared videos," *Journal of Aerospace Information Systems*, vol. 13, no. 7, pp. 266–277, 2016.

[7] "Is autonomy the future of aerial mobility?" https://www.airbus.com/newsroom/stories/autonomy-aerial-mobility.html, accessed: 2021-03-29.

[8] "Instrument landing system - wikipedia," https://en.wikipedia.org/wiki/Instrument_landing_system, accessed: 2021-05-01.

[9] "Autoland - wikipedia," https://en.wikipedia.org/wiki/Autoland, accessed: 2021-04-29.

[10] M. E. Kügler, N. C. Mumm, F. Holzapfel, A. Schwithal, and M. Angermann, "Vision-augmented automatic landing of a general aviation fly-by-wire demonstrator," in *AIAA Scitech 2019 Forum*, 2019, p. 1641.

[11] "Investigating can bus network integrity in avionics systems," https://www.rapid7.com/research/report/investigating-can-bus-network-integrity-in-avionics-systems/, accessed: 2021-04-30.

[12] K. Koscher, S. Savage, F. Roesner, S. Patel, T. Kohno, A. Czeskis, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2010, pp. 447–462.

[13] J. Li, F. Schmidt, and Z. Kolter, "Adversarial camera stickers: A physical camera-based attack on deep learning systems," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3896–3904.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[15] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: an open source differentiable computer vision library for pytorch," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 3674–3683.

[16] J. Sun, J. M. Hoekstra, and J. Ellerbroek, "Openap: An open-source aircraft performance model for air transportation studies and simulations," *Aerospace*, vol. 7, no. 8, p. 104, 2020.

[17] "How wayfinder is using neural networks for vision-based autonomous landing," https://acubed.airbus.com/blog/wayfinder/how-wayfinder-is-using-neural-networks-for-vision-based-autonomous-landing/, accessed: 2021-04-29.