The OODA Loop of Cloudlet-based Autonomous Drones

Mihir Bala, Aditya Chanana, Xiangliang Chen, Qifei Dong, Thomas Eiszler, Jingao Xu,
Padmanabhan Pillai†, Mahadev Satyanarayanan
Carnegie Mellon University and †Intel Labs

***Abstract*—We present a benchmark-driven experimental study of autonomous drone agility relative to edge offload pipeline attributes. This pipeline includes a monocular gimbal-actuated on-drone camera, hardware RTSP video encoding, 4G LTE wireless network transmission, and computer vision processing on a ground-based GPU-equipped cloudlet. Our parameterized and reproducible agility benchmarks stress the OODA ("Observe, Orient, Decide, Act") loop of the drone on obstacle avoidance and object tracking tasks. We characterize the latency and throughput of components of this OODA loop through software profiling, and identify opportunities for optimization.**

***Index Terms*—edge computing, machine learning, computer vision, drones, autonomous robotics, wireless networks, benchmarks, latency, throughput**

I. Introduction

Fully autonomous drone flight is defined as "pre-programmed flight without a remote human pilot, including mission-specific actions in response to runtime observations" [1]. Achieving this on an ultralight drone via edge offload was first demonstrated in 2023 [2]. By overcoming the weight and size limitations of on-board drone intelligence, edge offload inspires the vision of small, cheap, and lightweight but brilliant drones operating in crowded urban settings without a human pilot. This is attractive from a public safety and regulatory approval viewpoint because the kinetic energy of such drones is much lower than that of larger and heavier drones [3]. It is also attractive from a business viewpoint because the most expensive part of drone operations today is the trained human pilot who has to continously fly the drone [4]. We focus on payload-free active vision tasks [5], [6] such as recognizing and tracking a target, rather than tasks such as merchandise delivery that involve substantial payloads.

In this paper, we explore how close we are today to making this vision a commercial reality. If an entirely new ecosystem of custom-built drones and low-latency wireless networks is needed, the path to commercialization will be long and risky. On the other hand, if cloudlet-based deployments with commercial off-the-shelf (COTS) components available today can be integrated into a working system that meets the performance and agility demands of real-world use cases, the vision is well within reach. We therefore ask:

"Using COTS ultralight drones, 4G LTE wireless networks, and cloudlet hardware, is the end-to-end performance of edge offload sufficient for real-world active vision tasks?"

To answer this question, we introduce the concept of a drone *OODA Loop*. Originally conceived in the 1950s to characterize man-machine symbiosis in combat aircraft, this concept has since been extended to many other domains [7]. The components of an OODA loop ("Observe", "Orient," "Decide," and "Act") define the stages of any reactive pipeline that involves a human in the loop. In this paper, we extend this concept from its cyber-human origins to the cyber-physical context of an autonomous drone. Viewing such an AI pipeline through the lens of an OODA loop enables us to better understand its performance attributes. It enables us to tease apart latency and throughput limitations at fine granularity, thus enabling bottlenecks to be identified and optimized.

An OODA loop's attributes limit what a drone can do. Because of throughput limitations, closely-spaced real-world events may not be resolvable as separate events. A moving target with a jerky motion will appear to move smoothly. Large, but brief, deviations from the smoothed path may not be detected. High end-to-end latency will also hurt agility. Predictive approaches in drone or gimbal actuation will be needed for tracking fast-moving targets. This leads to greater likelihood of errors and loss of target due to mis-prediction.

A cloudlet-based drone OODA loop includes: (a) on-drone sensing, (b) on-drone pre-processing, (c) transmission to cloudlet, (d) processing on a (possibly multi-tenant) cloudlet, (e) transmission to drone, (f) on-drone post-processing, and (g) drone actuation. Through latency profiling and bottleneck optimization, we characterize the OODA loop of the pipeline described by Bala et al [2]. We then define parameterized and reproducible benchmarks to quantify its agility.

This paper makes four contributions:

- First, it introduces the concept of a drone OODA loop.
- Second, it profiles latency in a cloudlet-based drone OODA loop, and shows how it can be optimized.
- Third, it describes parameterized and reproducible agility benchmarks that mimic real-world active vision.
- Fourth, it presents experimental results that quantify agility relative to OODA loop latency and throughput.

II. Background and Related Work

A. Networked Control Systems

Over the last twenty years, networked control systems have seen significant advancements within the field of robotics. As described in the book *Networked Control Systems: Theory and Applications* by Wang et al, a networked control system (NCS) is a system that "consists of sensors, actuators, and controllers whose operations are distributed at different geographical locations and coordinated through information exchanged over communication networks" [8]. NCS have been

2837-4827/24/$31.00 ©2024 IEEE
DOI 10.1109/SEC62691.2024.00022

used in many robotics applications, from planet exploring rovers [9] to driverless vehicles [10]. Recently, they have also been applied to drones [11]–[14]. In these systems, drones use a ground station in conjunction with other nearby aircraft to offload high compute loads. However, previous work either focused on the theory of how such a system should be designed or on solutions using custom components. In contrast, our work explores whether COTS-based solutions can be made sufficiently performant for edge offload in real drone flight. A specific contribution of the current paper is to deeply analyze the performance limitations of such a COTS-based solution.

B. The OODA Loop

In his 1986 work *Patterns of Conflict* [15], Colonel John Boyd formalized the concept of an OODA loop which had been used since the 1950s to study a pilot's reaction time in combat. Since then, it has been used to model reaction time in a variety of military settings. In recent years, some research has started using the OODA loop to analyze human-AI hybrid decision-making pipelines [16], [17]. As mentioned earlier (§I), a key contribution of this paper is the extension of the OODA loop concept from its cyber-human origin to the cyber-physical context of an autonomous drone.

C. Drone Benchmarks

There have been many efforts in the computer vision and machine learning community to create benchmarks for comparing drone performance on specific tasks. These focus exclusively on the accuracy of algorithms such as drone-based object tracking and face recognition, ignoring system attributes such as agility and end-to-end processing latency. Du et al [18], Li et al [19], Kalra et al [20], and Zhao et al [21] are examples of this genre.

Many drone benchmarks do measure agility but involve only simulated tests. MAVBench [22] is one popular example. It consists of a closed-loop simulator and end-to-end application benchmark suite of five workloads pertaining to scanning, aerial photography, package delivery, 3D Mapping, and Search and Rescue. These workloads lack customization options, and often represent a specific simulated scenario which can only give limited perspective on real performance. Another simulation-based benchmark, FlightBench [23], has agility workloads which provide several levels of difficulty. However, this difficulty is determined arbitrarily by the authors and the obstacle courses are too complex to practically replicate outside of the simulator. Additionally, simulations typically do not fully capture real flight performance, where sensors can experience noise which can influence actuation.

There are live flight drone benchmarks that measure agility, but they are not as common. One example is the disturbance benchmark proposed by Wu et al [24] which uses an indoor course along with a fan to emulate obstacle avoidance in windy conditions. The course is fixed and does not provide guidance for replicating the described experiments. For this reason, while it is useful for evaluating the paper's proposed trajectory planner, it is not as useful for measuring the performance differences between different avoidance methods.

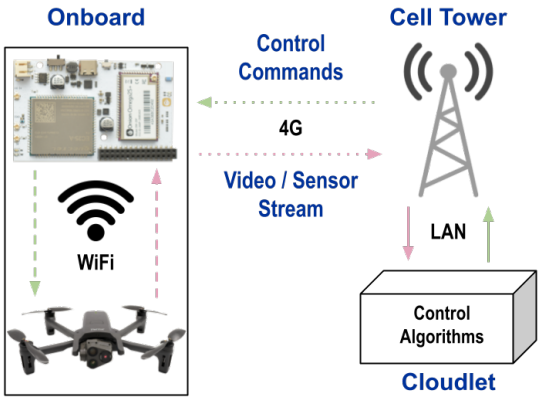


Fig. 1: Edge Offload Pipeline

Koubaa et al [25] describe an experimental study that compares on-board drone processing versus offloading to the cloud. The metrics of interest in their work are energy cost, bandwidth demand, and timeliness of results. The last of these metrics is closest to our focus on the agility of drones. However, the experiments described do not include drone actuation in response to real-time observations. They are purely open loop experiments, with timeliness to cloud users being the metric of interest. Further, this work only provides microbenchmarks to evaluate these metrics. There are no end-to-end benchmarks that include the full OODA pipeline of sensing, processing and drone actuation.

Beyond these experimental efforts is a vast body of published literature on analytical or simulation-based evaluations of algorithms for specific drone tasks. Examples include the work of Chen et al [26], Hayat et al [27], Wang et al [28], and Wu et al [29]. These studies abstract away the physical drone, relying instead on hypothetical cost models of processing and communication. AdaDrone [30] is a slightly more realistic approach that leverages a drone simulator. None of these efforts use real drones, with their intrinsic limitations of weight and maneuverability. In contrast to these prior works, the focus of this paper is on providing *parameterized* and *reproducible* benchmarking of drones in actual flight.

III. Profiling & Optimizing the OODA Loop

We replicate the setup (Figure 1) described by Bala et al [2] using a slightly larger and heavier variant (550 g vs. 360 g) of the Parrot ANAFI drone. This variant is approved on the BlueUAS list [31] for US government applications. As payload, we use the 53 g Onion Omega 2 LTE [32] single board computer which was briefly described in that paper. The Onion Omega receives the sensor stream from the drone over WiFi and forwards it to the cloudlet over 4G LTE. The resulting computation for Orient and Decide (e.g. DNN model inference) causes a control message to be sent back via 4G LTE to the Onion Omega, and thence over WiFi to the drone.

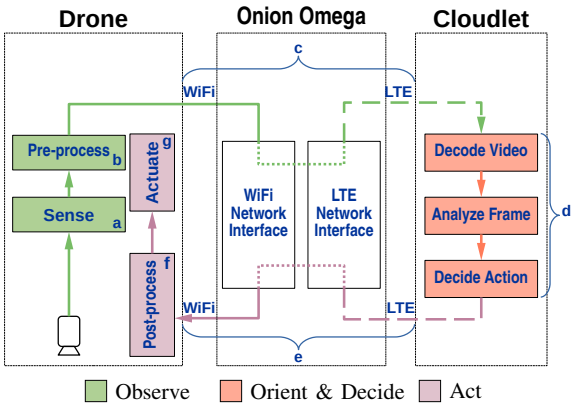
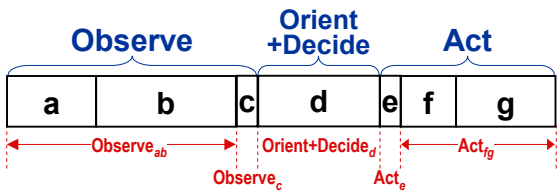


Fig. 2: Detailed View of Our OODA Loop



Only items in red above can be measured.

a = on-drone sensing
b = on-drone pre-processing
c = transmission to cloudlet
d = processing on cloudlet
e = transmission to drone
f = on-drone post-processing
g = drone actuation

Fig. 3: Measurable Components of Our OODA Loop

The drone to cloudlet pipeline is used for both data plane and control plane operations. Its intrinsic end-to-end performance defines the experimental baseline. To explore pipeline degradation, we add network latency using FireQoS [33] and drop frames to throttle bandwidth.

A. Mapping the OODA Loop

Figure 2 maps our end-to-end pipeline to OODA loop components. Components (a), (b) and (c) together map to the "Observe" phase; component (d) maps to its "Orient" and "Decide" phases; and, components (e), (f) and (g) together map to the "Act" phase. Due to closed-source restrictions of our COTS pipeline, some OODA loop components have to be aggregated for purposes of measurement, as shown by Figure 3. Total end-to-end latency is given by the sum of these components; total throughput is that of its bottleneck.

The earliest point in the pipeline where software instrumentation can be inserted is between the Wi-Fi and LTE interfaces on the Onion Omega. The Wi-Fi part of this pipeline is thus attributed to Observe$_{ab}$ rather than to Observe$_c$. Similarly, on the return path, the Wi-Fi part is attributed to Act$_{fg}$ rather than to Act$_e$. Only LTE transmission is attributed to Observe$_c$ and Act$_e$. The resulting error is likely to be very small since Wi-Fi is much more performant than LTE. We present our detailed measurements in §III-B to §III-F.

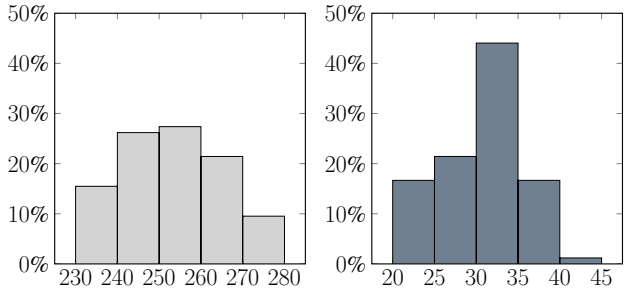


Fig. 4: Observe$_{ab}$ Measurements

B. Observe$_{ab}$

Our drone is a commercial product that uses black box hardware and software to seamlessly integrate (a) and (b). Its camera creates a stream of raw video frames. On-board processing transforms these raw frames into a sequence of UDP packets that slice-encode a 720p H.264 RTSP video stream at 30 fps [34]. Neither the resolution nor frame rate of this video stream are configurable. The slice encoding aims to reduce the visual impact of UDP packet loss. The black box nature of this transformation makes attribution of latency costs difficult. It is not possible to insert instrumentation to separate (a) and (b); they merge into an indivisible component.

Figure 4 presents our measurements. The latency distribution has a mean of 253 ms, with a standard deviation of 12 ms and a p99 of 277 ms. Instantaneous throughput has a mean of 31 fps, with a standard deviation of 5 fps and a p1 of 22 fps. Due to streaming, throughput can be higher than the reciprocal of latency. The short WiFi Observe$_{ab}$ segment is partly responsible for this observed variation.

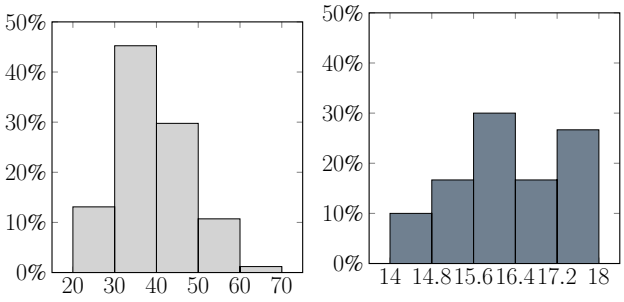
C. Observe$_c$

As Figure 1 illustrates, the wireless network path from drone to cloudlet consists of a very short Wi-Fi segment, transit through the Onion router carried as payload, and then a longer 4G LTE segment to the cloudlet. Figure 5 presents the latency and throughput distributions of Observe$_c$. Its latency has a mean of 39 ms, with a standard deviation of 8 ms and a p99 of 60 ms. Instantaneous throughput has a mean of 16.3 Mbps, with a standard deviation of 1 Mbps and a p1 of 14.4 Mbps. Since 720p video at 30 fps only demands an average bit rate of about 6.5 Mbps [35], Observe$_c$ is definitely not the bottleneck.

D. Orient+Decide$_d$

Processing on the cloudlet involves three stages:

- Stage-1: Decoding the UDP packet stream to produce individual frames from H.264 video.
- Stage-2: Application-specific processing of each frame to interpret its contents. For example, this could involve DNN inferencing with a pre-trained model to detect objects of interest currently visible to the drone.



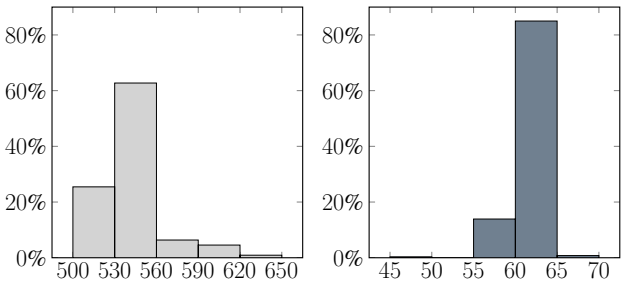
Mean: 39 ±8 ms p99: 60 ms

(a) Latency (ms)

Mean: 16.3±1 Mbps p1: 14.4Mbps

(b) Throughput (Mbps)

Fig. 5: Observe$_c$ Measurements



Mean: 541 ±22 ms p99: 620 ms

(a) Latency (ms)

Mean: 62 ±1.5 fps p1: 59 fps

(b) Instant. Throughput (fps)

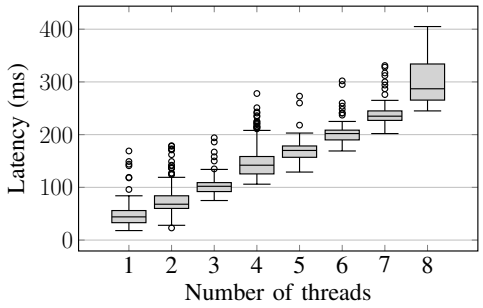
Fig. 6: Original FFmpeg-based Stage-1 Performance

- Stage-3: Application-specific logic to determine salient changes revealed by Stage-2. This early part of Stage-3, together with Stage-1 and Stage-2, constitute the "Orient" part of the OODA loop. The rest of Stage-3 is the "Decide" part. Drone actuation (if any) is determined, and the command to perform this actuation is generated. For example, Stage-2 may show that an object being tracked has moved and the gimbal has to be adjusted to re-center the object in the camera's field of view (FOV).

Stage-2 can be viewed as perception and Stage-3 as cognition. The latency and throughput of Stage-1 constrain the performance of Orient+Decide$_d$ since decoding has to be performed even if Stage-2 and Stage-3 take a negligible amount of time.

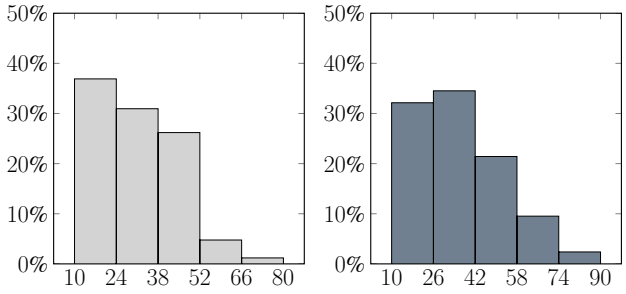
Figure 6 presents our measurements of Stage-1. We were surprised by the magnitude of the latency, with a mean of 541 ms. Our cloudlet has two Intel Xeon processors with a total of 36 cores, 128GB of RAM and an NVIDIA GeForce GTX 1080 Ti GPU. This should be ample for efficient software decoding of an H.264 video stream, as confirmed by the mean throughput of 62 fps shown in Figure 6(b). The high latency observed has no obvious explanation, but it has a large negative impact on the OODA pipeline. As detailed elsewhere [36], we determined that the culprit was negative latency scaleout of FFmpeg when performing single stream decoding (Figure 7).

By switching to different decoding software [37], we were able to reduce the latency from a mean of 541 ms in Figure 6(a) to a mean of 32 ms in Figure 8(a). This has been achieved with a mean throughput of 37 fps (Figure 8(b)), which is well above the demand of 31 fps from Observe$_{ab}$. Assuming negligible processing in Stage-2 and Stage-3, Figure 8 shows the best-case latency and throughput of Orient+Decide$_d$.



Each box extends from the first quartile (Q_1) to the third quartile (Q_3), with a line at the median. Whiskers extend from the box to the farthest data point lying within 1.5x the inter-quartile range ($IQR = Q_3 - Q_1$) from the box. Circles represent outliers.

Fig. 7: Negative Scale-out of FFmpeg Latency



Mean: 32 ±13 ms p99: 59 ms

(a) Latency (ms)

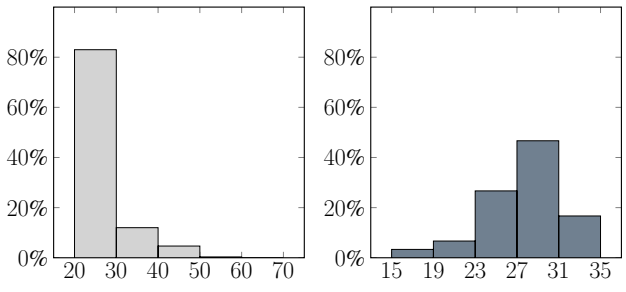
Mean: 37 ±16 fps p1: 17 fps

(b) Instant. Throughput (fps)

Fig. 8: Improved Performance With FFmpeg Alternative

E. Act$_e$

Figure 9 presents our measurements of the wireless network path from cloudlet to drone. The latency has a mean of 30 ms, with a standard deviation of 4 ms and a p99 of 49 ms. The throughput has a mean of 28 Mbps, with a standard



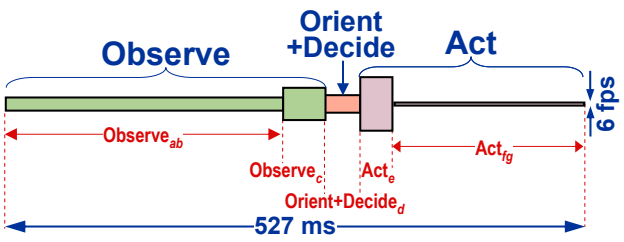
Mean: 30 ±4 ms p99: 49 ms

(a) Latency (ms)

Mean: 28±3.7 Mbps p1: 19 Mbps

(b) Throughput (Mbps)

Fig. 9: Act$_e$ Measurements



This figure uses the same notation as Figure 3, and color coding as Figure 2. Width is scaled to represent latency, and height to represent throughput. Orient+Decide$_d$ here assumes negligible application-specific processing. For the electromechanical actuation represented by component Act$_{fg}$, the reciprocal of its latency is used as its throughput.

Fig. 11: OODA Loop Latency & Throughput

deviation of 3.7 Mbps and a p1 of 19 Mbps. Since no video is transmitted back to the drone, Act$_e$ is not a bottleneck.

F. Act$_{fg}$

Black box hardware and software on the drone seamlessly integrate components (f) and (g). All that is externally visible is a set of commands that are accessible via the drone's SDK [38]. The processing of a command and initiation of actuation are integrated into Act$_{fg}$ in Figure 3. In this context, latency corresponds to the time difference between the receipt of an actuation command by the drone, and the start of actuation. To measure this difference, we position the stationary drone in front of a display connected to the cloudlet. The display shows the current timestamp in milliseconds. We send a command to the drone to move its camera gimbal, while recording the display and gimbal using a slow-motion video camera. In post-processing, we manually identify the timestamp of the command and that of the first video frame showing gimbal movement. Act$_{fg}$ latency is the difference between these two timestamps. Our slow-motion camera operates at 240 fps, resulting in a frame interval of ˜4 ms. Our measurement has an error margin of ˜5 frames, translating to experimental error of ˜20 ms.

Run	Latency (ms)
1	188
2	170
3	162
4	189
5	155
Mean	173 ±15

Fig. 10: Act$_{fg}$ Latency

Figure 10 presents our measurements. The latency distribution has a mean of 173 ms, with a standard deviation of 15 ms. Electromechanical actuation is far slower than processing or network transmission, and there is no concept of streaming. Our benchmarks do not involve any back-to-back actuations without intervening sensing and processing. Hence, throughput is best interpreted as the reciprocal of latency.

G. The Full OODA Loop

Using the same notation as Figure 3, a visual summary of the measurements reported in §III-B to §III-F is shown in Figure 11. This captures the best-case OODA loop, where no time is spent in Stage-2 and Stage-3 of Orient+Decide$_d$. In practice, it is those stages that perform the processing for drone autonomy such as object detection, object tracking, Kalman filtering, and route planning. They also do the processing to generate the commands for drone actuation such as gimbal movement, flight path alteration, or altitude change. The height and width of the resulting Orient+Decide$_d$ component in Figure 11 would need to be scaled to include such application-specific processing. In some cases, that component may dominate the entire OODA loop.

In a typical application, many iterations of the OODA loop may involve no actuation, thus eliminating Act$_e$ and Act$_{fg}$. For example, consider a target that is moving in a straight line at constant speed. Successive OODA loops of a drone that is following that target only need to confirm that it remains centered in the FOV. Only abrupt change of motion by the target will stress the OODA loop. Fast reaction is then needed to discover that the target is off-center, and to actuate the gimbal or drone to re-center it before it is lost from the FOV. Figure 11 shows that the latency and throughput of Observe$_{ab}$ are the limiting constraint in uneventful settings. It is thus the inherent attributes of the drone, rather than network bandwidth or the cloudlet processing power, that limits us today.

IV. Measuring Agility

"What is the agility of an autonomous drone whose OODA *loop is shown in Figure 11?"* That is the central question of interest. Unfortunately, answering that question is difficult because we do not have a yardstick for measuring agility in outdoor flight settings. Although drone racing is popular, the races are typically tests of human piloting skill. There is no reproducible metric for outdoor autonomous drone agility that defines the units in which to express an answer.

Agility is a complex emergent cyber-physical property that depends both on cyber properties such as latency, throughput, and accuracy of the OODA loop, as well as physical properties such as the drone's size, weight, thrust, lift, drag and moment of inertia. Under benign conditions, a non-agile drone may do as well as an agile one. Only under adversarial conditions does the cost of agility become valuable. This cost may include increased size, weight, and bandwidth/latency demand arising from the need to be faster and more accurate in sensing and actuation. The only way to quantify this complex property is to stress a drone on a precisely-defined task in a reproducible environment, and to use task-level metrics as surrogates for agility. This leads directly to the creation of benchmarks for evaluating autonomous drone agility.

We define two agility benchmarks in this paper. The first benchmark (§V) embodies obstacle avoidance in tight spaces. The adversarial aspect of this benchmark lies in the close proximity of obstacles, the need to sense them in real time (e.g., to account for wind effects), and occlusion that prevents full foreknowledge of the optimal flight path. The second benchmark (§VIII) embodies tracking of an object that moves in an unpredictable manner, with many abrupt changes. The adversarial aspect of this benchmark lies in the existence of an active mobile agent that randomly changes its trajectory. Both benchmarks are parameterized, thereby enabling many levels of difficulty within a common benchmark framework.

A valuable extension of our work would be to add the ability to control external disturbances such as wind velocity using the method proposed by Wu et al [24]. This would be straightforward for the physically compact avoidance benchmark (§V), but more challenging for the tracking benchmark (§VIII) because it covers a larger open area.

V. Obstacle Avoidance: Benchmark

A. Importance of Task

Obstacle avoidance is vital for drone flights at low altitude (up to a few hundred feet) in urban or forested settings. Otherwise, being restricted to only high altitude flight impairs visual detections and hides many details. The most dangerous obstacles are typically trees, lightposts, and telephone poles, which can easily reach altitudes usually used by drones. Being relatively thin, they are difficult to detect from afar.

Efficient avoidance of such obstacles is a challenge. Since drone flight is limited by battery life (typically on the order of 30-50 minutes), bypassing obstacles without wasting too much flight time is important. If flight is too slow or avoidance maneuvers are too convoluted, mission performance will be impaired. At the same time, reckless flight could be catastrophic. Striking the right balance between safety and speed for the given flight conditions is essential. Since effective but rapid avoidance of obstacles is a valuable capability in a drone, this task is a good candidate for an agility benchmark.

B. Benchmark Requirements

A good benchmark for this task should capture the essential difficulty of obstacle avoidance. It should be parameterized, so that it is easy to vary the difficulty of the benchmark. The benchmark should only use standardized, off-the-shelf components that can be easily purchased or fabricated. There should be no ambiguity in the experimental setup or interpretation of results, thereby simplifying independent attempts to reproduce published experimental results. §V-C presents our candidate benchmark that meets these requirements.

C. Benchmark Description

To emulate tall and skinny obstacles such as lightposts, we use 1.8 m long drone racing flags (Figure 12). The flags are arranged in a precisely-defined slalom pattern (Figure 13), that forces a drone to sequentially evade several obstacles. The difficulty of the course is controlled by a spacing parameter, w, that determines the separation of the flags along both the azimuth and range axes. A smaller value of w defines a more difficult course because of higher density of obstacles in both directions. All our experiments were conducted with 4 tiers of obstacles. Adding more tiers to make the obstacle course longer, while preserving obstacle spacing, would add further difficulty to the course. The drone's goal is to navigate the course as fast as possible, without touching the flags or striking a flagpole. The metric of interest is the transit time through the course.

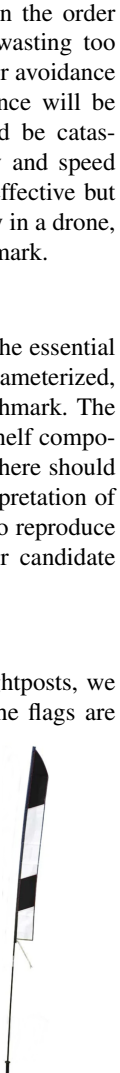


Fig. 12: Flag

To execute this benchmark, the obstacle course is set up as in Figure 13. The drone is placed on a pad that is centered 4 m in front of the leading line of flags. A human spotter stands a safe distance behind the drone, and a human timekeeper is positioned along the finish line. The remote pilot-in-command (RPIC) continuously monitors the video stream from the drone, and stands ready to wrest back manual control if the drone's autonomous flight software appears to be getting it into trouble. Such a manually aborted flight is scored as "Did Not Finish (DNF)." A flight in which the drone touches a flag or pole is also scored as DNF.

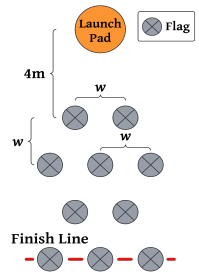


Fig. 13: Course Layout

The drone takes off and then hovers at an altitude of 1 m. It is then directed to autonomously move to a destination beyond the obstacle course. When forward motion begins, the spotter visually signals the timekeeper to start a stopwatch. The spotter then follows the drone through the course, warning the RPIC of imminent collision, if any. Such a warning aborts the experiment without damage to the drone. On a successful flight, timing is stopped as soon as the trailing end of the drone crosses the finish line. We deem an obstacle spacing, w, as viable if the drone successfully flies through the course on at least 80% of its attempts. The average time of these successful flights at the smallest viable w is the figure of merit. For small values of w, a more agile drone can fly more aggressively and therefore requires less time to complete the benchmark. However, at higher values of w, agility may not be important because the obstacle course is so easy.

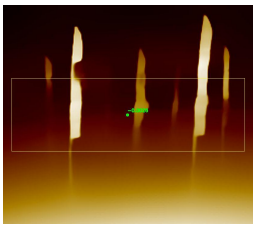
D. Benchmark Scoring

The average time, t_{avg}, for multiple experimental runs at the lowest viable w is a raw measure of agility. However, this needs to be normalized with respect to attributes other than agility. For example, a drone whose top speed is low relative to other drones may be penalized unfairly when measuring agility. A low value of t_{avg} in that case is not due to a poor OODA loop, but simply reflects the "brute force" attribute of top speed. The normalization is performed by removing flags from the course and conducting a control experiment at top speed. We denote the average time for multiple runs of the control experiment as t_{min}. The score, S_w, is then given by:

$$S_w = \frac{t_{\text{min}}}{t_{\text{avg}}} \tag{1}$$

Scores for this benchmark thus lie in the interval $0 < S_w \leq 1$ where $S_w = 1$ is the best possible score. A score of 0 is awarded when a drone cannot achieve a successful completion rate of at least 80% of the runs for the given value of w.

(a) Raw Input



(b) Output of Our Algorithm

Fig. 14: Monocular Obstacle Avoidance Using MiDaS

Model	Latency (ms)	Throughput (fps)
MiDaS Small	61	10
DPT Hybrid	105	8
DPT Large	132	7

These numbers were obtained on the cloudlet described in §III-D

Fig. 15: Inference Speeds of MiDaS DNN Models

VI. Obstacle Avoidance: Depth Sensing

Some drones utilize stereo cameras or LIDAR to detect and avoid obstacles. These give accurate depth data (i.e., distance to objects), allowing the drone to map out its environment and to calculate optimal collision-free trajectories. Since our drone is only equipped with a monocular camera, it cannot infer depth via simple geometric methods. We therefore use a DNN-based algorithm called MiDaS [39] to provide relative depth estimates. Using MiDaS on each frame received by the cloudlet, we construct the inverse relative depth map. Based on the rate of change of relative depth across frames, we identify obstacles in the flight path and actuate away from them. Figure 14(a) shows an input frame from one of our flights, as the drone approaches the obstacle course. Figure 14(b) shows the depth-encoded output of our algorithm on this frame. The drone actuates towards the green dot using a tuned PID-loop [40] to avoid the closest flags. Once past these flags, it re-computes a new safe objective towards which it can fly to avoid the second tier of flags. It repeats these steps until it is past all obstacles. This simple approach to obstacle avoidance serves as a good experimental baseline. Future work could benchmark more sophisticated avoidance algorithms such as those proposed by Wu et al [24].

For this task, the OODA loop determines the speed and accuracy with which the drone can acquire fresh frames, execute the MiDaS algorithm on them, calculate new headings for safety, and perform actuations towards those headings. In the context of §III-D, MiDaS represents Stage-2 of cloudlet processing. Stage-3 is the processing of MiDaS output to determine a new heading, and generating the actuation command.

We explore three DNN variants of MiDaS that vary in accuracy and speed: *MiDaS Small, DPT Hybrid*, and *DPT Large*. MiDaS Small prioritizes throughput and inference latency at the cost of lower accuracy. DPT Hybrid strikes a compromise between speed and accuracy. DPT Large prioritizes accuracy above all else. Figure 15 shows the inference latency and throughput of these three models on our cloudlet. It is not *a priori* obvious which model is best for obstacle avoidance. We therefore explore use of all three in our experiments.

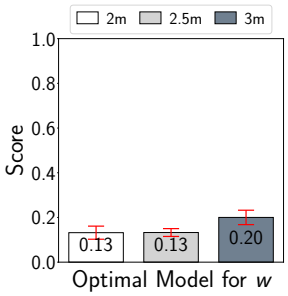


Fig. 16: Baseline Scores

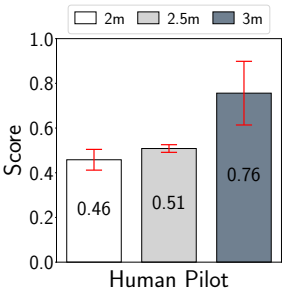


Fig. 17: Human Pilot

VII. Obstacle Avoidance: Results

A. Baseline

The most basic questions in our evaluation are as follows:

- *What is the smallest value of w for which our drone can successfully complete the benchmark?*
- *At that w, how fast is benchmark completion?*
- *As w is increased, how much faster is the drone able to complete the benchmark?*

Our initial experiments showed that 2 m is the smallest value of w that meets our criterion for successful benchmark completion (i.e., at least 80% of the flights are successful). Using the scoring criterion described in §V-D, Figure 16 shows how well our drone did for w set to 2 m, 2.5 m, and 3.0 m. The results shown are the mean of 5 runs of each experiment. For each value of w, the drone results were obtained with the choice of MiDaS model that gave the best results. These choices were DPT Large for w =2 m, and MiDaS Small for $w = 2.5$ m and $w = 3$ m. The scores of 0.13 to 0.2 show that the drone suffers almost an order of magnitude slowdown when avoiding obstacles, relative to its unimpeded traversal of the course. This is the price of having to execute the OODA loop shown in Figure 11, with the additional Stage-2 and Stage-3 processing for depth estimation described in §VI. As w is increased from 2 m to 3 m, Figure 16 shows the score improving from 0.13 to 0.2. This confirms our expectation that less challenging courses are faster to traverse.

Since humans are the standard against which AI systems are measured, we ask how well a human pilot does under identical conditions. To explore this, an experienced RPIC with several dozen hours of flight time on the Parrot ANAFI platform manually executed the benchmark for the same values of w.[1] The OODA loop is now cyber-human: the RPIC uses the drone's live video stream to manually fly it. Of course, a human also has foreknowledge of the obstacle course and can subconsciously leverage that knowledge in planning the

[1]No IRB approval for human subjects was required for this experiment since the pilot is an author of this paper.

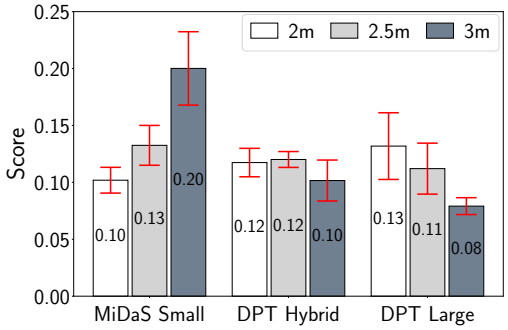


Fig. 18: Impact of MiDaS Model on Avoidance Benchmark

drone's flight. In contrast, the autonomous drone is purely reactive — what it sees right now is all that it knows.

Figure 17 presents our results. Relative to unimpeded traversal of the course, the scores of 0.76 to 0.46 show that even the RPIC suffers a slowdown. However, the slowdown is much smaller than that suffered by the autonomous drone in Figure 16. Since the drone hardware and wireless network are identical in the two sets of experiments, the difference is clearly due to the superior OODA loop of the human. There is clearly ample headroom for improvement of our drone's OODA loop. Exact attribution of the observed difference to different parts of the OODA loop will require more study.

B. Impact of Model Accuracy

The availability of the different MiDaS models shown in Figure 15 leads to the question:

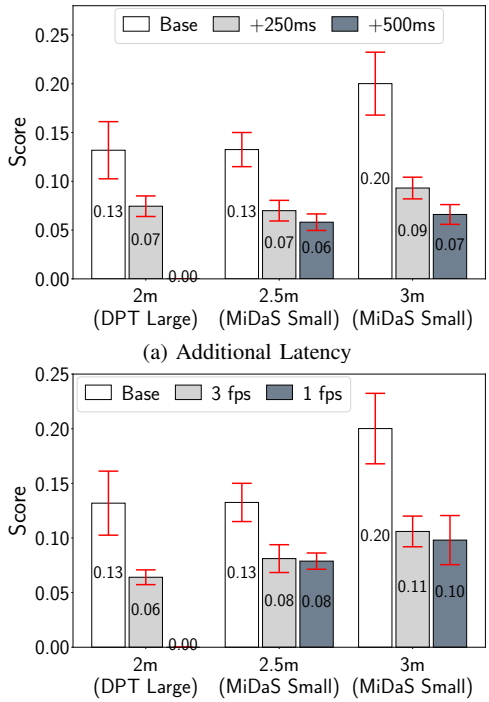
- *Is accuracy or speed more important?*

Our experiments indicate that there is no simple answer to this question. The results in Figure 16 were obtained using the best MiDaS model for each value of w. MiDaS Small performs the best on the 3 m course but worst on the 2 m course. DPT Large is the inverse, performing best on the 2 m course and worst on the 3 m course. DPT Hybrid stays consistently in the middle for all three courses. The fact that different models had to be used in each case to obtain the best results indicates that there is no single "best" model.

We conducted a set of experiments to better understand this tradeoff space. The results in Figure 18 show the score achieved on the benchmark for each model and value of w. Since MiDaS Small focuses on throughput and low latency over accuracy, a drone that uses it is able to sustain a higher maximum speed than one using DPT Large. At w =3 m, the course is sufficiently easy that the increased risk of collision is small. The higher accuracy of a better model is not useful. However, at w =2 m, the increased likelihood of collisions makes higher model accuracy worthwhile. Now, DPT Large attains the highest score. For a tight course, it is difficult to travel at high speed without collisions. Hence, a model that can navigate the gaps better gets a higher score.

C. Impact of Latency & Throughput

The results presented so far reflect best-case conditions. In practice, the wireless network or the cloudlet may suffer



(a) Additional Latency

(b) Reduced Throughput

Fig. 19: Impact of Latency and Throughput on Avoidance

degradation due to multi-tenancy. This leads to the question:

- *What is the impact of latency or throughput degradation of the* OODA *loop on benchmark score?*

The baseline scores (§VII-A) reflect what is achievable with an OODA loop whose latency is the sum of three components:

- a lower bound of 527 ms (Figure 11).
- the latency of the relevant MiDaS model (Figure 15).
- a small additional overhead (<1 ms) for Stage-3 processing in the Decide part of the OODA loop.

This total end-to-end latency is on the order of 600–650 ms. For OODA loop iterations that involve drone actuation, the bottleneck throughput is the smaller of 6 fps for Act_{fg} (§III-F) and the throughput of the MiDaS model (Figure 15). This is effectively 6 fps, regardless of model. If no drone actuation is involved, the bottleneck throughput becomes that of the MiDaS model. Since even MiDaS Small has lower throughput than Observe_{ab}, Observe_{c}, or best-case Orient+Decide_{d}, throughput is always in the 6–10 fps range.

Figure 19(a) shows how benchmark score drops as latency is artificially added to the OODA loop. Even 250 ms of additional latency (i.e., a roughly 35–40% increase from baseline) causes benchmark score to drop to nearly half its baseline value, for all values of w and regardless of MiDaS model. If 500 ms of latency is added, the score drops further. For the most challenging course (w = 2 m), the score drops to zero because not even 80% of the flights are successful. These results are consistent with a long-standing design principle of deeply-immersive closed-loop interactive systems: *increased latency is deadly, even if throughput remains good.*

Figure 19(b) shows how benchmark score drops as the throughput of the OODA loop is artificially reduced from its baseline value. Reducing throughput to 3 fps causes benchmark score to drop to nearly 40–50% of its baseline value. A further drop is observed when throughput is reduced to 1 fps. For $w = 2$ m, the benchmark score drops to zero. These results confirm that latency is not the sole determinant of task performance in an OODA loop — throughput also matters.

VIII. Visual Object Tracking: Benchmark

A. Importance of Task

In visual object tracking, a drone follows a moving target and tries to keep it centered in its FOV. Many surveillance-related instances of this task, where the target may be adversarial and actively try to escape tracking, occur in law enforcement and military settings. The task is also relevant to wildlife conservation research, where an animal of an endangered species is identified and followed in the wild. It is also used in filmmaking to capture evolving scenes. In such use cases, using an autonomous drone for tracking could reduce attention demand on mission personnel.

B. Benchmark Requirements

The size and visual appearance of a target plays an important role in tracking success. An object that is just a few pixels in size from the altitude of the drone will be inherently difficult to detect [41]. Poor contrast with the background, as happens when camouflaged, also contributes to poor detection. Objects that are hard to detect are also hard to track, since actuating to re-center the target in each frame is key to success. The object being tracked and the background on which it moves both need to be specified. Only when these factors and drone optics are held constant will OODA loop performance come to the fore in determining tracking performance.

The other benchmark requirements for tracking are similar to those described for obstacle avoidance (§V-B). Parameterization that controls the difficulty of the task is valuable. Use of standardized, off-the-shelf components and careful attention to reproducibility of results are important.

C. Benchmark Description

Fig. 20: Target

The object followed in our tracking benchmark is a DJI Robomaster S1 robot [42]. This robot is roughly the size of a small dog (Figure 20), and can be programmed to follow a predefined route. As shown in Figure 21, tracking is done on a level, green background such as a football field. For this combination of target and background, DNN-based object detection from an altitude of 10 m is successful at a confidence level of 0.9 or higher on frames from our drone's video camera.

Our benchmark is a random walk with turns in a randomly-chosen cardinal direction at each step. Figure 22(a) shows one example with 5 steps, and Figure 22(b) shows other examples with more steps. The benchmark has three parameters: the number of steps; the mean length of each step; and the target speed of 1.5 m/s, 2.5 m/s, or 3.5 m/s. The benchmark could be made more complex by making the turns to be at any angle rather than just cardinal directions, and by making step size and target speed non-uniform. All our experiments were conducted across the full range of speeds, using a stepcount of 35 steps and stepsize set to 5 m.

To execute the benchmark, the target is placed in a large open outdoor area. The drone is manually piloted to the desired altitude, and its FOV is adjusted to center the target. Once the drone has locked onto its target, the target is instructed to start its pre-programmed random walk and a timekeeper starts a stopwatch. The experiment continues until one minute has elapsed or the drone loses the target from its FOV. The termination time and the black box footage of the flight are logged for post-flight scoring (§VIII-D).

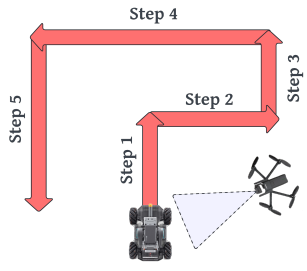
D. Benchmark Scoring

In postprocessing after a flight, we score the recorded video footage using an automated process. Figures 23 to 25 show the scoring calculation, using Figure 26 as an example. On each frame, a DNN is first used to create a bounding box around the target. With the center of the frame as the origin, the relative distance of the target from the origin is obtained. Using the notation shown in Figure 23, the pixel offset vector, $\vec{O}$, gives the L2 distance of the target's centroid from the origin. This is scaled to the vector dimension, $\vec{D}$, of the bounding box to give the centering ratio c_i (formula 2). We then calculate the score of the frame, s_i, by using an inverse exponential, as shown in formula 3. The rationale for using an exponential is to super-linearly penalize distance from origin. We use a compounding 10% penalty in reporting our results, leading to the value of 1.1 in formula 3. Using s_n to denote a penalty of n%, Figure 25 shows the scores for penalties of 10%, 20% and 30% for the example frame in Figure 26. A score of zero is awarded when the frame does not contain the target at all, or if the target is too small to be detected in post-processing by a specified model. In our case, this model is YOLOv5x trained on aerial images of the target.

From the per-frame scores, the entire flight is scored by simple averaging (formula 4). The overall score, s_{avg}, lies between 0 and 1, with higher being better. For example, an average score of 0.70 based on s_{10} is achieved when the drone is able to keep the target within about three normalized lengths of the center of the FOV for the entire duration of the flight.

The bounding box shows target detection at a confidence level of 0.9.

Fig. 21: Background for Object Tracking Benchmark



(a) Detail for 5-Step Example

(b) Other Examples

Fig. 22: Parameterized Random Walk

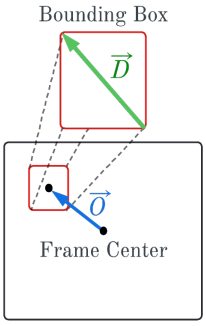


Fig. 23: $\vec{O}$ & $\vec{D}$

$$c_i = \left\|\vec{O}_i\right\| / \left\|\vec{D}_i\right\| \quad (2)$$

$$s_i = 1.1^{-c_i} \quad (3)$$

$$s_{\text{avg}} = \frac{\Sigma_i^n s_i}{n} \quad (4)$$

Fig. 24: Calculating Score

$$\left\|\vec{O}\right\| = 0.11,\ \left\|\vec{D}\right\| = 0.03,\ c = 3.67$$

$$s_{10} = 1.1^{-c} = 0.70$$
$$s_{20} = 1.2^{-c} = 0.51$$
$$s_{30} = 1.3^{-c} = 0.38$$

Fig. 25: Scoring Figure 26

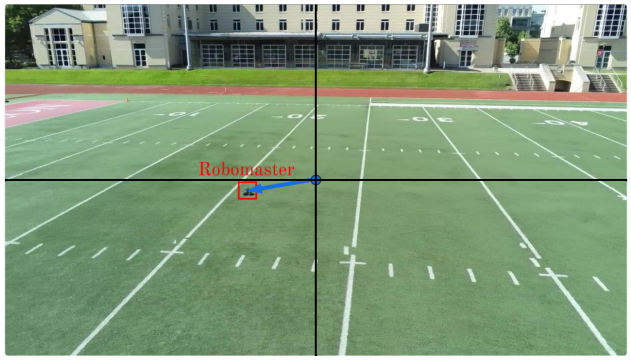


Fig. 26: Example Frame for Scoring

Model	**Latency** (ms)	**Throughput** (fps)	**mAP**
YOLOv5s	28	25	56.8
YOLOv5m	37	20	64.1
YOLOv5l	42	20	67.3

The inference and throughput were obtained on our cloudlet (§III-D). The mean average precision (mAP) is from the YOLO documentation [43].

Fig. 27: YOLOv5 Performance in Our Pipeline

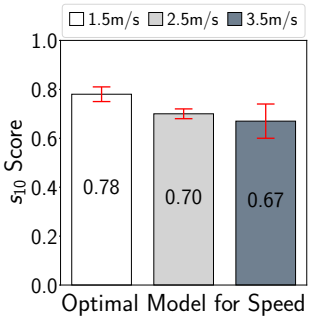


Fig. 28: Baseline Scores

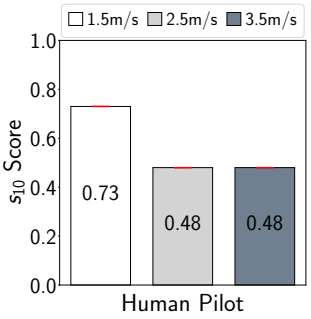


Fig. 29: Human Pilot

IX. Visual Object Tracking: Algorithm

Edge offload to a powerful cloudlet enables tracking via DNN inferencing on every frame received. This "brute force" approach eliminates the need for predictive heuristics, such as those based on optical flow algorithms. Heuristics are often needed by on-board tracking implementations because the computational demand would otherwise be too high. The brute force approach makes tracking robust with respect to transient occlusions. Flow-based approaches, in contrast, are typically unable to reacquire the target after occlusion ends.

In our system, the cloudlet inferences each frame through an object detection DNN. The highest confidence bounding box is then chosen as the target, and its offset from the center of the frame is calculated in field-of-view degrees. The drone then actuates according to a PID-loop [40] based on the offset error. This simple approach provides a good baseline at low complexity. More sophisticated algorithms such as dynamic Kalman models [44] can be benchmarked in the future.

Figure 27 shows the latency, throughput and accuracy of the three DNN models that are used for tracking in our system. Even using the slowest of these as Stage-2 of Orient+Decide$_d$ only adds 42 milliseconds of latency to the base value of 527 ms (Figure 11). Its throughput of 20 fps is well above that of the bottleneck (Act$_{fg}$). However, there may be situations where load on a multi-tenant cloudlet may need to be reduced, and the smaller models may be valuable for that purpose.

X. Visual Object Tracking: Results

A. Baseline

The basic question we ask about tracking is as follows:

- *How well does our platform follow a target that makes random, rapid changes in direction?*

As Figure 28 shows, our platform is able to track the target on our benchmark even at the fastest speed (3.5 m/s) without ever completely losing it. However, as the scores show, the target is off-center in some frames at all speeds. As target speed decreases, the score achieved shows a modest improvement. The results shown here are based on the best model for each speed. This dependence is explored further in §X-B.

As in the case of obstacle avoidance, we ask how well an experienced human pilot performs under identical conditions. The pilot is held constant from §VII-A. Figure 29 shows how

well the human pilot scored on the benchmark. Comparing Figures 28 and 29, we see that the autonomous drone and the human are comparable at 1.5 m/s, but at higher speeds the autonomous drone outperforms the human. This is in contrast to obstacle avoidance (Figures 16 and 17), where the human consistently outperformed the autonomous drone. We conjecture that at least part of this difference is attributable to the fact that the obstacle course is static, and hence subconscious pre-planning by the human helps in navigating it. In contrast, the human is no better than the drone in anticipating random turns made by the target. At higher speeds, raw reaction speed (i.e., the OODA loop) is all that matters, and the autonomous drone proves to be better in this regard.

B. Impact of Model Accuracy

Since multiple DNN models are available to use in tracking (Figure 27), we ask the following question:

- *Does the use of a better model improve tracking?*

Figure 30 presents our results. For any given speed, there is little difference across models. The increased cloudlet load of a more accurate model does not pay off. However, it should be noted that this observation may only be true for this specific tracking benchmark. As described in §VIII-C, the benchmark is defined as being conducted in an open area free of clutter. If we were to create a different tracking benchmark that embodies extensive clutter (such as that of a busy street filled with moving cars, bicycles, and pedestrians, along with static objects such as parked cars and trees), the results may be quite different. In that case, the improved accuracy of the larger models may prove decisive. We plan to explore the creation of such a benchmark in our future work.

C. Impact of Latency & Throughput

As in the case of obstacle avoidance (§VII-C), we ask:

- *What is the impact of latency or throughput degradation of the* OODA *loop on benchmark score?*

Figure 31(a) shows what happens when additional latency of 250 ms and 500 ms are added to the OODA loop. For all target speeds and models, there is a noticeable drop in benchmark score. The drop is worse at higher speeds. This is directly attributable to the inability of the more sluggish OODA loop to keep the target centered in the FOV. At higher speeds, the drone often loses sight of the target early in the tracking. This results in a zero score for the remaining frames of that experiment, and hence an overall low average score.

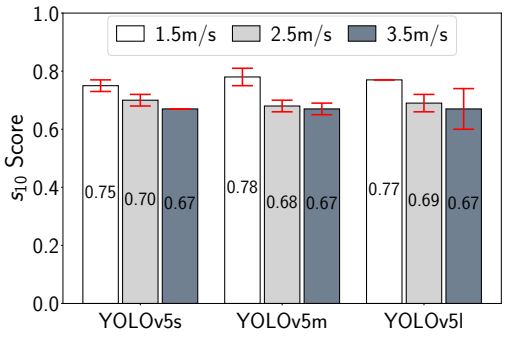
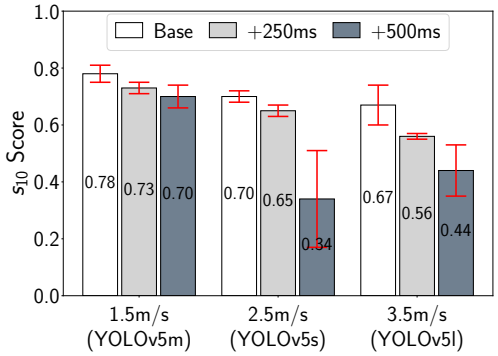
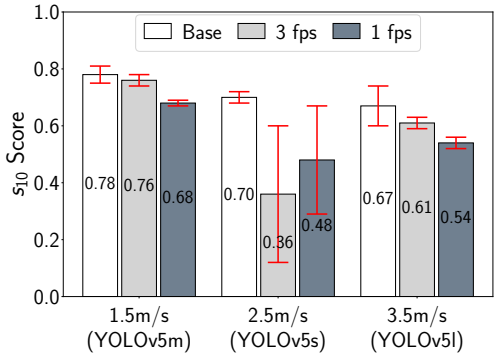


Fig. 30: Impact of YOLO Model on Tracking Benchmark



(a) Additional Latency



(b) Reduced Throughput

Fig. 31: Impact of Latency and Throughput on Tracking

Figure 31(b) shows the same trend when OODA loop throughput is artificially throttled to 3 fps or 1 fps. At all target speeds and for all models, there is a noticeable drop in benchmark score. This drop is greater at higher speeds.

The results in Figures 31(a) and (b) confirm that both end-to-end latency and bottleneck throughput are important independent factors in determining tracking ability. Optimizing one at the cost of the other, as occurs when using strategies such as batching of operations, is unlikely to be beneficial.

XI. Value of On-board Drone Intelligence

Edge computing allows use of compute resources that are far larger and heavier than could be carried by an ultralight drone. In the context of AI, this translates to *generality* and *versatility*. Purely through software development on the cloudlet, it is easy to re-purpose the drone for new tasks that were not anticipated earlier. For example, a cloudlet-based zero-shot object detection transformer model [45] could be made available for use in the OODA loop.

The drone marketplace, however, is moving in the opposite direction. Drone vendors are constantly identifying specific new functionality to add to drones. There is a well-understood tradeoff between generality, energy-efficiency, development cost, and weight/size that applies in this context. As Figure 32 from Chung et al [46] shows, an ASIC is by far the most

energy efficient alternative for a given functionality. It is also likely to have the lowest weight. However, it takes much longer to develop, is much more expensive to create than pure software, and only provides fixed functionality.

To quantify the value of on-board intelligence, we re-ran our obstacle avoidance benchmark using a DJI Mini 4 Pro drone. This consumer photography drone weighs 249 g, and is equipped with 6 stereo cameras and an onboard obstacle avoidance system. It has two modes, Normal and Nifty. Normal prioritizes safe flight while Nifty attempts to pass obstacles as quickly as possible. The obstacle avoidance feature of this drone is intended as a form of "pilot assist." The RPIC flies the drone without worrying about obstacles. The drone's builtin capability performs all the necessary sensing and actuation needed to avoid collisions.

Figure 33 presents the scores obtained by the DJI Mini 4 Pro on our obstacle avoidance benchmark. The baseline results for our platform from Figure 16 are also shown for comparison. For all values of w, the DJI Mini 4 Pro is at a clear advantage. This the direct result of additional sensors and a much faster OODA loop that avoids edge offload.

These results suggest that even when using edge offload, there is very clear value in taking advantage of on-board capabilities when they are available. The "pilot assist" approach to obstacle avoidance implemented by the DJI Mini 4 Pro could equally well be used by cloudlet-based software to control the drone's flight path. Only the macro components of that flight path would incur the overhead of the OODA loop from drone to cloudlet. The micro components of the flight path that maneuver the drone around obstacles would only be subject to its much tighter on-board OODA loop.

Complementing on-board fixed functionality with edge offload provides extensibility and versatility. The DJI Mini 4 Pro, for example, is unable to do general-purpose object detection even though it can detect people and vehicles. It cannot detect our target, and is hence unable to execute our tracking benchmark. Edge offload could remedy that limitation, thereby increasing the versatility of the drone.

Architecture	Efficiency (GFLOP / J)
CPU (Core i7)	1.14
FPGA (Xilinx LX760)	3.62
GPU (NVIDIA GTX285)	6.78
GPU (AMD R5870)	9.87
ASIC	50.73

Source: Table 4 in Chung et al [46]

Fig. 32: Matrix Multiplication Kernel Implementations

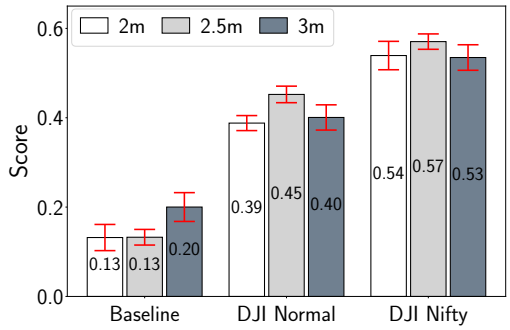


Fig. 33: On-Board Obstacle Avoidance

XII. Conclusion

We began this paper by asking whether the end-to-end performance of edge offload using COTS ultralight drones, commercial 4G LTE wireless networks, and COTS cloudlet hardware, is sufficient for real-world active vision tasks. We quantitatively analyzed such a COTS pipeline through the lens of an OODA loop, developing parameterized and reproducible benchmarks for avoidance and tracking tasks. Our results confirm that a cloudlet-based system today is already very close to being useful for real-world active vision tasks. No new ecosystem of custom-built drones and low-latency wireless networks is needed for commercialization. The most important short-term targets for optimization are on-drone sensing and encoding of video, as well as drone actuation. The benchmarks we have presented on obstacle avoidance and object tracking, together with our flight platform's results on them, provide a baseline against which future improvements in drone autonomy can be measured.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their thoughtful and constructive feedback that improved the presentation of this paper. This material is based upon work supported by the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W519TC-23-C-0003 and by the National Science Foundation under grant number CNS-2106862. The content of the information does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred. This work was done in the CMU Living Edge Lab, which is supported by Intel, Arm, Vodafone, Deutsche Telekom, CableLabs, Crown Castle, InterDigital, Seagate, Microsoft, the VMware University Research Fund, IAI, and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

References

[1] H.-M. Huang, "Autonomy Levels for Unmanned Systems (ALFUS) Framework, Volume I: Terminology, Version 2.0," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 1011-I-2.0, October 2008.

[2] M. Bala, T. Eiszler, X. Chen, J. Harkes, J. Blakley, P. Pillai, and M. Satyanarayanan, "Democratizing Drone Autonomy via Edge Computing," in *2023 IEEE/ACM Symposium on Edge Computing (SEC)*, Wilmington, DE, 2023.

[3] R. E. Weibel and J. R. John Hansman, "Safety Considerations for Operation of Different Classes of UAVs in the NAS," in *Proceedings of the 4th Aviation Technology, Integration and Operations (ATIO) Forum*, Chicago, IL, September 2004.

[4] "Drone Services Pricing: An Introductory Guide," (https://www.dronepilotgroundschool.com/drone-services-pricing/), August 2024, Last accessed on September 26, 2024.

[5] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International Journal of Computer Vision*, vol. 1, pp. 333–356, 1988.

[6] D. Ognibene and Y. Demiris, "Towards active event recognition," in *The 23rd International Joint Conference on Artificial Intelligence, IJCAI-13*, August 2013.

[7] R. E. Enck, "The OODA Loop," *Home Health Care Management & Practice*, vol. 24, no. 3, pp. 123–124, 2012, dOI: 10.1177/1084822312439314.

[8] F.-Y. Wang and D. Liu, *Networked control systems*. Springer, 2008.

[9] T. Huntsberger, P. Pirjanian, A. Trebi-Ollennu, H. Das Nayar, H. Aghazarian, A. Ganino, M. Garrett, S. Joshi, and P. Schenker, "Campout: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 33, no. 5, pp. 550–559, 2003.

[10] Z. Qu, *Cooperative control of dynamical systems: applications to autonomous vehicles*. Springer Science & Business Media, 2009.

[11] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 73–82, 2017.

[12] L. Bertizzolo, S. DOro, L. Ferranti, L. Bonati, E. Demirors, Z. Guan, T. Melodia, and S. Pudlewski, "Swarmcontrol: An automated distributed control framework for self-optimizing drone networks," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1768–1777.

[13] G. Asaamoning, P. Mendes, D. Rosário, and E. Cerqueira, "Drone swarms as networked control systems by integration of networking and computing," *Sensors*, vol. 21, no. 8, p. 2642, 2021.

[14] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.

[15] J. R. Boyd, "Patterns of Conflict," (https://www.ausairpower.net/JRB/poc.pdf), 1986, Slides from a presentation by John R. Boyd, Last accessed October 18, 2024.

[16] L. M. Blaha, "Interactive OODA processes for operational joint human-machine intelligence," in *NATO IST-160 Specialists Meeting: Big Data and Military Decision Making*. NATO, 2018, pp. 3–1.

[17] J. Johnson, "Automating the OODA loop in the age of intelligent machines: reaffirming the role of humans in command-and-control decision-making in the digital age," *Defence Studies*, vol. 23, no. 1, pp. 43–67, 2023.

[18] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking," (https://arxiv.org/abs/1804.00518), 2018, last accessed on July 1, 2024.

[19] S. Li and D.-Y. Yeung, "Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[20] I. Kalra, M. Singh, S. Nagpal, R. Singh, M. Vatsa, and P. B. Sujit, "DroneSURF: Benchmark Dataset for Drone-based Face Recognition," in *Proceedings of the 14th IEEE International Conference on Automatic Face & Gesture Recognition*, Lille, France, 2019.

[21] X. Zhao, S. Hu, Y. Wang, R. Li, K. Liu, J. Li, J. Zhang, Y. Hu, R. Liu, H. Ling, and Y. Li, "BioDrone: A Bionic Drone-Based Single Object Tracking Benchmark for Robust Vision," *International Journal of Computer Vision*, vol. 132, pp. 1659–1684, 2024.

[22] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. Reddi, "MAVBench: Micro Aerial Vehicle Benchmarking," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 894–907.

[23] S.-A. Yu, C. Yu, F. Gao, Y. Wu, and Y. Wang, "Flightbench: A comprehensive benchmark of spatial planning methods for quadrotors," *arXiv preprint arXiv:2406.05687*, 2024.

[24] Y. Wu, Z. Ding, C. Xu, and F. Gao, "External forces resilient safe motion planning for quadrotor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8506–8513, 2021.

[25] A. Koubaa, A. Ammar, M. Alahdab, A. Kanhouch, and A. T. Azar, "DeepBrain: Experimental Evaluation of Cloud-Based Computation Of-

[27] S. Hayat, R. Jung, H. Hellwagner, C. Bettstetter, D. Emini, and D. Schnieders, "Edge computing in 5g for drone navigation: What to offload?" *IEEE Robotics and Automation Letters*, vol. 6, no. 2, 2021.

floading and Edge Computing in the Internet-of-Drones for Deep Learning Applications," *Sensors*, vol. 20, no. 18, 2020, (https://www.mdpi.com/1424-8220/20/18/5240).

[26] J. Chen, S. Chen, S. Luo, Q. Wang, B. Cao, and X. Li, "An intelligent task offloading algorithm (iTOA) for UAV edge computing network," *Digital Communications and Networks*, vol. 6, pp. 433–443, 2020.

[28] R. Wang, Y. Cao, A. Noor, T. A. Alamoudi, and R. Nour, "Agent-enabled task offloading in UAV-aided mobile edge computing," *Computer Communications*, vol. 149, pp. 324–331, 2020.

[29] G. Wu, Y. Miao, Y. Zhang, and A. Barnawi, "Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading," *Computer Communications*, vol. 150, pp. 556–562, 2020.

[30] H. Chen, L. Zeng, X. Zhang, and X. Chen, "AdaDrone: Quality of Navigation Based Neural Adaptive Scheduling for Edge-Assisted Drones," in *Proceedings of 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022.

[31] Defense Innovation Unit, "Blue UAS," (https://www.diu.mil/blue-uas), Last accessed on May 25, 2024.

[32] "Onion Omega2 LTE," (https://docs.onion.io/omega2-docs/omega2-lte.html), Last accessed June 8, 2024.

[33] C. Tsaousis and P. Whineray, "FireQOS Reference," (https://firehol.org/fireqos-manual.html), Last accessed on May 26, 2024.

[34] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, "Real-Time Streaming Protocol Version 2.0," https://www.rfc-editor.org/rfc/rfc7826, Internet Engineering Task Force (IETF), Tech. Rep. RFC 7826, December 2016.

[35] Adobe, "What is video bitrate?" (https://www.adobe.com/creativecloud/video/discover/bit-rate.html), Last accessed on June 23, 2024.

[36] A. Chanana, M. Bala, T. Eiszler, J. Blakley, and M. Satyanarayanan, "Benchmarking Drone Video Stream Latency in SteelEagle," (http://ra.adm.cs.cmu.edu/anon/2024/CMU-CS-24-128.pdf), Computer Science Department, Carnegie Mellon University, Tech. Rep. CMU-CS-24-128, May 2024.

[37] "PDrAW – Parrot Drones Awesome Video Viewer," (https://github.com/Parrot-Developers/pdraw), Last accessed June 19, 2024.

[38] Parrot, "Olympe Documentation," (https://developer.parrot.com/docs/olympe/index.html), Last accessed June 18, 2024.

[39] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, 2022.

[40] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[41] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," in *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017.

[42] DJI, "Robomaster S1," https://www.dji.com/robomaster-s1, Last Accessed November 14, 2022.

[43] "Yolov5," (https://github.com/ultralytics/yolov5), last accessed on July 1, 2024.

[44] Y. Wu, Y. Sui, and G. Wang, "Vision-based real-time aerial object localization and tracking for uav sensing system," *IEEE Access*, vol. 5, pp. 23 969–23 978, 2017.

[45] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," (https://arxiv.org/abs/2303.05499), 2023, last accessed on July 2, 2024.

[46] E. Chung, P. A. Milder, J. C. Hoe, and K. Mai, "Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?" in *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010