ChatGPT

Introduction

Micro aerial robots, such as small quadrotors, are increasingly tasked with vision-based autonomy like surveillance, search-and-rescue, and navigation. These tasks demand **real-time object detection and multi-object tracking (MOT)** using onboard cameras, but achieving this on lightweight, battery-powered hardware (e.g. Raspberry Pi or NVIDIA Jetson Nano) is challenging. In the past five years (2020–2025), significant research has focused on **efficient computer vision models and systems optimized for deployment on embedded UAV platforms**. Key challenges include the limited computing resources and power on micro drones, the need to maintain high detection accuracy for small, distant objects, and integration of perception outputs with flight control in real time [1] [2]. This report reviews recent advances in onboard object detection and MOT for micro aerial robots, emphasizing **lightweight models, hardware acceleration techniques, onboard perception-control integration**, performance benchmarks, and open-source resources.

Lightweight Object Detection Models for Embedded UAVs

Real-time **object detection** on micro UAVs has gravitated toward one-stage CNN models that balance accuracy and speed. The YOLO family (You Only Look Once) has been particularly popular, with "tiny" or nano versions designed for efficiency. For example, *YOLOv5-Nano* and *YOLOv7-Tiny* models (few million parameters) have been demonstrated on embedded GPUs [3]. In tests, a YOLOv7-tiny detector on a Raspberry Pi 4 could only achieve ~0.9 FPS, whereas on a Jetson Nano it reached ~5–7 FPS, and up to ~30 FPS on the more powerful Jetson Xavier NX [3]. Newer YOLO versions continue this trend: a 2024 study reported that YOLOv8-Nano can run at ~60 FPS on a Jetson Orin Nano (using TensorRT optimization), whereas larger YOLOv8 models (e.g. Medium) run slower (~20 FPS) but with higher accuracy [4]. These results highlight the trade-off between model size and real-time performance on edge devices [1].

Researchers have also explored **alternative lightweight detectors**. One 2023 study modified the *CenterNet* one-stage detector for small object detection on drones [5] [6]. The authors tested CNN backbones from ResNet18 up to Res2Net101 and hourglass networks, comparing them against YOLOv1–v7, SSD-MobileNet and others on aerial datasets (VisDrone, UAV datasets). Their optimized CenterNet (with Res2Net101 backbone) achieved the highest mAP, especially on small objects, outperforming YOLOv7 in accuracy on drone-specific data [7] [8]. However, on an embedded Jetson Nano, even the fastest configuration (ResNet18 backbone) only ran ~4.5 FPS [9] [10], underscoring how heavy models must be pared down for real-time use. The authors note an "acceptable FPS for a drone would be at least 25–30" for safe autonomous flight [2], a target not met by most unaccelerated models on Nano.

Model compression and optimization techniques are therefore crucial. A notable 2024 work by Zhu *et al.* introduced a *lightweight optimization of YOLOv3* for Jetson Nano [11]. They replaced YOLOv3's backbone with an Efficient-RepVGG network optimized for GPU throughput, designed a compact bidirectional feature pyramid, and an "optimized couple head" for detection. This drastically reduced the model size – cutting ~92% of parameters and compute – and yielded a **9.95× speedup** in inference, with minimal accuracy loss [12]. The optimized YOLOv3 ran 1.6–2× faster than other detectors of similar accuracy, or conversely achieved 3–6% higher mAP at similar speed [13]. Such architecture tailoring (e.g. using depthwise separable convolutions, smaller feature maps, etc.) has become a common strategy to **maintain accuracy while shrinking models** for edge deployment. Other research has benchmarked

EfficientDet-Lite models and SSD variants on micro UAV hardware, since those were originally designed for mobile/embedded use [14]. For instance, EfficientDet-Lite0 (the smallest EfficientDet) and SSD-MobileNet were found to be very fast on devices like Pi+TPU or Orin, albeit with lower mAP than newer YOLO models [15] [4]. **Transformer-based detectors** are generally too heavy for micro drones, but simplified versions like *RT-DETR* (Real-Time DETR) have been tested in simulation. A recent work used RT-DETR on a ground station for high-accuracy detection, while keeping a YOLOv5-Nano onboard the drone for coarse detection [16] [17]. In summary, the past five years show a clear trend of developing **smaller, CNN-based detectors (YOLO nanos, MobileNet-SSD, etc.) and custom-tailored networks** that can fit the strict latency and memory constraints of onboard hardware.

Real-Time Multi-Object Tracking on Drones

Building on object detection, **multi-object tracking (MOT)** algorithms link detections across frames to track trajectories of multiple targets (e.g. people, vehicles, or animals) in view. On resource-constrained UAVs, most approaches adopt a *tracking-by-detection* paradigm: a lightweight detector processes each frame (or every Nth frame), and a tracker handles association. Simpler trackers like SORT and its variants are favored for embedded use due to their low overhead [18] [19]. *SORT* (Simple Online Real-time Tracking) uses a Kalman filter and frame-to-frame data association via IoU matching, running at hundreds of FPS on a CPU [20] [21]. However, SORT can suffer frequent ID switches under occlusion. Improved versions like *DeepSORT* add a re-identification CNN to enhance robustness (re-identifying objects after brief disappearance) [22] [23]. DeepSORT's ReID model adds computational load, but can still run in real time on a Jetson-class GPU when the number of objects is modest. For example, in a ground-based test, DeepSORT with YOLOv8x was used to track vehicles and re-identify targets that the drone's onboard detector missed [17] [24]. Recent trackers like *ByteTrack* (which keeps low-confidence detections to improve tracking) and *OC-SORT* have also been tested on drone imagery [25]. In a 2025 wildlife monitoring study (WildLive), the authors benchmarked these trackers onboard and confirmed that modern MOT algorithms (OC-SORT, ByteTrack) are *computationally feasible on edge devices* while providing high accuracy [26] [25].

To push MOT performance for drones, researchers have proposed combining classical tracking (e.g. optical flow) with deep learning. The **WildLive** system (2025) specifically targets *onboard tracking of wildlife in high-resolution (4K) drone video* [27] [28]. It runs on a Jetson AGX Orin and achieves near real-time performance (17+ FPS at 1080p, ~7 FPS at 4K) by using a sparse **Lucas-Kanade optical flow tracker** for most frames and only running the heavy YOLO-based detector intermittently [29] [30]. By focusing expensive CNN inference on "regions of uncertainty" and tracking objects with cheap point trackers between detections, the system accelerates overall throughput [30] [31]. This hybrid approach yielded accurate multi-animal tracking onboard a drone, and the authors demonstrated it in field flights over a Kenyan conservancy, releasing both the **WildLive dataset** (200k+ annotated animal instances) and their source code to the community [32] [33]. Another frontier is **cross-scene MOT** for drones: Wang *et al.* (2023) tackled the challenge of a UAV tracking targets through highly dynamic scenarios (varying altitude, motion, etc.) [34] [35]. They proposed a meta-learning enhanced tracker that can adapt to new scenes on the fly, and a re-identification method fusing camera and drone state (IMU/GPS) data via Dempster–Shafer theory [36] [37]. Their framework included a *"lightweight detection module"* with attention mechanisms to handle small aerial targets [38], and a trajectory predictor using an LSTM improved by Model-Agnostic Meta-Learning [39] [40]. A new **MIDDTD dataset** (Multi-Info Drone Detection and Tracking) was introduced with diverse scenes and drone metadata to evaluate such cross-scene trackers [41] [42]. This indicates a growing interest in **domain-specific MOT solutions for drones**, although many of these advanced techniques are still tested offline or on high-end hardware. On actual micro drones, simpler real-time tracking-by-detection (possibly augmented with occasional ReID or flow tracking) remains the pragmatic choice.

Embedded Hardware and Acceleration Techniques

Embedded AI hardware has rapidly improved in recent years, enabling more complex vision models to run onboard small drones. NVIDIA's Jetson series (Nano, TX2, Xavier NX, Orin Nano, etc.) provides GPU acceleration in a compact form factor, while alternatives like the Raspberry Pi rely on CPU or external accelerators (Google Coral Edge TPU, Intel Movidius NCS2) for neural network inference. A 2024 review noted that the Jetson Xavier NX outperforms Jetson Nano and Raspberry Pi 4 + NCS2 setups in real-time object detection workloads [43]. In fact, the Jetson Xavier NX can achieve ~30 FPS with a YOLOv7-tiny model, versus single-digit FPS on the Nano or Pi [3]. The **Jetson Orin Nano (2023)** has further closed the gap, delivering up to 20 TOPS of AI performance. As mentioned, tests show it can run modern detectors at tens of FPS with modest power draw [4] [44]. Among CPU-based platforms, the **Raspberry Pi 5** (2023) improved on its predecessors but still only reaches a few FPS on small YOLO models without acceleration [45] [46]. Thus, **neural accelerators** are essential on Pi-class boards: the Coral Edge TPU, for example, can run quantized MobileNet-SSD detectors at over 100 FPS in ideal conditions [47] (though real-world performance is lower due to I/O overhead). One study demonstrated YOLOv5s running on a Coral USB accelerator attached to a Pi, but noted the USB2 vs USB3 connection significantly throttles throughput (USB2 incurred a 3× slowdown) [48] [49]. Overall, leveraging hardware like the **Edge TPU, NCS2, or GPU Tensor Cores** is critical for achieving real-time inference on small drones.

In tandem, researchers apply a variety of **model optimization techniques** to maximize inference speed on constrained hardware. **Quantization** is widely used: converting 32-bit floating point networks to 8-bit integer precision can dramatically speed up inference on supported hardware (TPUs, many DSPs) and reduce memory usage. The trade-off is slight accuracy degradation if not carefully done [50] [51]. Many UAV vision models now use INT8 or FP16 weights. For instance, TakuNet (2024) – a tiny CNN for drone image classification – was trained with FP16 and deploys via NVIDIA TensorRT to exploit low-bit operations, achieving an impressive 650 FPS on a Jetson Orin Nano (15 W) [52]. **Pruning and weight compression** are also explored: redundant channels/filters are pruned to lighten models [50]. This must be balanced to avoid too much accuracy loss. The optimized YOLOv3 by Zhu *et al.* effectively pruned YOLOv3 by replacing layers and heads, yielding a model an order of magnitude faster on Jetson Nano with minimal accuracy drop [53]. Another tactic is using **neural architecture search (NAS)** or manual architecture redesign to create efficient models. We've seen efficient backbones (MobileNetV3, ShuffleNet, etc.) and novel blocks (e.g. CSP, depthwise conv) adopted in nearly all recent detectors [54] [55]. Many of these ideas were crucial for producing lightweight drone-ready networks. Additionally, **inference optimizers and libraries** play a big role on embedded devices. NVIDIA's TensorRT is frequently used to compile models to optimized GPU kernels, often doubling FPS compared to naive execution [56] [57]. For CPUs, frameworks like TensorFlow Lite and OpenVINO are used to leverage NEON instructions and multi-threading. In a late-2024 benchmark, Alqahtani *et al.* deployed various detectors (YOLOv8-n/s/m, EfficientDet-Lite0-2, SSD) across devices using TensorRT on Orin Nano and TFLite on Raspberry Pi/TPU, carefully measuring energy and latency [14] [4]. They found the Orin Nano in TensorRT mode offered the best overall speed/accuracy/energy balance – e.g. only 16 ms inference for YOLOv8-n and ~20 ms for EfficientDet-Lite on Orin [15] [4] – while the Pi 4 with Edge TPU achieved faster inference per watt on some models but at a cost of lower accuracy due to aggressive quantization [44] [58]. This underscores that **system-level optimization (choosing the right model, precision, and runtime for the hardware) is as important as the model design itself**.

Onboard Perception-Control Integration

A critical aspect of these advances is how detection and tracking results are integrated into the **drone's control and navigation system**. Running perception onboard only becomes useful if the drone can autonomously react to the identified objects in real time. Many researchers leverage the Robot

Operating System (ROS) or similar middleware to connect vision outputs to flight controllers. For example, object detection packages running on a Jetson can publish target coordinates to a PX4 flight controller to implement "follow that object" behaviors (community projects have demonstrated a Jetson Nano controlling a drone to track a selected object via live video [59]). In the academic works reviewed, several explicitly close the loop between perception and control. The *Dual-Stage UAV Processing Architecture* by Ntousis *et al.* (2025) uses an onboard RPi+NCS2 to detect vehicles (via YOLOv5n) and immediately adjusts the drone's heading towards the detected target, while a ground server handles more advanced trajectory planning [16] [60]. Their system maintained a high-speed link to offload heavy computations, but the initial detection and guidance commands were generated onboard in real time. Similarly, the WildLive system aims for **animal-reactive flight**: the eventual goal is for the UAV to autonomously adjust its path to keep animals in view or avoid startling them, using the onboard detection/tracking feed [27] [61]. They note this capability is essential for beyond-visual-line-of-sight wildlife monitoring, where relying on a remote pilot or data link is impractical [27] [62]. Initial tests of WildLive were run with manual supervision, but it demonstrated that the drone could *onboardly* identify multiple animals and their trajectories in a large 4K scene [29] [25], which is a step toward closed-loop animal-following drones.

In many cases, **state estimation and sensor fusion** are integrated with vision to improve robustness. Drones have onboard IMUs, GPS, altitude sensors, etc. Some recent trackers (e.g. Wang *et al.* 2023) explicitly incorporate drone state into the tracking algorithm to help predict target motion relative to the moving camera [63] [37]. This hints at a broader integration where the vision module might send *observations* to the flight control stack, which fuses them with inertial navigation data. For instance, a detected object's bearing and range (if depth is available or via size cues) could be treated as a measurement in the drone's EKF, enabling target localization and pursuit. There have also been demonstrations of visual odometry and SLAM running alongside object detection on micro drones, so that obstacles/objects are mapped and avoided in real time. The tight coupling of **perception and control** is facilitated by middleware and frameworks. DJI's SDK and autonomous drone platforms (like the Parrot ANAFI Ai) have started to allow running custom neural networks onboard, feeding the results into their guidance systems. On the research side, open-source frameworks like **AirSim** and **Flightmare** (a simulator from 2021) have been used to simulate how a drone's control responds to onboard vision outputs, before real flight trials. Overall, the last five years show a move from using onboard vision purely for data collection to using it for **closed-loop autonomy**, where the drone can adjust its flight path on the fly based on what it sees. Achieving low-latency processing (often requiring the optimizations discussed earlier) is key to such integration, so that control decisions (e.g. to avoid an object or track a person) are made quickly enough for stable flight.

Benchmarks and Comparisons

To quantify progress, researchers have published **benchmark studies** comparing models and hardware for drone vision. One comprehensive benchmark is **YOLOBench (2023)**, which evaluated 550+ YOLO-family models across 4 different embedded hardware setups (x86 CPU, ARM CPU, Nvidia Jetson GPU, Edge TPU) [64] [65]. By measuring accuracy (mAP) vs latency for each model, they constructed Pareto-optimal frontiers for different platforms. An interesting finding was that **older YOLO models (e.g. YOLOv3, YOLOv4)**, when retrained with modern techniques, can be as efficient as newer models on certain edge hardware [66]. For example, on a Raspberry Pi 4 CPU, a pruned-and-quantized YOLOv4 might achieve a better accuracy-speed trade-off than an out-of-the-box YOLOv8, due to the latter's greater complexity [65] [67]. Another benchmark (Santos *et al.* 2024) specifically measured *YOLOv7-tiny* on three devices, as noted earlier: **RPi 4 (0.9 FPS), Jetson Nano (~6 FPS), Xavier NX (30 FPS)** [3]. They also observed that Jetson Nano's two power modes make a difference (max-N mode yielded ~7.4 FPS vs ~5.2 FPS in low-power mode) [3]. A related 2024 study tested *YOLOv7-tiny with TensorRT* on the Jetson Xavier NX and achieved the full 30 FPS at just 10 W power, which demonstrates how much optimization

can improve edge performance [68] [69]. For Raspberry Pi-based setups, benchmarks indicate that without offloading, high-performance real-time detection is infeasible – hence the reliance on USB accelerators. A comparison of the **Intel Movidius NCS2 vs Google Coral TPU** on a Raspberry Pi 4 showed that the Coral can reach ~15–20 FPS on MobileNet-SSD at 300×300 resolution, about 3× faster than the NCS2 on the same task [47] [70]. However, the Coral's requirement of 8-bit quantization meant the accuracy on COCO dropped a few points compared to FP16 models on Jetson [44] [71].

Table 1 summarizes some performance metrics from recent works, illustrating the range of trade-offs:

Model & Hardware	Accuracy (mAP)	Speed (FPS)	Notes
YOLOv7-tiny on **Raspberry Pi 4 CPU**	53% (COCO val)	**0.9 FPS** [3]	No accelerator; very slow.
YOLOv7-tiny on **Jetson Nano**	53% (COCO val)	**~6 FPS** [3]	~7 FPS in 10 W mode; limited by 128 CUDA cores.
YOLOv7-tiny on **Jetson Xavier NX**	53% (COCO val)	**30 FPS** [3]	Real-time achieved with TensorRT optimizations.
YOLOv8-Nano on **Jetson Orin Nano**	~40% (COCO val)	**60 FPS** [4]	FP16 TensorRT engine; ~16 ms inference.
EfficientDet-Lite0 on **Jetson Orin Nano**	~32% (COCO val)	**50 FPS** [4]	Small TPU-optimized model, GPU TensorRT used.
YOLOv5n on **Raspberry Pi + NCS2**	~45% (COCO val)	**5–8 FPS** (est.)	Used in dual-stage system [16] (vehicles).
YOLOv5s on **Raspberry Pi + Coral TPU**	~47% (COCO val)	**16 FPS** [48] [49]	USB3 interface; INT8 quantized model.
CenterNet (ResNet18) on **Jetson Nano**	33% (VisDrone) [72] [73]	**4.5 FPS** [9]	Small-object focused detector (Drones 2023) [2].
WildLive (YOLO+flow) on Jetson Orin	85% (wildlife dataset)	**17 FPS** (1080p) [28] [30]	Hybrid tracking mode; 7 FPS at 4K resolution.

Table 1: Performance examples of object detection/tracking models on embedded platforms. Accuracy figures are approximate for context (different datasets), and FPS measured under specific conditions per cited sources.

These comparisons show that **real-time (≥30 FPS) object detection on micro UAVs is now achievable** on higher-end embedded boards (Xavier NX, Orin), while lower-end setups (Nano, Pi) typically reach 5–15 FPS without offloading. Techniques like quantization and TensorRT provide significant boosts, often 2–3× speedups [11] [13]. It is also evident that there is no one-size-fits-all solution: the best model depends on the hardware and the task. For instance, a tiny model might suffice for obstacle avoidance (where ~60% mAP on simple objects is fine) but a surveillance drone might require a larger model for reliable person detection, accepting lower FPS or using a stronger onboard computer.

Open Source Implementations and Datasets

The rapid progress in this field has been supported by the release of **open-source code and datasets** tailored to drones. The **VisDrone dataset** (2018–2021) and **UAVDT** provided thousands of annotated

images and videos of urban scenes from drones, spurring research into aerial pedestrian and vehicle detection. By 2020, VisDrone's challenge had a dedicated MOT track, and many of the lightweight models reviewed (e.g. YOLO variants, CenterNet adaptations) report results on it [74] [75]. More recently, specialized datasets have emerged: the **AU-AIR dataset** (2020) focuses on UAV imagery with objects annotated for autonomous navigation scenarios; the **Drone-Crowd** dataset (2021) provides dense annotations for crowd counting and tiny person detection from drones. In 2023–2024, as mentioned, **WildLive** released a large wildlife tracking dataset [32], and **MIDDTD** (2023) was made available with multi-modal data for drone tracking [41] [42]. These datasets enable researchers to train and evaluate models under realistic conditions (e.g., animals in savannah terrain for WildLive, or mixed urban-rural scenes with drone telemetry in MIDDTD).

On the implementation side, many researchers have shared their frameworks. The WildLive project provides all source code and trained model weights for its YOLO-based detection and LK tracking framework, accessible on GitHub [33]. This allows the community to reproduce and build upon their near-real-time tracking results. The YOLOBench benchmark similarly released its code and the 550+ model zoo for others to analyze or use in NAS research [67]. Popular open-source repositories like **Ultralytics YOLOv5/YOLOv8** (which are frequently cited in papers) have made it easy to obtain and modify state-of-the-art detectors for custom drone tasks. There are also ROS packages integrating these detectors, and drone control software (e.g., **DJI UX SDK**, **MAVSDK**) with examples of using onboard vision. A notable open project is the **search-and-rescue drone with Coral TPU** on Hackster.io [76], which provides a step-by-step implementation of a drone using a Pi + Coral to detect and localize missing persons. Academic efforts like the dual-stage architecture [16] [77] and cross-scene tracker have pseudocode or algorithm details in their papers that can guide implementation. As hardware becomes more accessible, we also see **community benchmarks** (on forums and GitHub) comparing, say, TensorFlow Lite vs. TensorRT on the same Jetson, or testing new models like YOLOv8 on the Jetson Nano – these informal reports often complement academic studies and help practitioners choose the right setup.

In summary, the ecosystem of **datasets and open implementations** has matured from general-purpose to drone-specific. This accelerates development of better models: for instance, having aerial datasets allowed researchers to fine-tune detectors for the unique appearance of objects from a bird's-eye view (small, occluded, different angles) [78] [79]. Open benchmarks and code ensure that claims of "real-time" truly translate on the target hardware, and enable continued improvements. The community's commitment to open science in this domain is evident from the number of releases in the last few years.

Conclusion

Efficient real-time vision on micro aerial robots has advanced markedly since 2020. Researchers have devised **ultra-lightweight object detectors**, from tiny YOLO variants to custom networks, that push the limits of what can run on a few watts. Multi-object tracking approaches have been tailored for UAV constraints, often by smartly reducing workload (e.g. combining fast optical flow with intermittent CNN detection). These perception modules are increasingly being integrated with **onboard flight control**, moving drones closer to true autonomy in complex missions. While a sub-250g quadrotor may still struggle to carry hardware for high-resolution video analytics at high FPS, the trend of more powerful yet efficient edge AI devices (like the Jetson Orin Nano and upcoming Qualcomm drone AI SOCs) is rapidly closing that gap. The latest works demonstrate that even tasks like 4K animal tracking or on-drone pedestrian detection are within reach of onboard processing [30] [80]. Continued innovation in model compression, hardware acceleration (including dedicated AI chips for drones), and cooperative architectures (onboard vs offboard processing) will further improve performance.

Crucially, the **real-world deployments** reviewed – from wildlife conservation drones to search-and-rescue prototypes – show that these research advances are not just academic. There are now open-source reference designs and datasets that pave the way for engineers to deploy object detection and MOT on their own UAV platforms. In the coming years, we can expect micro drones to reliably detect and track multiple objects in real time, enabling applications like autonomous inspection, tracking moving subjects for filming, and smarter collision avoidance. The progress from 2020 to 2025 in real-time onboard vision is laying the groundwork for the next generation of fully autonomous micro aerial robots.

Sources:

- Lago, Patel & Singh (2024) – *Low-cost real-time aerial object detection and tracking pipeline* 39 38
- Hossain & Lee (2019) – *Onboard multi-object detection/tracking on UAV with GPU* 81
- Zhu et al. (2024) – *Lightweight optimization of YOLOv3 on Jetson Nano* 11 13
- Bhowmik et al. (2023) – *On-board small-scale object detection on Jetson & NCS2* 2 9
- Nguyen et al. (2025) – *WildLive: real-time animal detection/tracking on UAV (Jetson Orin)* 30 25
- Wang et al. (2023) – *Cross-scene MOT for drones with meta-learning (MIDDTD dataset)* 41 40
- Ntousis et al. (2025) – *Dual-stage UAV detection (RPi+NCS2 onboard & YOLOv8x+DeepSORT offboard)* 16 77
- Santos et al. (2024) – *YOLOv7-tiny performance on edge devices (Raspberry Pi, Jetson)* 3
- Alqahtani et al. (2024) – *Benchmark of YOLOv8, EfficientDet, SSD on Pi 3/4/5 + TPU, and Jetson Orin* 15 4
- Lazarevich et al. (2023) – *YOLOBench: 550+ models benchmark on embedded systems* 64 66 .

---

1 3 11 12 13 43 53 68 69 Performance analysis of real-time object detection on Jetson device | Request PDF
https://www.researchgate.net/publication/363515934_Performance_analysis_of_real-time_object_detection_on_Jetson_device

2 5 6 7 8 9 10 72 73 74 75 78 On-Board Small-Scale Object Detection for Unmanned Aerial Vehicles (UAVs)
https://e-archivo.uc3m.es/rest/api/core/bitstreams/27b16e8c-e6bb-46ae-b0b3-de1efa5bae0a/content

4 14 15 44 45 46 58 71 Benchmarking Deep Learning Models for Object Detection on Edge Computing Devices
https://arxiv.org/html/2409.16808v1

16 17 24 60 77 A Dual-Stage Processing Architecture for Unmanned Aerial Vehicle Object Detection and Tracking Using Lightweight Onboard and Ground Server Computations
https://www.mdpi.com/2227-7080/13/1/35

18 19 25 26 27 28 29 30 31 32 33 61 62 WildLive: Near Real-time Visual Wildlife Tracking onboard UAVs
https://arxiv.org/html/2504.10165v1

20 21 22 23 34 35 36 37 38 39 40 41 42 63 79 Cross-Scene Multi-Object Tracking for Drones: Leveraging Meta-Learning and Onboard Parameters with the New MIDDTD
https://www.mdpi.com/2504-446X/9/5/341

47 Models - Object Detection - Coral
https://coral.ai/models/object-detection/

48 49 70 riverpublishers.com
https://www.riverpublishers.com/downloadchapter.php?file=RP_9788770227902C11.pdf

50 51 52 54 55 56 57 TakuNet: an Energy-Efficient CNN for Real-Time Inference on Embedded UAV systems in Emergency Response Scenarios
https://arxiv.org/html/2501.05880v2

59 Object tracking system - Jetson Nano - NVIDIA Developer Forums
https://forums.developer.nvidia.com/t/object-tracking-system/178600

64 65 66 67 [2307.13901] YOLOBench: Benchmarking Efficient Object Detectors on Embedded Systems
https://arxiv.org/abs/2307.13901

76 Search And Rescue Drone With Google Coral - Hackster.io
https://www.hackster.io/bandofpv/search-and-rescue-drone-with-google-coral-a485c7

80 Towards Real-Time On-Drone Pedestrian Tracking in 4K Inputs
https://www.mdpi.com/2504-446X/7/10/623

81 Deep Learning-Based Real-Time Multiple-Object Detection ... - MDPI
https://www.mdpi.com/1424-8220/19/15/3371