

Towards Wearable Mishap Avoidance

Qifei Dong, Mihir Bala, Jingao Xu, Padmanabhan Pillai,
Roberta Klatzky, Mahadev Satyanarayanan

August 2026
CMU-CS-26-131

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We introduce *Wearable Mishap Avoidance (WMA)*, in which always-on AI for a wearable device detects impending mishaps in an imperfect legacy system (human or cyber-physical) and issues a corrective alert before it is too late. The time-critical sensing and actuation loop of this new class of applications involves a tradeoff between rapid reaction (to meet externally-imposed mishap deadlines) and patience (to allow the legacy system to self-correct in time). We show that WMA can be implemented today using off-the-shelf wearable hardware, wireless networking, edge computing, and AI models. We present a model for reasoning about WMA, implement proofs of concept for three different use cases, and discuss future research directions.

This research was supported by the National Science Foundation under grant number CNS-2106862. 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 funding sources.

Keywords: Real-time Mobile AI, Edge AI, Edge Computing, Machine Learning, Wearable Cognitive Assistance, ADAS, OODA Loop, Accident Prevention, Online Action Detection, Drone Safety

1 Introduction

Mishaps due to distraction, forgetfulness, fatigue, visual impairment, or other human limitations impose significant pain and expense. According to the US Center for Disease Control [1], “One in four adults aged 65 and older experience a fall each year. Falls are a leading cause of both fatal and non-fatal injuries.” In manufacturing, estimates suggest that human error can account for 5% to 30% of total manufacturing costs, and contribute to up to 80% of quality defects [2, 3]. In air traffic control, there is growing evidence that the distraction of personnel led to the March 2026 collision of an Air Canada aircraft with a truck at La Guardia airport [4].

Can real-time AI on wearable devices be made fast and accurate enough to reduce the occurrence of mishaps? We use the term *Wearable Mishap Avoidance (WMA)* for this class of assistive applications. When our sight degrades, we wear glasses. When our hearing degrades, we get hearing aids. Can human cognitive degradation be corrected by wearable real-time AI? Wearability enables the corrective functionality to move with the user, rather than being confined to specific locations. A key challenge is that wearable devices are optimized for size, weight, and power (SWaP). A typical hearing aid weighs 1.5–3 g and a pair of eyeglasses weighs 25–40 g. The lightest smartglasses today weigh on the order of 50–60 g, and the lightest smartwatches weigh on the order of 35–40 g. It is unknown whether the latency and accuracy of sensing and AI processing implementable within such tight SWaP constraints is adequate for WMA. While offloading to a cloudlet can help, the resulting network delays eat into the available latency budget.

In this work, we ask if WMA implemented using a commercial off-the-shelf (COTS) SWaP device, wireless edge offload, and off-the-shelf AI models can meet stringent end-to-end requirements to avoid mishaps. To this end, we implement *Wingman*, a proof-of-concept WMA system whose execution we model as an OODA (Observe, Orient, Decide, Act) loop (§2.3). While WMA applications are diverse in their sensing, AI, and logic, they have a deep underlying unity that we abstract into a single framework with reusable design principles: dual concurrent OODA loops, backwards timing budgets derived from measured correction times, and a single tunable parameter balancing prevention against false alarms. We apply these principles to three different case studies, define key performance indicators (KPIs), and experimentally validate that WMA is not a distant dream but achievable today. This paper makes the following contributions:

- Introduction of the WMA concept as a real-time AI problem for SWaP devices.
- Formulation of an abstract reasoning framework for WMA, centered on dual concurrent OODA loops.
- Implementation of the proof-of-concept Wingman system, with three exemplar applications.
- Framing of open research directions for WMA, based on study of the exemplars.

2 Background and Related Work

2.1 Mobility Penalty of SWaP Devices

Wearable devices suffer a *mobility penalty* [5, 6] because Moore’s Law affects server and mobile hardware differently. Servers manifest all their improvement in compute capability. Mobile devices express at least some of their improvement in SWaP attributes. Empirical data over three

decades confirms a persistent order-of-magnitude performance gap between mobile and server hardware [7]. Recent work [8] shows that the mobility penalty in 2026 remains far too large for SOTA AI algorithms to be executed below 100 ms. While mobile devices will continue to improve, SOTA AI algorithms will also improve and demand ever-more resources. For the foreseeable future, real-time SOTA AI on a wearable device will require low-latency edge offload to a cloudlet.

2.2 Advanced Driver Assistance Systems

This work is inspired by Advanced Driver Assistance Systems (ADAS) [9] that provide safety features such as pre-collision warning, safe car spacing, and lane-keeping assist. ADAS has been deployed worldwide for over a decade. Over a 30-year period, it is expected to prevent 37 million crashes, 14 million injuries, and 250,000 deaths [10]. Today, ADAS is implemented fully in-vehicle because vehicles can carry heavy sensors, abundant power, and GPU-rich computers. In contrast, WMA applications running on wearable devices may need to use edge offload.

2.3 The OODA Loop

The OODA loop abstraction has been used since the 1950s to study human reaction times in tightly-coupled cyber-human systems [11]. More recently, it has been used to analyze human-AI hybrid decision-making pipelines [12, 13]. Bala et al’s characterization of AI offload from a drone to a cloudlet in the SteelEagle system [14] can also be applied to Wingman. Like ultralight drones in SteelEagle, wearable devices in Wingman use edge offload to overcome the mobility penalty. Figure 1 maps Wingman’s end-to-end sensing, processing, and actuation pipeline to its OODA loop components. The “Observe” phase consists of on-device sensing (a), on-device pre-processing such as video encoding (b), and network transmission from device to cloudlet (c). On closed-source COTS devices, (a) and (b) may not be independently measurable; only their sum, Observe_{ab} , can be measured. Observe_c corresponds to “uplink” network latency. The AI processing on the cloudlet maps to both the “Orient” and “Decide” phases of the OODA loop. It is shown as a single monolithic entity, Orient+Decide_d , in Figure 1. The Orient phase consists of both low-level perception (such as object detection) and higher-level context establishment (such as realizing that the recognized object is a hazard or threat). In Wingman, the Orient phase is where a potential mishap is detected. The Decide phase corresponds to the processing needed to decide whether to raise an alert. This may not be a trivial decision because it may have to take into account the usability impact of false positives, the risk of false negatives, and the possibility of self-correction for true positives. The “downlink” network transmission, Act_e in Figure 1, conveys the outcome of the offloaded AI even if no alert is triggered. Only rarely is Act_{fg} , the cost of preparing and actuating the alert, incurred.

2.4 Wearable Cognitive Assistance

WMA superficially resembles WCA, which has been extensively studied [15, 16, 17, 18, 19, 20, 21]. In both cases, a wearable device transmits first-person video to a cloudlet that performs AI processing in real time and returns guidance. However, the similarity ends there. At a deeper level, WMA and WCA have totally different goals, constraints, and implementations. Inspired by GPS navigation apps, WCA provides step-by-step guidance on an unfamiliar task. The user’s speed

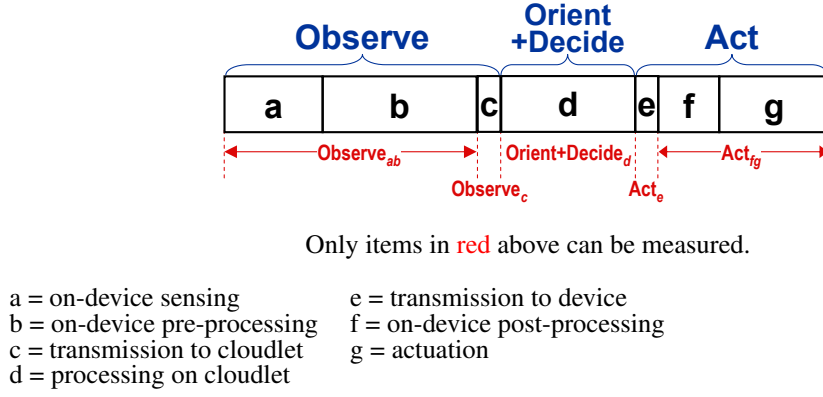


Figure 1: Wingman OODA Loop

of execution on the task is gated by the WCA app: she cannot proceed to the next step before receiving guidance. WCA can detect an error made by the user only *after* it has been committed.

In contrast, WMA is inspired by ADAS, whose goal is help to reduce mishaps *before* they occur. A user's speed of execution of a task is completely independent of the WMA mechanism. Indeed, it is possible for a user to be too fast for WMA. Most importantly, WMA has to infer an imminent mishap in time to avoid it. The sensory evidence available for this is likely to be much more tenuous than in WCA, where the error has already been committed. Both in terms of stringent real-time execution bounds and in the complexity of AI processing, WMA is much more difficult than WCA.

3 Model of WMA

3.1 Misperformance of Tasks

A mishap is an unfortunate accident arising from a task that is misperformed by an imperfect legacy system, which may refer to cyber-physical systems or human beings. For example, elderly adults fall when their impaired sense of balance fails to signal an unstable posture in time for their slowed motor system to implement a correction. Similarly, defects in factory-assembled products can arise from lapses of attention by humans charged with assembly. Such defects may also arise in very old factory automation systems whose degraded sensing or actuation capabilities are unrepairable.

There is a long tradition of using fail-safe mechanisms for reducing the occurrence or impact of mishaps. For example, guard rails prevent a visitor from falling into an abyss from a misplaced step. Safety nets save a worker who falls from scaffolding. Air bags inflate upon vehicle collision. Hospital beds have rails to prevent patients from falling out. A life preserver ensures that a person who falls from a boat doesn't drown. Wingman is conceived in the spirit of these well-known mechanisms. It implements sensing and reasoning that is external to a legacy system that is performing a task. It continuously observes task performance, and rapidly alerts the legacy system if a mishap appears imminent. For the alert to be useful, it has to be delivered fast enough for the legacy system to correct in time to avoid the mishap.

3.2 Two Concurrent OODA Loops

WMA spans two concurrently-executing independent systems: the closed-source legacy system and its Wingman system. The Wingman system is custom-designed to complement the legacy system: its design, sensing modalities, and AI algorithms are tuned to the specific tasks and known defects of the legacy system. A one-size-fits-all universal Wingman system is beyond our ability to implement at this time. Both the legacy system and Wingman independently sample the real world at their respective rates and execute their own OODA loops to process real-time observations. When appropriate, Wingman triggers an alert that is delivered to the legacy system. From the viewpoint of the legacy system, Wingman can be viewed as “an angel on the shoulder” that stays silent unless a mishap is about to occur, and only then rapidly alerts.

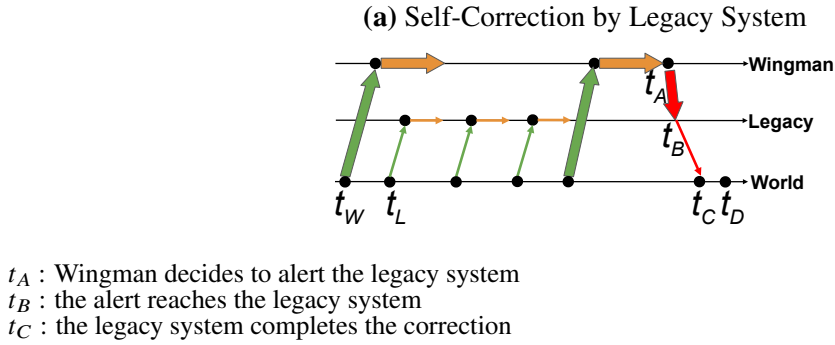
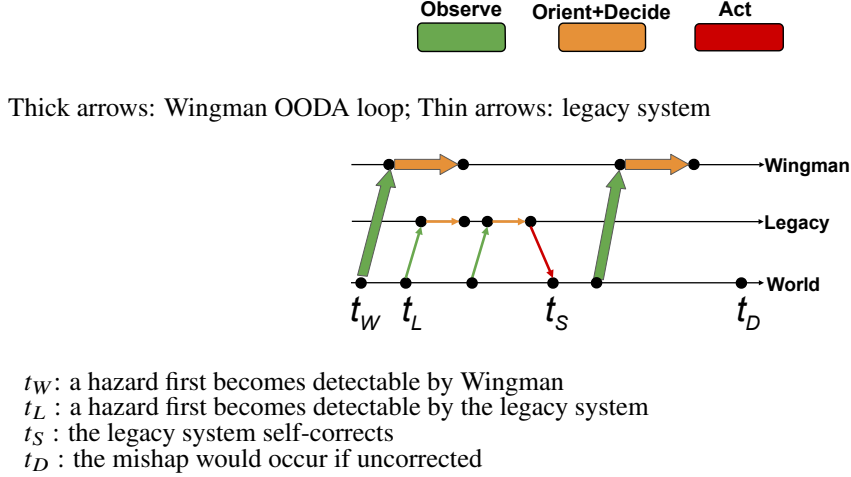
Legacy System: This is the imperfect autonomous actor performing a task. It can be a human (e.g. a person walking, a pilot steering a drone, a factory worker assembling a part) or a robotic system executing a motion plan. The legacy system has its own objectives, sensing capabilities, and operational logic, and executes its own OODA loop to carry out its task. It observes its surroundings, orients itself based on its current knowledge, decides on a course of action, and acts upon it. If this internal OODA loop runs perfectly, every potential mishap can be avoided without external help. However, the legacy system is inherently susceptible to limitations such as distraction, fatigue, sensory degradation, or imperfect algorithms. As a result, an error can arise at any stage of its OODA loop. It may fail to observe a critical detail, orient incorrectly to a situation, make a wrong decision, or act in a dangerous way, any of which can put it on a path towards a mishap.

Wingman: This is an independent external layer of sensing and AI processing designed to monitor the legacy system and its environment. Wingman’s sensors and processing units are not directly accessible to the legacy system, and vice versa. In our implementations, Wingman runs on a wearable device and processes camera streams with on-device or edge-based AI models. Its role is to infer whether the legacy system is approaching a mishap state and, if so, to deliver a timely alert. Because Wingman operates out-of-band, it can see the situation from a different perspective and detect risks that the legacy system cannot fully perceive or react to in time.

Corrective AI: When the OODA loop of the legacy system fails to resolve a prospective mishap situation due to imperfection, Wingman becomes useful as a second line of defense. The Wingman system also runs its own OODA loop in parallel to the legacy system. It observes the legacy system and the environment through sensor streams, orients using real-time AI, decides whether an immediate intervention is necessary to prevent a mishap, and acts accordingly by staying silent or issuing an alert. Wingman can successfully prevent a mishap only if the alert reaches the legacy system before the situation becomes irrecoverable. Therefore, a low end-to-end latency of Wingman’s OODA loop is essential for timely intervention.

3.3 Mishap Avoidance Timeline

To describe mishap avoidance more precisely, we consider a *prospective mishap situation* in which the current task trajectory of the legacy system is heading towards a mishap. This situation is initiated by a *hazard*: a specific condition, obstacle, or flawed action that can be detected as a signal by both Wingman and the legacy system. If left unaddressed, the hazard will likely lead to



(b) Avoidance via Wingman Alert

Figure 2: Timeline for Mishap Avoidance

a mishap in the near future. After a hazard arises, the mishap can be avoided via one of the two timelines shown in Figure 2(a) and 2(b).

Self-correction (Figure 2(a)): The legacy system successfully processes the hazard through its own OODA loop. As shown in Figure 2(a), the hazard becomes conceivably detectable by the legacy system at t_L , though it may not lead to an immediate recognition of the situation. After a short period of time, the legacy system orients to the danger, makes the correct decision, and performs an action to return to a safe state at t_S , before the mishap deadline t_D . This happens without any need for external intervention. Concurrently, Wingman runs its own OODA loop. The hazard is detectable by Wingman after time t_W and disappears after time t_S . If Wingman were to alert as soon as it detects the hazard, the legacy system may not have a chance to self-correct before it is alerted. We classify this as a *false alarm*.

Saved by Wingman (Figure 2(b)): This path of mishap avoidance occurs when the OODA loop of the legacy system fails to process the hazard. As shown in Figure 2(b), the legacy system never completes an action to resolve the potential mishap via its own OODA loop. We refer to this case as an *imminent mishap*, in which a mishap is inevitable if there is no Wingman. In Figure 2(b), Wingman concludes at t_A that self-correction is not going to occur in time to avoid the mishap. It therefore issues an alert. The legacy system receives the alert at t_B , and completes the corrective

action at t_C to avoid the mishap before the deadline t_D . If Wingman’s own OODA loop is too slow or too inaccurate, it may not detect the hazard in time for mishap avoidance. Its alert that arrives at t_B may be too late for the legacy system to correct before t_D . This is a *mishap pass-through*.

In practice, t_L and t_W can occur in either order. When the hazard arises from an existing condition in the environment, Wingman’s superior perception may detect it earlier. On the other hand, if the hazard arises from an action performed by the legacy system, its better observation perspective or closer distance to the hazard may enable earlier perception. Even in the latter case, however, the earlier perception may not result in earlier recognition of the hazard due to factors such as distraction. The best-case situation for Wingman is to issue an alert as soon as it detects a hazard, ignoring the possibility of timely self-correction by the legacy system. This may be the optimal approach when the legacy system is not a human, and shows Wingman in the best light for mishap avoidance. If the legacy system is a human, however, annoyance may result from a hasty alert that arrives when self-correction was already beginning. This is a difficult needle to thread.

4 Problem Formulation and KPIs

4.1 The Essence of Mishap Avoidance

Using the notation from Section 3.3 and Figure 2, we give an abstract formulation of the mishap avoidance problem. At the core of mishap avoidance is a real-time binary classification problem over a sequential stream of events. At each time step, Wingman observes the world’s state. An AI prediction model $p(\cdot)$ maps current and recent observations to the likelihood of a mishap happening in the near future. A threshold, θ , is used to compute a binary decision variable $M(t)$ at time t :

$$M(t) = \begin{cases} 1 & \text{if } p(state_t) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where $state_t$ is the latest observation(s) available at time t . A decision of $M(t) = 0$ is the base case, in which Wingman remains silent, allowing the legacy system to proceed uninterrupted. A decision of $M(t) = 1$ triggers an alert.

A successful avoidance is made when two conditions hold. First, the AI model correctly recognizes a likely mishap state before it is reached: i.e., before t_D in Figure 2(b). Second, Wingman triggers an alert early enough for the legacy system to react in a timely manner, and hence avoid the mishap: i.e., $M(t) = 1$ occurs early enough in the timeline of Figure 2(b) for the correction at t_C to happen before t_D . This has to factor in any explicit delays to allow for self-correction. In Figure 2(b), Wingman decides to alert at time t_A where $p(state_{t_A}) \geq \theta$. Under the assumption that $p(\cdot)$ increases as the potential mishap approaches, the threshold θ embodies how aggressively alerts are made. Setting a lower θ will move the decision time t_A earlier in the timeline, leading to an early alert with a lower confidence. This improves the likelihood of avoidance, but also increases the chance of false alarms, unnecessary workflow interruptions, and possible user annoyance. Higher θ values will tend to postpone the alert until a higher confidence, thereby reducing the above negatives but risking mishap avoidance. Empirical data from the actual legacy system in question, combined with user preference, will be needed to optimally set the value of θ .

4.2 Resulting KPIs

This framing of mishap avoidance leads to three key performance indicators (KPIs). We describe them below, and use them as the basis of our case study evaluations (§6–§8).

- *Wingman OODA Loop Latency* ($T_{OODA.Wingman}$) reflects the computation and communication latencies that limit how fast a mishap can be detected and signaled. A necessary condition for mishap avoidance is:

$$T_{OODA.Wingman} + T_{correction} < t_D - t_W$$

where $T_{correction}$ is the time required by the legacy system to perform a correction once alerted by Wingman, and is equal to $t_C - t_B$ in Figure 2(b). Thus, the latest successful alert must be delivered by $t_D - T_{correction}$.

- *Wingman Prevention Rate* (WPR) equals the fraction of imminent mishaps that are successfully avoided because Wingman alerted in time. Note that this is different from the True Positive Rate (TPR) of the AI decision model. WPR is limited by TPR since a correct but late intervention can still result in a mishap. Equivalently, the mishap pass-through rate is the inverse of the prevention rate which equals the fraction of imminent mishaps that are not prevented by Wingman.
- *Wingman False Alarm Rate* (WFAR) is the fraction of alerts that are issued when no mishap would have occurred. This situation could arise because of a false positive from Wingman’s AI, or because of self-correction by the legacy system. False alarms interrupt the workflow of the legacy system and thereby impose operational costs. In human-centric systems, a high WFAR can lead to alert fatigue, causing the legacy system to eventually ignore Wingman entirely [22, 23].

4.3 Cost Model

When a task-specific cost model is available, Wingman can also be evaluated with the total expected cost. With the cost of an unprevented mishap C_{mishap} and the cost of disrupting the legacy system with a false alarm C_{FA} , the total expected cost over time can be expressed as:

$$\begin{aligned} \mathbb{E}[\text{cost}] = & \Pr(\text{mishap}) \cdot \Pr(\text{mishap pass-through}) \cdot C_{mishap} \\ & + \Pr(\text{no mishap}) \cdot \Pr(\text{false alarm}) \cdot C_{FA} \end{aligned}$$

where $\Pr(\text{mishap})$ is the prior probability of imminent mishaps and $\Pr(\text{no mishap}) = 1 - \Pr(\text{mishap})$. Given a probability model and expected cost, we can formulate an optimization problem for determining the best threshold, θ^* :

$$\theta^* = \arg \min_{\theta} \mathbb{E}[\text{cost}]$$

Due to the complexity of creating validated models of these costs and probabilities, determining θ^* will have to await real-world WMA deployment.

5 Validating the WMA Concept

A full validation of the WMA concept would require at least three distinct sets of questions to be answered:

- **Mechanism:** How can we use today’s real-time AI technologies to meet the stringent timing and accuracy needs of Wingman applications? How capable are they? How significant are any gaps, and what techniques can we use to bridge these gaps?
- **Policy:** For legacy systems that involve human users, can the AI in Wingman systems be tuned to successfully balance the complex user experience tradeoffs involving timely corrective AI, false positives, false negatives, and self-correction?
- **Experience:** In real-world settings, how effective are Wingman systems in reducing the number of mishaps? Are they able to do this in a way that user acceptance of these systems is high? Can these wins be quantified to show that Wingman systems deliver substantial benefit for their cost and inconvenience?

In this paper, we focus exclusively on the first question (mechanism). At this early stage, just establishing the feasibility of implementing Wingman systems with current hardware and AI technology is nontrivial. We fully recognize the importance of the other two questions, but defer them to future work. To assess the adequacy of today’s real-time mobile AI for Wingman systems, we explore three different case studies of increasing complexity (§6–§8).

6 Case Study 1: Quality Control Mishaps

In spite of industrial automation, many assembly and inspection tasks continue to rely on the visual acuity, mental agility and manual dexterity of human workers. Alas, human performance can be affected by fatigue, sleep deprivation, distraction, etc. as discussed earlier (§1). Our first case study creates a Wingman system for an assembly line worker whose task is to detect and remove unwanted objects from a conveyor belt feed. To ensure reproducible results, our legacy system is a robotic emulation of a human worker rather than a real human worker. The Wingman system observes the legacy system’s actions, detects when it appears to have missed an arriving unwanted item on the conveyor belt, and issues a timely alert to correct the mistake. Since the sensing pipelines of the legacy and Wingman systems are independent of each other, this experimental method does not compromise the validity of our results relative to the KPIs (§4.2).

6.1 Mapping to WMA model

We map the abstract concepts from §3.3 to this case study as follows. A *prospective mishap situation* occurs when an unwanted object arrives on the moving conveyor. A *miss* occurs when the worker fails to remove the target on their own. Without intervention, this will result in a *mishap* when the target object moves down the conveyor belt out of reach. When Wingman’s AI determines a miss is happening, it makes a decision to intervene, alerting the worker. To successfully avoid a mishap, the decision must be made sufficiently early so that the worker can complete any corrective action in time, accounting for alert delivery and worker reaction times. If a mishap still occurs because Wingman fails to predict the error or does so too late for a successful correction, the mishap is unavoidable and we deem this as a *mishap pass-through*. If the worker resolves the prospective mishap situation by themselves, while Wingman wrongly decides that a mishap is imminent, this will result in an unnecessary *false alarm*.

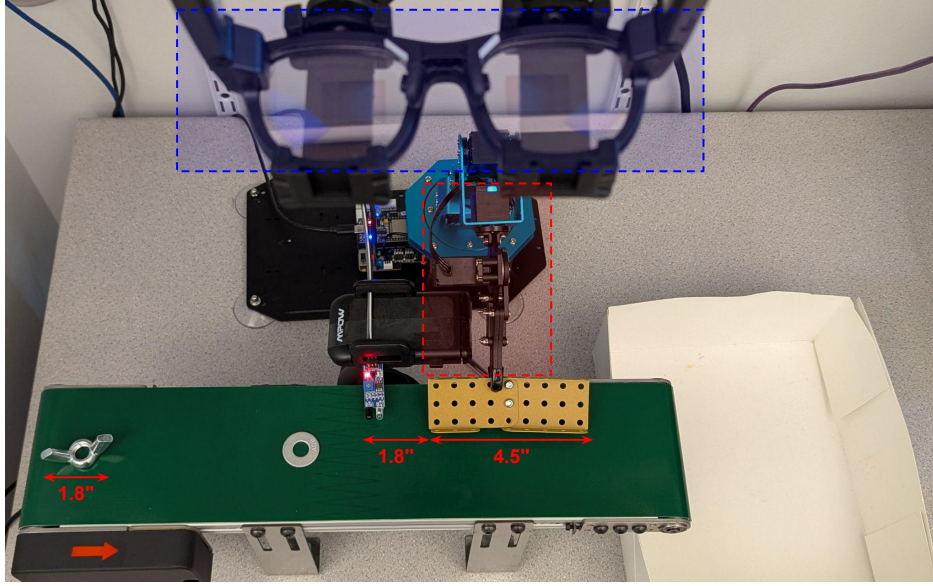


Figure 3: Experimental Setup for Case Study 1

6.2 Experimental Setup

As Figure 3 shows, a 20-inch-long model conveyor belt carries small objects at adjustable speeds between 2.0 and 10.0 inches per second (ips). Two types of objects may appear on the belt: washers (legitimate objects) and wing nuts (unwanted objects). A small, programmable robot arm (shown in the red box in Figure 3) is the legacy system emulating a human worker. The robot senses and classifies objects as they pass under an infrared (IR) LED sensor. When an unwanted object is detected, the arm moves to push it off the conveyor belt. This actuation takes an average of 240 ms to initiate, and a further 320 ms to complete. To emulate human errors, the legacy system is programmed to ignore correctly-detected unwanted objects with a specified probability.

A Wingman application to detect such errors runs on an Android-based RayNeo X2 smartglass (shown in the blue box in Figure 3). It captures a top-down view of the conveyor belt, and runs a customized YOLO object detection model. For each unwanted object detected, it computes the predicted time of arrival at the center of the robot arm's push plate. Wingman also assesses when the legacy system has failed to react to the unwanted object (imminent mishap). If it has high enough confidence in its assessment, it alerts the legacy system through a Bluetooth signal. The alerted legacy system will then make a fast push operation (taking 200 ms) in an attempt to correct the imminent mishap. Note that the Wingman and the legacy systems run completely independent OODA loops, and only interact directly through the alert mechanism.

6.3 Results

6.3.1 Wingman OODA Loop Latency

In this case study, Wingman is implemented entirely on device, so network and cloudlet processing times are not relevant. Video frame capture on the RayNeo smartglass occurs at 30 FPS, but incurs 44 ms latency. On-board object detection takes 130 milliseconds on average. Assessing the motion of the robot arm takes 7 ms on average. The simple decision policy uses a fixed time interval of $T_{wait} = 100$ ms – if there is no robot arm motion within T_{wait} after an unwanted object reaches the

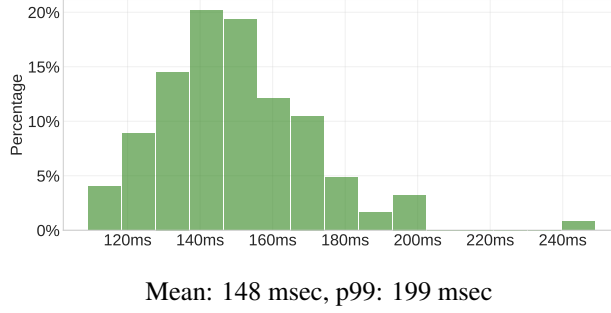


Figure 4: Case Study 1: Wingman OODA-Loop Latency

push plate, an imminent mishap is inferred.

However, the total OODA loop latency is not just a sum of these parts. Wingman may see the target and perform object detection long before the target reaches the worker, so this is not in the critical path. Likewise, T_{wait} only limits how early an alert may occur, not the latency of the OODA loop. On the critical path, only camera capture, motion assessment, and alert delivery over Bluetooth affect Wingman’s OODA loop latency, whose distribution is shown in Figure 4.

6.3.2 Wingman Prevention Rate

We configure the robot to always miss “seeing” unwanted objects, and record the number of mishaps avoided and passed-through by Wingman. We test at different conveyor belt speeds, running 15 trials at each speed. The results are shown by the orange curve in Figure 5. At belt speeds between 2 and 7 ips, all worker misses are detected and corrected, leading to zero mishap pass-through. At higher speeds, there is less time for correction. This causes the mishap pass-through rate (complement of WPR) to ramp up quickly.

These experimental results confirm our modeling. For mishap avoidance to succeed, the sum of Wingman’s wait-and-watch time T_{wait} , alert propagation time T_{alert} , and worker’s correction time $T_{correct}$ needs to fit in the action time window:

$$T_{wait} + T_{alert} + T_{correct} < d_{plate}/v_{target}$$

where the size of the window equals the length of the metal plate divided by the speed of the target. This inequality is satisfied when v_{target} is small, but may not hold when v_{target} is greater than 8 ips and the window size is smaller than 460 ms, due to the variation of T_{alert} and $T_{correct}$.

6.3.3 Wingman False Alarm Rate

We run the robot with no injected errors, and record how many times Wingman generates a false alarm. The results are shown by the blue curve in Figure 5. At speeds below 4 ips, Wingman correctly identifies the worker’s self-correction and stays silent. At higher belt speeds, the time it takes for an unwanted object to move from the IR sensor to the metal plate is decreased, but the worker reaction time remains unchanged. This means that the unwanted object will travel a longer distance before being removed by the worker. With the fixed time decision policy, Wingman’s decision point moves earlier relative to worker action, leading to an increasing number of missed self-corrections. Above 9 ips, Wingman raises false alarms on all unwanted objects.

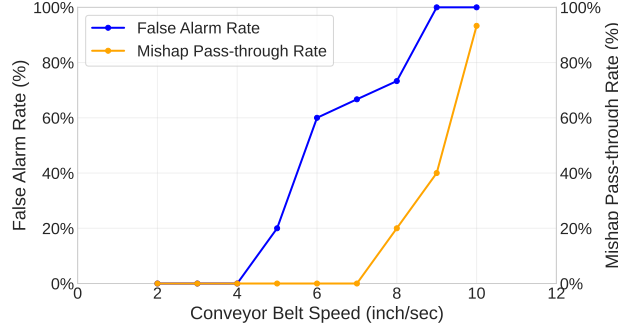


Figure 5: Impact of Conveyor Belt Speed on WPR and WFAR

These experimental results also confirm our modeling. For Wingman to correctly recognize self-correction, the alert decision needs to be made after observing the worker’s action. In other words, the worker’s reaction time T_{react} needs to be less than Wingman’s observation start time plus T_{wait} :

$$T_{react} < (d_{target}/2)/v_{target} + T_{wait}$$

where Wingman’s observation starts when the center of the target reaches the metal plate. This inequality holds until 5 ips, after which the tail of the distribution of T_{react} starts to fall behind Wingman’s decision.

6.3.4 Discussion

Although not our primary focus, we can gain some insight into optimizing Wingman’s alert policy. Specifically, we can use the false-alarm-to-mishap ratio as a proxy for the relative cost of false alarms. A desirable policy could be to minimize the number of mishap pass-throughs while keeping the false-alarm-to-mishap ratio below a predefined value. For example, at a speed of 8 ips, with a wait-and-watch time of 100 ms, Wingman raises false alarms in 11 out of 15 runs; in 3 out of 15 runs, the mishap is not avoided in time. Setting a wait-and-watch time limit of 100 ms thus yields a false-alarm-to-mishap ratio of 3.67. This tradeoff is tunable through the T_{wait} parameter.

7 Case Study 2: Assembly Mishaps

The first case study (§6) involved object classification, a simple enough computer vision task that on-device AI inference provides sufficient accuracy and speed. Our second case study involves the much harder computer vision task of Online Action Detection (OAD) in video sequences [24, 25, 26] for mishap avoidance in procedural tasks. We detect divergence from a canonical procedure and alert for correction before a mishap can occur. We assume that the ground truth finite state machine (FSM) of an assembly procedure is available, and focus on real-time state tracking and divergence detection under streaming video input. To achieve adequate speed and accuracy, AI inference for OAD requires offloading from the wearable device to a cloudlet. Note that some recent work on mistake detection [27, 28, 29] also uses OAD models, but with goals and assumptions inconsistent with a live video pipeline.

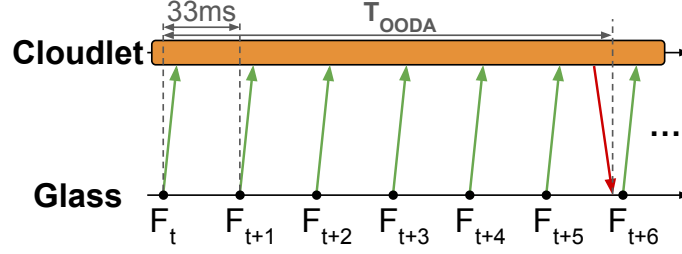


Figure 6: Case Study 2: Wingman System Pipeline

7.1 Mapping to WMA model

This case study can be mapped to the WMA model from §3.3. A *prospective mishap situation* occurs when the user’s action starts to diverge from a correct step in the ground truth procedure. If the user is unable to realize and correct the mistake, this will result in a *mishap* at some future time, depending on the task. For example, a mistake of cutting the wrong wire results in an immediate mishap, while using the wrong screw may result in the subassembly falling apart in a much later assembly step. With the help of Wingman, if the error state is detected by Wingman’s OODA loop and an alert is delivered to the user to stop the current action before the task- and mistake-specific deadline, the mishap can be successfully avoided. If Wingman fails to detect the erroneous step or if the alert is issued too late, the avoidance would fail, leading to a *mishap pass-through*. If Wingman wrongly recognizes a correct step as a user mistake and sends an unnecessary alert, this would be a *false alarm*.

7.2 Experimental Setup

To ensure experimental reproducibility, we use pre-recorded videos from the publicly-available Assembly101 dataset of labeled assembly tasks [30]. The dataset includes multi-view recordings of assembly and disassembly procedures of toy vehicles, with step-level annotations that make it suitable for step recognition and procedural error analysis. We manually create a training split that covers all classes, and a test split of 28 videos that each starts with a correct step and ends with a mistake step. The average duration of a test video is 63 s, and the average duration of a mistake step is 11.2 s.

The Wingman wearable device is a RayNeo X2 smartglass that is WiFi-connected to a cloudlet consisting of a Lambda Vector AI workstation with dual NVIDIA RTX 4090 GPUs. Cloudlet AI consists of the E2E-LOAD model [26], with a mean average precision (mAP) of 35.8 over all dataset classes. Processing is in chunks of 6 frames using the MViT backbone [31], achieving 5 predictions per second at 30 FPS (Figure 6). Video playback is at 1X speed. None of the test clips involve self-correction — all detected hazards are imminent mishaps. For simplicity, we use 3 different fixed deadlines.

7.3 Results

7.3.1 Wingman OODA Loop Latency

From Figure 6, Wingman’s OODA loop latency includes: (a) the time for a chunk of 6 frames spaced 33 ms apart to be transmitted to the cloudlet; (b) the model inference and decision time

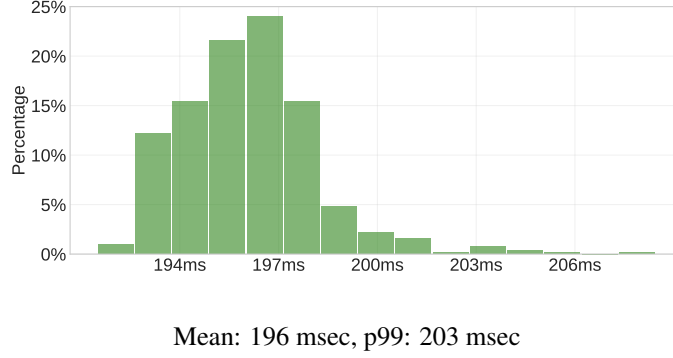


Figure 7: Case Study 2: Wingman OODA-Loop Latency

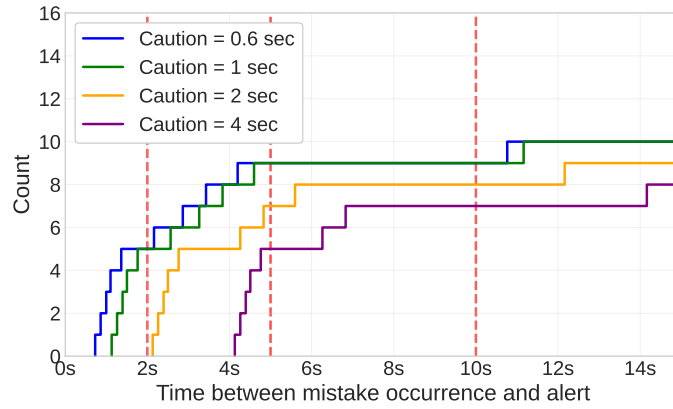


Figure 8: Cumulative Distribution of WPR

on the cloudlet for that chunk; and (c) the time for the decision to be transmitted back. Our measurements from 500 runs show that, after startup transients, the inference time on a 6-frame chunk is stable with a mean of 20.9 ms and a p99 value of 22.1 ms. The resulting OODA loop latency (Figure 7) has an average of 196 ms, with a p99 value of 203.3 ms. As this is much smaller than the average duration of 11.2 s for a mistake step, there is headroom for Wingman to wait for more observations before making an alert decision. In a system with a live user, there would be ample time for the user to self-correct before Wingman issues an alert.

7.3.2 Wingman Prevention Rate

To compensate for noisy input, Wingman waits for multiple consecutive concurring step detection results before accepting the verdict. We refer to the required number or duration of identical detections as Wingman’s *caution window*. Large caution windows reduce noise, but also yield slower detections. For example, since the OAD model produces output at 5 predictions per second, a caution window of 3 predictions maps to 600 ms.

With caution window as a tunable parameter, Figure 8 characterizes the WPR for Case Study 2. The results are presented as a CDF of the number of successes — 100% detection would mean that all 28 videos with a mistake result result in a timely alert. As shown in Figure 8(a), WPR changes as we choose different values for a mishap deadline. If we set the deadline for mishap prevention as 2 seconds after the start of a mistake step, Wingman can prevent 5 out of 28 mishaps when the caution is set to 0.6 seconds or 1 second. When the caution window is longer than 2 seconds,

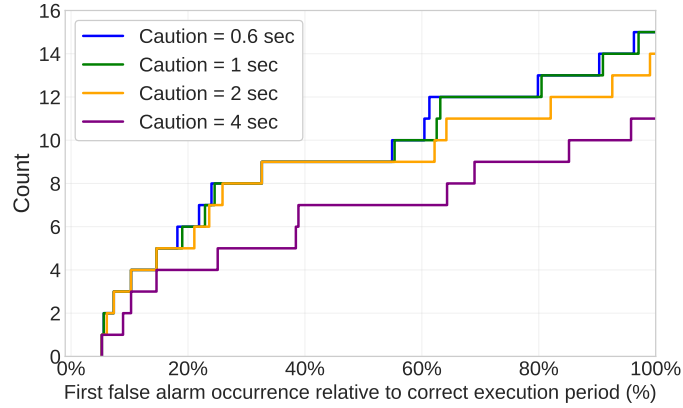


Figure 9: Cumulative Distribution of WFAR

avoidance always fails. If the deadline is defined as 5 seconds after mistake occurrence, 9 out of 28 mishaps can be avoided with a caution window below 1 second; 7 can be avoided with a caution window of 2 seconds; and 5 with a caution window of 4 seconds. The same trend continues when we further relax the deadline to 10 seconds.

7.3.3 Wingman False Alarm Rate

Figure 9 shows the number of false alarms from Wingman during correct procedures and their relative times of occurrence. With a caution window shorter than 2 seconds, 9 out of 28 test cases see a false alarm during the first half of the correct execution period, and the total number of false alarms increases to 14 to 15 by the end. With a 4-second caution window, there are fewer false alarm cases: Wingman generates a false alarm in 11 out of 28 test cases while successfully recognizing the correct steps in the remaining 17 test cases. Clearly, greater caution window leads to lower WFAR.

7.3.4 Discussion

These results show that a short caution window enables a quick detection of assembly mistakes and an early alert. This increases the chance of mishap avoidance, but also leads to more false alarms. In contrast, a long caution window means more conservative step recognition and alert decision, reducing the false alarm rate. However, this comes at a price of slower mistake detection and a higher risk of missing a mishap avoidance deadline.

Using reasoning similar to Case Study 1 (§6.3.4) a practical policy is to prioritize mishap avoidance while keeping the ratio between false alarms and unprevented mishaps below a target value. For example, with a mishap deadline of 10 seconds after a mistake step begins, a caution window of 2 seconds would achieve a false-alarm-to-mishap ratio below 0.7.

8 Case Study 3: Drone Flyover Mishaps

Unmanned aerial vehicles, commonly referred to as drones, are important tools for building inspection and surveys. Typically, they are controlled by a human pilot and assisted by several human visual observers who monitor the aircraft. By law, large drones must avoid flying over people and

vehicles to minimize damage in the event of a crash [32]. One of the main tasks of a visual observer is to alert the pilot of a potential pedestrian flyover.

As drone operations expand in scope, the requirement for several people to operate a single aircraft is not scalable. Wingman fits into the visual observer role well. If the drone strays too close to a pedestrian, Wingman can alert the pilot before a flyover occurs. As Wingman can use data streams (e.g. drone telemetry) not available to humans, it may provide super-human accuracy at such position estimation tasks.

In this case study, we set up Wingman in a drone visual observer scenario where it warns the pilot to avoid flyover of nearby pedestrians. To do this, Wingman must track the drone’s position via its telemetry while simultaneously watching the scene for people. Once a person is detected, Wingman establishes their position by extrapolating the detection from the camera’s known pose. If the drone and person seem to be on intersecting paths, Wingman will consider this a prospective mishap situation and intervene if it assesses that a flyover will happen soon.

8.1 Mapping to WMA model

This case study can be mapped to §3.3 as follows. A *prospective mishap situation* occurs when the drone pilot flies close enough to a pedestrian such that their current velocity will result in a flyover. If this is correctly detected by Wingman’s OODA loop and an alert is delivered to the pilot to stop the drone before the deadline, the mishap can be successfully avoided. If Wingman fails to detect a possible flyover or if the alert is issued too late, the drone will fly over the person, which is classified as a *mishap pass-through*.

Unlike §6 and §7, the legacy system (human pilot) is almost always unaware of danger. As shown in Figure 10(a), the pilot often must make a trade-off; either angle the camera down to check for flyover risk, or angle it straight ahead to fly the drone, removing ground objects from the camera field of view. With limited flight time, pilots usually defer flyover detection to a visual observer, which in this case is Wingman. As a result, Wingman should not expect corrective action to be taken by the pilot without intervention.

8.2 Experimental Setup

We implement and test flyover prevention with Wingman on real drone flight using the Parrot Anafi Gov, a popular drone used by several international government agencies [33]. We first measure the reactivity of the legacy system (pilot+drone). An experienced pilot with several dozen flight hours¹ flies the drone in a straight line at three speeds, 3 m/s (slow), 5 m/s (fast), and 6 m/s (very fast). At a random time, a human observer yells out a ”Stop!” warning at which point the pilot immediately stops the aircraft. By recording the warning time, telemetry, and video of each trial, we determine the warning-to-actuation time for the drone, and measure the distance traveled post-warning, which we define as *stopping distance*. We measure the latency of Wingman’s critical path, and estimate warning thresholds based on the p99 results of legacy reaction time. As the drone speed increases, the stopping distance will increase, so Wingman must adapt its alert decision point accordingly.

To determine the efficacy of Wingman, we first collect videos of flights in the experimental setup shown in Figure 10(b). A pilot flies the drone over a sequence of cones forming a rhombus

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

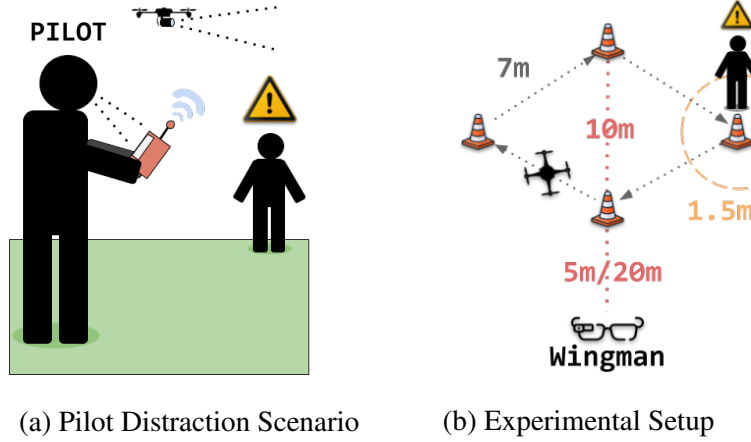


Figure 10: Case Study 3 Scenario and Experiment

with 10 m diagonal. A pedestrian is positioned on the flight path so that the drone passes within a 1.5 m *lateral separation* (distance between ground projected position of the drone and the object on the ground). A lateral separation of 1.5 or less is considered a mishap. Video is captured from 5 m and 20 m away, which simulates a close and far visual observer, along with drone telemetry.

This prerecorded data is then replayed at original speed to Wingman, running on the RayNeo X2 smart glass, which visually detects, tracks, and maps pedestrian positions, and determines if a flyover mishap is imminent. Our experiments estimate, assuming p99 legacy system reaction times, how often the drone could have stopped in time after Wingman alerts. This particular experiment does not consider false alarms, but we will study this in the future.

8.3 Results

8.3.1 Reaction Time of Legacy System to Warnings

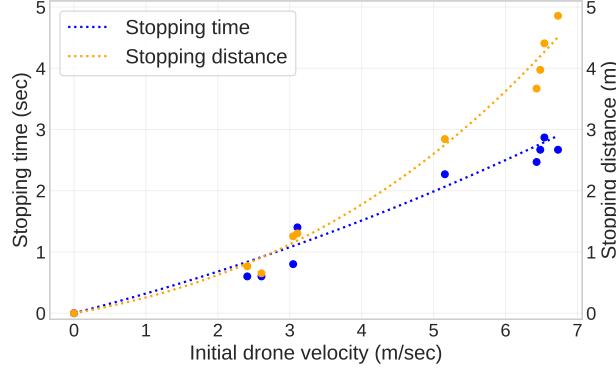
To adequately characterize the reactivity of the legacy system, two measurements are needed. The first is the reaction time of the pilot to the warning, which we label t_X . This includes hearing the warning, actuating the controller joystick to zero, and signal propagation to the drone. We measure this by taking the difference between the warning time and the time at which the drone starts slowing down. The second measure is the time it takes for the drone to slow down to a stop, which we label t_Y . The total reaction time t_R is the sum of t_X and t_Y , which corresponds to $t_C - t_B$ in Figure 2(b).

Mean (ms)	σ (ms)	Samples
297	69	11

The time it takes the pilot to react to a warning (t_X). Includes human reaction and controller actuation, plus signal propagation from controller to drone.

Figure 11: Human Pilot Reaction Time

As shown in Figure 11, t_X is on average 297 ms. This is expected since human auditory reaction time is around 200 ms [34] and actuation of the controller and drone add additional latency. t_Y is



Data points are represented by dots, line of best fit by dotted line.

Figure 12: Drone Stopping Speed and Distance by Velocity

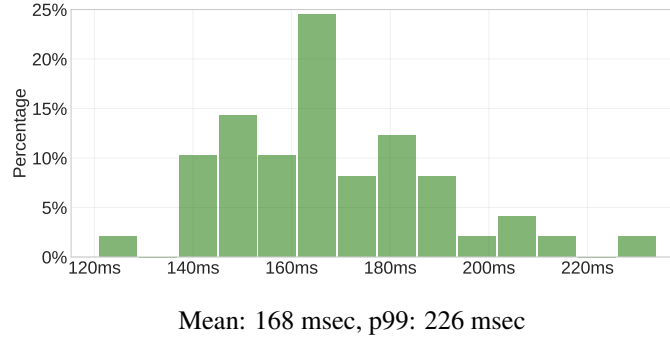


Figure 13: Case Study 3: Wingman OODA-Loop Latency

more complicated to measure because it is speed dependent. Figure 12 shows the relationship between the starting velocity, the time to stop, and the distance covered before stopping. From this data, we can estimate a worst case t_Y of 2.9 s at the highest velocity (6 m/s) and a best case t_Y of 0.6 s at the lowest velocity (3 m/s). This gives us a final $t_R = 3.2$ s at 6 m/s, and $t_R = 0.9$ s at 3 m/s.

8.3.2 Wingman OODA Loop Latency

The main component of Wingman's OODA loop is the processing of video for pedestrian tracking and drone telemetry for drone tracking. Wingman ingests video from the RayNeo X2 at 30 fps and either runs object detection onboard or forwards the frames to the cloudlet for processing. Previous research has shown that the end-to-end loop for offloaded processing of a YOLO11n model is on average 10.5 ms while the end-to-end loop for onboard processing of the same model is around 53.9 ms [8]. Drone telemetry is transmitted at 15 samples/s to the RayNeo X2 over Wi-Fi. The transmission time, which would require accurate time synchronization over networks to measure accurately, is approximately 4 ms. Wingman post-processing, which uses a Kalman filter to reduce position noise from telemetry and person detection, takes less than 1 ms.

As in Case Study 1 (§6), the processing of object detection is not in the critical path, as the pedestrian is detected and tracked long before a potential flyover. Therefore, Wingman's OODA loop time includes: (a) telemetry/camera frame receipt; (b) updating the tracked positions of the

Distance	3 m/s	5 m/s	6 m/s
5 m	+0.11 m	+0.17 m	+0.00 m
20 m	+0.12 m	+0.00 m	+0.00 m

(a) No Latency Added

Added Latency	3 m/s	5 m/s	6 m/s
30 ms	+0.06 m	+0.10 m	-0.10 m
50 ms	+0.06 m	+0.10 m	-0.10 m
100 ms	+0.06 m	-0.27 m	-0.50 m

(b) Latency Added, Wingman at 5 m Distance

Added Latency	3 m/s	5 m/s	6 m/s
30 ms	+0.07 m	+0.00 m	-0.08 m
50 ms	+0.07 m	+0.00 m	-0.14 m
100 ms	+0.07 m	-0.15 m	-0.38 m

(c) Latency Added, Wingman at 20 m Distance

Positive values are the distance buffer between the drone and the 1.5 m bubble after it stops. Negative values are the flyover distance the drone flew into the 1.5 m bubble.

Figure 14: Wingman Prevention Experiment

drone and the pedestrian with Kalman filtering; and (c) sending alert via RayNeo X2's audio speaker. The result in Figure 13 shows an average OODA loop latency of 168 ms, with a p99 value of 226 ms. This OODA loop makes timely mishap avoidance possible by leaving ample time for the legacy system to react.

8.3.3 Wingman Prevention Rate

A faster drone velocity results in a longer stopping distance, S , which in turn makes the latest possible t_A (the time at which an alert can be instantiated to prevent the mishap) earlier. Thus, Wingman must adapt its warning policy based on the drone's velocity. By estimating S , Wingman can decide when to warn the pilot of a potential flyover. As shown in Figure 12, the stopping distance of the drone grows non-linearly with respect to speed. We can fit this data with a polynomial function, shown with the dotted line, using the following formula where S is the stopping distance:

$$S = \alpha v^3 + \beta v^2 + \gamma v$$

for some tuned constants α, β, γ . The values of $\alpha = 0.0035, \beta = 0.0450, \gamma = 0.2100$ are the best fit for the Parrot Anafi Gov.

Using the experimental setup from Figure 10(b), we tested Wingman's warning system at the Anafi's three speed settings (3, 5, 6 m/s). Figure 14(a) shows the results of the experiment. The reported numbers are the distance from where the Anafi stopped to the edge of the 1.5 m flyover bubble around the pedestrian. A positive number indicates a successful stop that many meters from the bubble. A negative number indicates an unsuccessful stop that many meters within the bubble. The results show that Wingman is effective in preventing flyover at all speeds and at both distances. The quality fall-off at 20 m range is likely due to higher error for pedestrian location estimation.

In offload-only or offload-assisted mode, Wingman will stream video frames to the cloudlet for inference. Its network connection may be pristine (24-30 ms) or degraded (50-100 ms) depending on ambient conditions. To model the effect of this latency, we re-ran the flyover experiment with artificial added latency. Figures 14(b) and 14(c) show the results. At low speeds, latency does not affect performance at all. But at higher speeds, increased latency has a noticeable effect, with 6 m/s speed and 100 ms latency producing consistent failures at all values. These findings are consistent with those of similar live-flight drone perception experiments [14], and suggest an onboard-only approach for high speed flights.

9 Discussion and Future Research

Domain of Applicability: Our formulation of WMA as a means to augment a legacy system with an independent proactive monitor enables broad applicability. The legacy system may be human or cyber-physical. Relative to a human legacy system, Wingman is not subject to lapses of attention or fatigue. The independently running Wingman may leverage sensing data or perspectives that are unavailable to the legacy system, as demonstrated by Case Study 3. WMA is useful whenever the OODA loop of the legacy system leaves open chances of mishap that the independent Wingman may be able to detect and help correct.

Proactive vs. Reactive Operation: In contrast to other efforts in building assistive technologies (such as WCA), WMA requires proactively determining whether a mishap is imminent. This demands very different algorithms from those detecting after-the-fact if a mishap occurred (e.g., detecting a broken egg vs. predicting the egg is likely to break). Many of the huge gains in AI perception have been in detecting or understanding what currently is, not what will/may be. How such technology can be leveraged to predict future states remains an engineering and research challenge. Do new AI models or approaches need to be developed for the prediction of future physical states? Furthermore, from a system engineering perspective, the always-on nature of looking ahead also brings challenges in building a physical Wingman capable of extended, continuous use.

Adaptive Offload: Edge and cloud offload are now natural means of extending the capabilities of small, wearable devices. However, such offload resources may not always be available to a WMA system. In mobile settings (such as WMA for a drone or a hiker in the woods), ready access to offload resources may be fleeting and highly variable. The design of WMA applications that can cope with the variable availability of external resources, as well as the resulting variation in quality and latency of prospective mishap detection is a future challenge in this space. Whether WMA capabilities are even possible without edge or cloud offload may be application-specific.

Energy Considerations: The simplified model of a Wingman application presented here has a continuously running AI pipeline. This is clearly not practical in battery-powered wearable devices. It would be preferable to conserve precious mobile-device energy by running Wingman only when mishaps are likely. For example, a Wingman application that helps elderly users avoid tripping and falling mishaps only needs to be active while the user is walking. Low-energy mechanisms, such as occasional sampling of an accelerometer, may be used to determine when the user is moving or still, activating the Wingman application when movement is detected. This is, of course, a very application-specific solution. How might this be generalized to a broader set of applications? Do such approaches to reducing the battery consumption of WMA even exist for most applications? Finally, though we focused primarily on wearable device energy, given the increasing concern over the energy and resource use of large AI models, how much energy or carbon footprint may widespread WMA incur? Do WMA's societal gains overcome its costs?

Tolerance of Alerts: In many domains, earlier is always better. In WMA for human users, this is not necessarily the case. Early detection of a prospective mishap leaves greater opportunity for successful corrective action. However, alerting humans when they are already aware of the potential mishap, deem the alert to be a false positive, or are not yet ready to act, will only serve to annoy them and may lead to inattention to the warnings. The delicate balance between the

timeliness, frequency, and even necessity of alerts is critical to WMA success. How these goals can be balanced in a general way remains an open issue.

Online Learning from Feedback: The concept of online learning may be applied to Wingman to continuously improve its runtime KPIs. Based on a history of mishap avoidance outcomes, can Wingman’s corrective AI learn the characteristics of the legacy system and adapt to its self-correction speed to avoid unnecessary alerts? Perhaps it could also collect data to retrain AI models online to improve the accuracy of prospective mishap detection. Does such learning need to be supervised based on user feedback, or can automated unsupervised mechanisms be effective? Of course, such adaptation to specific users or situations may be susceptible to overfitting, so application-specific policies and limits on adaptation may be necessary for robustness.

Edge System Enhancements: WMA presents a type of problem that poses unique challenges to existing edge computing systems: a real-time predictive AI, a tight latency constraint that arises from two concurrent OODA loops, and a task-specific upper bound for the false alarm rate. Furthermore, there may be numerous concurrent WMA application instances vying for timely edge execution. Current edge services are largely best effort, and are low-latency only by virtue of short network distance. For reliable and robust WMA support, edge infrastructure may need to provide some notion of real-time services or statistical latency bounds on execution to meet OODA-loop requirements. A principled way of allocating limited edge resources to competing WMA demands to maximize value and fairness will require further study.

10 Conclusion

This paper introduced a general formulation for WMA and applied it to three scenarios. Across the three test cases, common patterns emerge that point to the conditions for success and identify common metrics of interest. In general, WMA will be called for when the OODA loop of the legacy system leaves open possibilities for error due to the stochastic nature of the environment and the latency required for corrective response. Wingman, our mishap avoidance assistant, will successfully intervene to prevent accidents when its own OODA loop of sensing, detecting the error signal, and transmitting the corrective instruction accommodates the situational demands. Our case studies varied in what leads to the prospective mishap, e.g., human inattention or an intruder into the path of a drone. Mishap avoidance is imperfect, leading to failed attempts at avoidance and false alarms. How any Wingman system operates will depend on the relative costs of these unwanted outcomes and how they are balanced. Taken together, the three test cases clearly establish the feasibility of WMA given current infrastructure devices, edge, and AI. Future work will consider how to extend this capability and further develop the WMA concept in terms of optimal policy guidelines and real-world evaluation metrics.

References

- [1] Center for Disease Control, “Older Adult Falls Data.” (<https://www.cdc.gov/falls/data-research/index.html>). Last accessed on July 4, 2025.
- [2] “The True Cost of Downtime from Human Error in Manufacturing.” (<https://www.rewo.io/the-true-cost-of-downtime-from-human-error-in-manufacturing/>). Last accessed on July 4, 2025.
- [3] Siemens, “The True Cost of Downtime 2024.” (<http://siemens.com/senseye-predictive-maintenance>). Last accessed on July 4, 2025.
- [4] J. Leland, P. Baskar, O. Bensimon, A. Southall, C. Killganon, and J. Ransom, “The Air Canada Crash: Before and After the Frantic Call to ‘Stop, Stop, Stop’,” *New York Times*, March 29 2026.
- [5] M. Satyanarayanan, “Hot Topics: Mobile Computing,” *IEEE Computer*, vol. 26, no. 9, pp. 81–82, 1993.
- [6] M. Satyanarayanan, N. Beckmann, G. A. Lewis, and B. Lucia, “The Role of Edge Offload for Hardware-Accelerated Mobile Devices,” in *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile 2021)*, 2021.
- [7] Q. Dong, J. Xu, P. Pillai, R. Klatzky, and M. Satyanarayanan, “Real-Time Fusion of Device, Cloudlet & Cloud AI,” in *Proceedings of the 9th International Workshop on Edge Systems, Analytics, and Networking (EdgeSys 2026)*, (Cambridge, UK), June 2026.
- [8] Q. Dong, J. Xu, P. Pillai, and M. Satyanarayanan, “Does Accurate Real-Time AI Need Edge Offload?,” in *Proceedings of the IEEE/ACM Symposium on Edge Computing*, 2025.
- [9] National Highway Transportation Safety Administration, “Driver Assistance Technologies.” (<https://www.nhtsa.gov/vehicle-safety/driver-assistance-technologies>). Last accessed on July 5, 2025.
- [10] R. B. Naumann, L. K. Kreuger, L. Sandt, K. H. Lich, B. Bauchwitz, W. Kumfer, and T. Combs, “Examining the Safety Benefits of Partial Vehicle Automation Technologies in an Uncertain Future,” Tech. Rep. AAA Foundation for Traffic Safety, August 2023.
- [11] 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.
- [12] 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.
- [13] L. M. Blaha, “Interactive OODA processes for operational joint human-machine intelligence,” in *NATO IST-160 Specialist’s Meeting: Big Data and Military Decision Making*, pp. 3–1, NATO, 2018.

- [14] M. Bala, T. Eiszler, X. Chen, J. Harkes, J. Blakley, P. Pillai, and M. Satyanarayanan, “The OODA Loop of Cloudlet-based Autonomous Drones,” in *2024 IEEE/ACM Symposium on Edge Computing (SEC)*, (Rome, Italy), 2024.
- [15] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, “Towards Wearable Cognitive Assistance,” in *Proceedings of the Twelfth International Conference on Mobile Systems, Applications, and Services*, (Bretton Woods, NH), June 2014.
- [16] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, D. Siewiorek, and M. Satyanarayanan, “An Empirical Study of Latency in an Emerging Class of Edge Computing Applications for Wearable Cognitive Assistance,” in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, (Fremont, CA), October 2017.
- [17] M. Satyanarayanan and N. Davies, “Augmenting Cognition through Edge Computing,” *IEEE Computer*, vol. 52, July 2019.
- [18] M. O. J. O. Munoz, R. Klatzky, J. Wang, P. Pillai, M. Satyanarayanan, and J. Gross, “Impact of delayed response on Wearable Cognitive Assistance,” *PLoS ONE*, vol. 16, no. 3, 2021.
- [19] R. Iyengar, E. Zhang, and M. Satyanarayanan, “Experience with Using Synthetic Training Images for Wearable Cognitive Assistance,” in *Proceedings of the AAAI Spring Symposium on AI Engineering*, March 2022.
- [20] R. L. Klatzky and M. Satyanarayanan, “Psychological Science Meets Wearable Cognitive Assistance,” *Current Directions in Psychological Science*, August 2023.
- [21] R. Iyengar, Q. Dong, C. Nguyen, P. Pillai, and M. Satyanarayanan, “Offload Shaping for Wearable Cognitive Assistance,” *Electronics (Special Issue AI for Edge Computing)*, vol. 13, no. 20, 2024. (<https://doi.org/10.3390/electronics13204083>).
- [22] J. P. Bliss, R. D. Gilson, and J. E. Deaton, “Human probability matching behaviour in response to alarms of varying reliability,” *Ergonomics*, vol. 38, no. 11, pp. 2300–2312, 1995. PMID: 7498189.
- [23] J. P. Bliss and M. C. Dunn, “Behavioural implications of alarm mistrust as a function of task workload,” *Ergonomics*, vol. 43, no. 9, pp. 1283–1300, 2000. PMID: 11014752.
- [24] X. Wang, S. Zhang, Z. Qing, Y. Shao, Z. Zuo, C. Gao, and N. Sang, “Oadtr: Online action detection with transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7565–7575, 2021.
- [25] J. An, H. Kang, S. H. Han, M.-H. Yang, and S. J. Kim, “Miniroad: Minimal rnn framework for online action detection,” in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10307–10316, 2023.
- [26] S. Cao, W. Luo, B. Wang, W. Zhang, and L. Ma, “E2e-load: End-to-end long-form online action detection,” *arXiv preprint arXiv:2306.07703*, 2023.

- [27] A. Flaborea, G. M. D. di Melendugno, L. Plini, L. Scofano, E. De Matteis, A. Furnari, G. M. Farinella, and F. Galasso, “Prego: Online mistake detection in procedural egocentric videos,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18483–18492, June 2024.
- [28] L. Plini, L. Scofano, E. De Matteis, G. M. D. di Melendugno, A. Flaborea, A. Sanchietti, G. M. Farinella, F. Galasso, and A. Furnari, “Ti-prego: Chain of thought and in-context learning for online mistake detection in procedural egocentric videos,” *Computer Vision and Image Understanding*, vol. 264, p. 104613, 2026.
- [29] L. Seminara, G. M. Farinella, and A. Furnari, “Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos,” in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [30] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao, “Assembly101: A large-scale multi-view video dataset for understanding procedural activities,” *CVPR 2022*.
- [31] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, “Multi-scale vision transformers,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6804–6815, 2021.
- [32] “Operations over people general overview federal aviation administration.” (https://www.faa.gov/uas/commercial_operators/operations_over_people), Nov 2022. Last accessed on May 7, 2026.
- [33] “Parrot anafi usa gov.” (<https://www.parrot.com/en/drones/anafi-usa/proven-and-trusted>). Last accessed on May 7, 2026.
- [34] A. Jain, R. Bansal, A. Kumar, and K. D. Singh, “A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students,” *Int. J. Appl. Basic Med. Res.*, vol. 5, pp. 124–127, May 2015.