Holistic Energy Awareness and Robustness for Intelligent Drones

RAVI RAJ SAXENA and JOYDEEP PAL, Indian Institute of Science, Bengaluru, India
SRINIVASAN IYENGAR, Microsoft Research India, Bengaluru, India
BHAWANA CHHAGLANI, University of Massachusetts Amherst, Amherst, USA
ANURAG GHOSH, Carnegie Mellon University, Pittsburgh, USA
VENKATA N. PADMANABHAN, Microsoft Research India, Bengaluru, India
PRABHAKAR T. VENKATA, Indian Institute of Science, Bangalore, India

Drones represent a significant technological shift at the convergence of on-demand cyber-physical systems and edge intelligence. However, realizing their full potential necessitates managing the limited energy resources carefully. Prior work looks at factors such as battery characteristics, intelligent edge sensing considerations, planning, and robustness in isolation. But a global view of energy awareness that considers these factors and looks at various tradeoffs is essential. To this end, we present results from our detailed empirical study of battery charge-discharge characteristics and the impact of altitude and lighting on edge inference accuracy. Our energy models, derived from these observations, predict energy usage while performing various manoeuvres with an error of 5.6%, a 2.5X improvement over the state-of-the-art. Furthermore, we propose a holistic energy-aware multi-drone scheduling system that decreases the energy consumed by 21.14% and the mission times by 46.91% over state-of-the-art baselines. To achieve system robustness in the event of link or drone failure, we observe trends in Packet Delivery Ratio to propose a methodology to establish reliable communication between nodes. We release an open-source implementation of our system. Finally, we tie all of these pieces together using a people-counting case study.

CCS Concepts: • **Computer systems organization** → **Sensor networks**; • **Hardware** → **Robustness**; • **Computing methodologies** → *Supervised learning*; • **Networks** → *Programmable networks;*

Additional Key Words and Phrases: Energy-awareness, scheduling, low-cost sensing, drones, UAVs, Battery Model, planning and surveying, robustness

ACM Reference Format:
Ravi Raj Saxena, Joydeep Pal, Srinivasan Iyengar, Bhawana Chhaglani, Anurag Ghosh, Venkata N. Padmanabhan, and Prabhakar T. Venkata. 2024. Holistic Energy Awareness and Robustness for Intelligent Drones. *ACM Trans. Sensor Netw.* 20, 3, Article 57 (March 2024), 31 pages. https://doi.org/10.1145/3641855

R. R. Saxena and J. Pal contributed equally to this research.
B. Chhaglani and A. Ghosh work performed while the author was at Microsoft Research India.
Authors' addresses: R. R. Saxena, J. Pal and, P. T. Venkata, Indian Institute of Science, CV Raman Rd, Bengaluru, Karnataka, 560012, India; e-mails: {ravisaxena, joydeeppal, tvprabs}@iisc.ac.in; S. Iyengar and V. N. Padmanabhan, Microsoft Research India, Lavelle Road, Ashok Nagar, Bengaluru, Karnataka, 560001, India; e-mails: {sriyengar, padmanab}@microsoft.com; B. Chhaglani, University of Massachusetts, Amherst, 37 Mather Drive, Amherst, MA 01003, USA; e-mail: bchhaglani@cs.umass.edu; A.Ghosh, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, United States; e-mail: anuraggh@andrew.cmu.edu.

ACM 1550-4859/2024/03-ART57
https://doi.org/10.1145/3641855

1 INTRODUCTION

Unmanned Aerial Vehicles (**UAVs**), popularly known as drones, represent a significant technological shift, at the convergence of on-demand cyber-physical systems and edge intelligence. They have shown tremendous promise in collecting data that are either dangerous, expensive, or impractical to obtain otherwise [Sachs 2017]. Drones have provided disruptive solutions for monitoring natural and built environments in varied applications such as agriculture [Vasisht et al. 2017], renewables [Li et al. 2020], wildlife conservation [Schiffman 2014], search and rescue [Silvagni et al. 2017], and so on. Drone-based applications are taking off because of the availability of open-source hardware and software platforms [Meier et al. 2015; Zurich 2013] at an affordable cost [DJI 2020], and regulatory clarity in many countries [Administration 2016; Agency 2019].

As drones are mostly battery-powered, managing their energy resources is critical, whether we consider an individual drone in isolation or a fleet of drones deployed for a task. Increased battery life leads to longer flight times, enabling more extensive coverage with the available drones, and a higher uptime with fewer charging pitstops. To highlight the significance of energy constraints on drones, consider a disaster recovery scenario illustrated in Figure 1 that shows a flood-affected area along the banks of a river. The two drones deployed—shown as blue and red crosses—fly along pre-determined routes, with the mission of identifying possible human subjects in harm's way. The aerial picture shows the objects detected by the blue drone using an on-board camera. Ensuring that the area of interest is covered quickly and effectively by the drones available would require careful management of their battery energy resource, which, in turn, requires overcoming several challenges.

First, charging should ideally happen autonomously (e.g., with a wireless charging pad [Khonji et al. 2017]), so that the drones can remain in a perpetual cycle of getting charged, taking off for their mission, and then landing to get charged again, without any human involvement. There is the question of the level to which to charge the drone batteries. To the extent that the rate of charging is a concave function of time, it might be beneficial to charge partially and free up the charging station for another drone to use. Moreover, in a setting with a fleet of drones—a common case for large-scale sensing tasks—there might be heterogeneity in the capacity, chemistry, and age of batteries. So, we need to create custom charging and discharging profiles for each drone battery. We also require an appropriate location of charging stations for recharging the drones' batteries for such large-scale sensing tasks.

Second, while most applications to date have used the drone as a "dumb" sensing device [Mahmoud et al. 2015; Mottola et al. 2014; Van't Hof and Nieh 2019], there are many situations where it is advantageous to have on-board intelligence on the drone for local sensor stream processing (E.g., bandwidth constraints and privacy concerns). Energy again plays a non-trivial role in edge computation i.e., the interplay between battery energy and the accuracy of computation with varying weather and lighting conditions (see Figure 2). For instance, in our disaster recovery scenario, drones flying at a greater height could capture a larger area using their cameras, thereby enabling greater coverage for a given amount of battery drain. However, the detection accuracy for the objects of interest (E.g., people, vehicles) would likely suffer due to the reduced size of the objects in the captured images (E.g., see Figure 1). Flying the drones at a lower height would likely improve the object detection accuracy but at the cost of reduced coverage and hence an increased survey time. Hence, there is a need to balance the energy cost and the task accuracy of each computing platform.

Finally, large-scale surveying tasks might need the services of multiple drones to complete in stipulated time. However, partitioning the task among a heterogeneous fleet of drones is non-trivial. Consider a scenario shown in Figure 3. Here, a surveying task entails assessing the ecological impact after creating a reservoir for a dam by counting the number and species of the

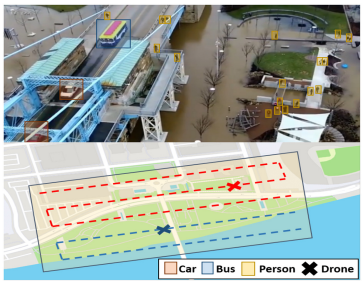


Fig. 1. Multiple drones surveying a given area for disaster recovery.

Fig. 2. Varying weather, lighting and altitude.

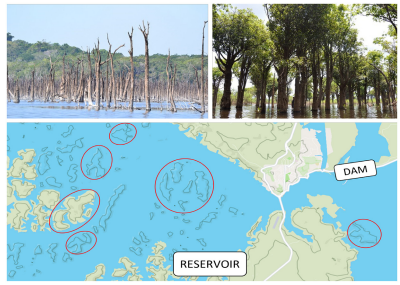


Fig. 3. Distributed irregular survey regions for ecological assessment of dam construction.

submerged trees—similar to a problem discussed in [Resende et al. 2020]. Due to the undulating bed of the reservoir lake, the partially submerged regions are distributed across a wide area. Thus, it is important to consider the location, shape, and square footage of these regions to divide the surveying task optimally across multiple drones available at one's disposal.

Prior work has looked at the effect of battery characteristics [Di Franco and Buttazzo 2016; Tseng et al. 2017], path planning and scheduling [Araújo et al. 2013; Huang 2001; Xie et al. 2019], and effect of variation in drone imagery [Du et al. 2018] on drone operation in isolation. However, a holistic view is needed to break the silos by considering tradeoffs between these factors to develop energy-aware drone scheduling. We address these challenges through the following contributions:

(1) **Drone energy modeling:** We present an extensive empirical study based on over 100 drone flights to arrive at several observations regarding the drone charging and discharging characteristics. Based on these, we perform a detailed and accurate modeling of battery performance. We report an error of just 1.94% in predicting the battery charge times. Furthermore, our novel predictor accurately predicts energy consumption for different drone manoeuvres, with an average error of 5.6%—a 2.5× reduction in error from the state-of-the-art [Tseng et al. 2017]. We also discuss the tradeoff between drone energy consumption and varying altitude, lighting conditions, and so on.

(2) **Energy-aware Scheduling:** Informed by our observations from the empirical study, we build a holistic system to create energy-aware flight paths for a given task. Our system generates energy-aware flight path for a given shape via a novel path planning algorithm and splits the task among a fleet of drones via an **Integer Linear Programming** (**ILP**) model. Thus, our system decreases the average energy consumption by 21.14% when the shape regularity is less than 0.85 (an overall improvement of 10.15%) and an average relative time reduction of 46.91% in a multiple drone setup, over the variety of baselines and the state-of-the-art. Our iterative greedy adjustment can also give further time reduction. We also discussed about impact of charging station placement and battery threshold to reach there.

(3) **Open-source implementation:** We create an open-source toolkit for charging and discharging energy models of drone. We also implement our system as an extension to the most widely used open-source mission planning software, qGroundControl [Zurich 2013]. Our implementation involved writing over 2000 lines of C++ code and 400 lines of QML scripts.

(4) **Robustness methodology:** We propose an algorithm to provide the system with information about link/node failure and to recover from such an event.

2 ENERGY CHARACTERIZATION AND MODELING

In this section, we discuss some interesting and non-intuitive patterns learned through an extensive empirical study, which included completing over 100 flights on drones with varying payload along with charging the batteries in varied settings (i.e., different initial charging current). We use a custom-built quadcopter of 1-meter diagonal length with 15-inch propellers and weighing around 900 grams without the battery. For the experiments presented in this article, we used an 8000 mAh LiPo battery. Moreover, we introduce accurate models for the time taken to charge the batteries and the energy consumption on various maneuvers.

2.1 Battery Charging Characterization

High discharge currents with high energy density, and flexibility in shape and size make **Lithium-Ion** (**Li-ion**) or **Lithium-Polymer** (**LiPo**) batteries ideal candidates for powering drones. Similar to any other battery chemistry, Li-based ones also have some peculiarities. Here, we highlight the effect of two main factors—(i) starting current, and (ii) age — on the LiPO battery used in our drones.

Impact of starting current: The initial battery charging happens at a constant current level (CC mode). After a while, the charging process reaches an inflection point beyond which the charging happens at a constant voltage level (CV mode). Higher currents allow faster charging, while at the same time, the battery capacity degrades more quickly over multiple cycles [Badam et al. 2015]. Over the entire empirical study, we experimented with different charging currents. Figure 4(a) illustrates the impact of the initial current on the charging time of the battery. The figure presents the change in the charging current, the battery voltage, and the cumulative energy delivered for the three different charging sessions. For this experiment, we considered the batteries of the same age, i.e., ones that had undergone the same number of charge-discharge cycles. Specifically, we looked at the time taken to deliver the last 140 Wh to charge the batteries fully.[1] The overall time taken with the initial current of 6.4A, 3.7A, and 1.8A is 72 minutes, 112 minutes, and 207 minutes, respectively. For applications that require higher drone uptime, with disregard for battery capacity degradation down the line, one must choose higher initial charging currents.

Observation. *Higher starting current leads to a commensurate reduction in charging times.*

Impact of Ageing: The energy density of Lithium-based batteries is affected by ageing [Badam et al. 2015]. Battery ageing is caused by cell oxidation—an irreversible process. Over multiple charge-discharge cycles, the usable capacity drops consistently. We observed the effects of ageing over 200 charging cycles spread across multiple batteries.

Figure 4(b) illustrates the impact of ageing on the time taken to deliver the final 140Wh of energy to charge two batteries fully. Both batteries are charged with the same initial charging current of 6.4A. **Battery 1** is a new battery with fewer than five charging cycles, whereas **Battery 2** had been put through over 130 charge-discharge cycles.As shown, the current and energy charging profiles are identical in the first 34 minutes, as both batteries are in the CC mode of operation. However, the voltage increase in **Battery 2** is much quicker as it reaches the inflection point faster. After reaching the inflection point, **Battery 2** continues to get charged in the CV mode at a much slower rate, as indicated by the flatter energy curve. **Battery 1** switches from CC to CV mode much later—around the 45-minute mark. To fully charge **Battery 1** and **Battery 2**, it takes 79 and 113 minutes, respectively. Thus, the time taken to fully charge batteries with the same quantum of energy depends on their respective age.

[1] As batteries degrade over time, their capacity reduces. In our setup, 140 Wh was the minimum energy needed to fully charge the oldest battery.

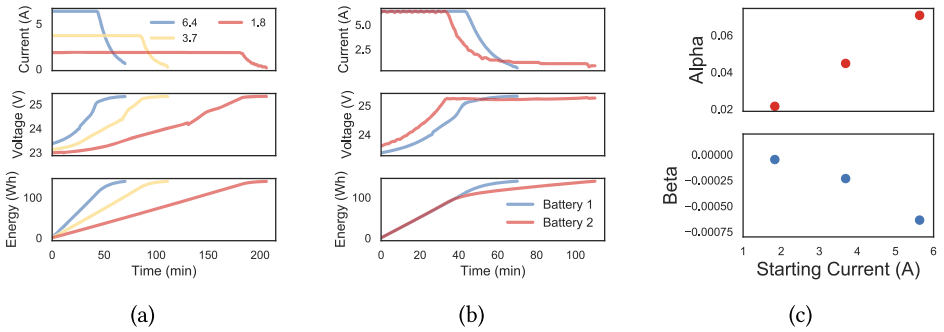


Fig. 4. (a) Impact of starting current, (b) Impact of aging on charging time, and (c) Charging parameters vs current.

Observation. *Older batteries reach the inflection point faster and take longer to charge a given amount of energy.*

2.2 Battery Charging Modeling

The charging process is non-linear (due to the CC and CV modes) and varies with charging current and battery age. Thus, there are opportunities for drone operators to create drone flight schedules that optimize drone uptime by knowing the charging profiles of various drone batteries in their fleet. Using the observations made through our empirical study, we now present our model to characterize the charging profiles accurately. Figure 4(a) shows how the current profile is flat until the inflection point and experiences an exponential reduction in the CV mode. On the other hand, the voltage profile follows a quadratic curve until it reaches the inflection point and stabilizes at a maximum value in the CV mode.

Let ρ represent the time elapsed since the start of charging until when the inflection point occurs. I^{start} is the starting current applied for charging, and the final maximum voltage attained in the CV mode is V^{max}, which is constant for a specific battery-charger pair. Let V_t and I_t represent the voltage and the current levels at time $t \in [0, T]$, where T is the total number of time intervals needed for charging. We model the current and the voltage profiles using a piece-wise curve fit. In the CC mode, when $\rho > t$, we have:

$$V_t = V_{max} - \alpha * (\rho - t) - \beta * (\rho - t)^2; \qquad I_t = I^{start}$$

α and β are the quadratic coefficients that fit the voltage curve. These parameters varies for different starting current, battery age, and so on. In the CV mode, when $\rho < t$, we have:

$$V_t = V^{max}; \qquad I_t = I^{start} * \gamma^{(t-\rho)}$$

Here, γ is the exponential parameter to fit the current curve. Once again, this parameter vary for different starting current, battery age, and so on. The above equations can be combined as

$$V_t = V^{max} - \alpha * (\rho - t)^+ - \beta * ((\rho - t)^+)^2$$
$$I_t = I^{start} * \gamma^{(t-\rho)^+}$$

For estimate the parameters such as ρ, α, β, and γ, we use a Bayesian Inference approach that provides probability density functions for each of these parameters. We describe the complete Bayesian formulation in Table 1. Here, we assume that the error associated with fitting the voltage and current profiles are normally distributed $\mathcal{N}(0, \sigma)$. Also, we use weakly informative prior for the different battery charging parameter based on the range of values relevant to the domain.

Table 1. Drone Charging Bayesian Model Formulation

Prior
$\rho \sim \mathcal{U}(0,T), \gamma \sim \mathcal{U}(0,1), \alpha \sim \mathcal{N}(0,2), \beta \sim \mathcal{N}(0,2),$
$\sigma^I \sim Cauchy(0,5), \quad \sigma^V \sim Cauchy(0,5)$
Regression Equation
$\mu_t^V = V^{max}$ - $\alpha * (\rho - t)^+ - \beta * ((\rho - t)^+)^2$
$\mu_t^I = I^{start} * \gamma^{(t-\rho)^+}$
Model Likelihood
$I_d \sim \mathcal{N}(\mu_t^I, \sigma^I), \quad V_d \sim \mathcal{N}(\mu_t^V, \sigma^V)$
Parameter Bounds
$0 < \rho < T, \quad 0 <= \gamma <= 1, \quad -1 <= \alpha, \quad \beta <= 1$

Using the **Markov chain Monte Carlo** (**MCMC**) method, we can utilize the *evidence* (i.e., known quantities such as V_t and $I_t \forall t \in T$) to get the *posterior* distributions for the charging model parameters such as α, β, γ, and ρ starting from the initial *prior* beliefs. The priors were set using the recommendations provided by Gelman et al. [Gelman et al. 2006].

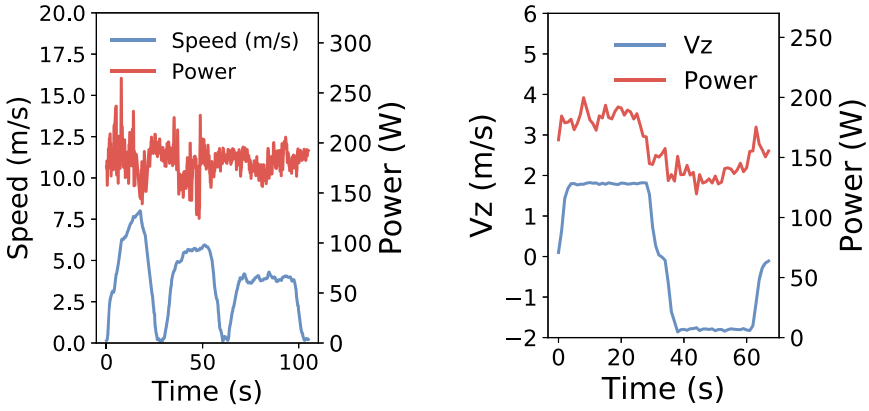
Based on the past charging sessions, we can learn the different charging parameters. From our empirical study, we found that the value of ρ, changes with the age of the battery. However, for same-aged batteries, ρ corresponds to reaching a specific battery energy level—irrespective of the starting current. From Figure 4(c), we observe that both α and β follow a linear relationship with the starting current, reflecting the fact that the higher the charging current, the more rapidly the voltage approaches V^{max} during the CC phase. This linear dependence helps in learning the parameters for a specific charging current by interpolating the parameter values for various other charging currents. On the other hand, the value of γ (≈ 0.91), which pertains to the CV phase, does not change with starting current for the battery. While all (except γ) these parameters change with battery age, the change is gradual. Thus, we can predict quite accurately the time taken to charge the battery.

2.3 Battery Discharging Characterization

Energy consumed to fly the drone depends on the current drawn by the motors to provide the lift and thrust needed to execute the various maneuvers. In turn, these depend on the prevalent environmental conditions (wind speed and direction), the weight of the payload carried, and so on. Here, we highlight the impact of two main factors—(i) the drone's speed, and (ii) weight of the payload—on the battery drain observed. This study involved recording energy consumption data over 100 flights involving a diverse set of maneuvers in different wind conditions and carrying different amounts of payload.

Impact of speed: While flying the drone at a higher speed will result in a shorter time to complete a mission, it is unclear if the energy consumed will also reduce. Thus, to illustrate the impact of speed, we show the results from two specific flight paths—(i) linear horizontal flights between two points approximately 130 meters apart at varying speeds, and (ii) vertical flights between the altitudes of 10 and 50 meters above the ground at 2 m/s.

Figure 5(a) shows the horizontal speed and the power consumed of the drone moving between two points while switching its maximum speed from 8 m/s to 6 m/s and finally to 4 m/s. Clearly, the maneuvers are completed in a shorter time at higher speeds. However, the power consumption does not increase proportional to the increase in speeds. Thus, the reduction in energy consumed at higher speeds is the result of a reduction in time without an equivalent



(a) Horizontal flights.

(b) Vertical flights

Fig. 5. Power usage for various drone maneuvers.

increase power. Figure 5(b) presents the relationship between vertical up and down movement and power consumption. Ascending higher is represented by a positive value of the vertical velocity component represented as V_z, whereas a negative value represents the drone descending toward the ground. As shown, working against gravity (while ascending) incurs a 30% higher power consumption on average compared with descending, assisted by gravity.

Observation. *At higher horizontal speeds, the energy usage is lower due to reduction in time without an equivalent increase power. Ascending incurs a 30% higher power compared to descending.*

Impact of weight: Drones often carry an additional payload for specific missions. Thus, it is essential to characterize the energy consumption when carrying different weights. To study the impact of weight in detail, we created two drone workloads, i.e., flight plans involving several maneuvers. We call the first workload *Horizontal Oblique Triangle*, in which the drone moves horizontally from the start point to the endpoint in a straight line separated by a distance of 240m at 10 m/s. After reaching the endpoint, the drone follows an oblique path upward to the center of this line but at a higher altitude of 20m at 5 m/s. Then, the drone returns to the start point by moving in an oblique path downward at 5m/s. This series of maneuvers forms a triangular flight path. The drone completes three such triangular flight paths for a given payload. This workload is completed in roughly 341 seconds. The second workload is called *Vertical Shift Hover*. Here, the drone ascends from a height of 10 to 30 meters in 10 meter increments. At each point, the drone hovers for 5 minutes before ascending. After hovering for 5 minutes at 30 meters height, the drone descends for landing. This workload takes approximately 920 seconds to complete. Table 2 summarizes our findings of executing these two workloads with different weights attached to the drone. We observe a linear relationship between the energy consumed and the drone weight.

Observation. *The increase in weight results in a linearly proportional increase in energy consumption.*

2.4 Battery Discharging Modeling

Having discussed the impact of speed and payload, we want to predict the instantaneous power consumed while performing various maneuvers. By integrating the predicted power values, we can easily calculate the energy spent on a given set of maneuvers. Below, we discuss our power prediction model in more detail.

Table 2. Impact of Weight on Power Consumption

Workload	Time (s)	Weight (kg)	Average Power (W)
Horizontal	341	1.895	170.01
Oblique		1.995	181.20
Triangle		2.095	190.87
Vertical	920	1.895	172.51
Shift		1.995	181.60
Hover		2.095	206.51

We can think of the power required to maintain a stable flight as a combination of several components. Let P_{xy}, and P_z be the power needed to sustain the flight in the horizontal and vertical direction, respectively. Similarly, let P_{wind} be the power demand to resist the deviation due to the wind. Finally, P_{drag} is the power necessary to move against drag. We know that Power = Force*Velocity = Mass*Acceleration*Velocity. Note that while hovering, drone's vertical acceleration (A_z) has to be equal and opposite to acceleration due to gravity, i.e., 9.8 m/s^2 near sea-level.

$$P_{xy} = m * A_{xy} * V_{xy}; P_z = m * A_z * V_z; P_{wind} = m * A_{xy} * V_{wind};$$

Here, m is the mass of the drone, A_{xy} and V_{xy} are acceleration and velocity in the horizontal direction. Similarly, A_z and V_z are acceleration and velocity in the vertical direction. V_{wind} denotes the velocity of the wind. Further, from drag equations [at National Aeronautics and Administration 2019], we know that P_{drag} is proportional to $V_{xy} \cdot (V_{wind} + V_{xy})^2$. Note that acceleration and velocities are vector quantities and have both magnitude and direction associated with them. One can find a closed-form expression linking the overall power consumption of the drone, given by P_{total}. However, these components may not be independent of each other. Thus, we choose **random forest** (**RF**) regression to learn a power prediction model using the four components of power as features to predict instantaneous power use. We choose RF regression out of **Support Vector Regression** (**SVR**) and **K Nearest Neighbour** (**KNN**) as RF gives the best performance, as shown in evaluation Section 7.

2.5 Energy-Aware Inference Characterization

We now turn to the inference performed on the drone, specifically for (aerial) vision-based tasks. We characterize the various tradeoffs in executing ML inference workloads, such as object detection on edge devices. We experiment with two edge platforms, a Raspberry Pi 4 with a Google Coral USB Accelerator (Coral+RPi4) and Nvidia Jetson Nano. To characterize the object detection performance, we fine-tune two **Single Shot Detectors** (**SSD**) [Liu et al. 2016], with MobileNetV1 [Howard et al. 2017] (MV1) and MobileNetV2 [Sandler et al. 2018] (MV2) backbone networks. The detectors are pretrained on MS-COCO dataset [Lin et al. 2014], which comprises natural and "on-ground" scenes. We conduct our experiments on the UAVDT dataset (details in Section 6.1). This combination of the SSD detection model with the MobileNet backbones is known to be accurate while being efficient and minimizing latency, even outperforming computationally more expensive models like FasterRCNN in UAV scenes [Du et al. 2018].

Impact of Altitude and Lighting conditions: Inference tasks are heavily dependent on the characteristics of the data. For instance, flying the drone at a lower altitude would consume more energy, as a smaller area is captured in the field of view of the camera, requiring the drone to travel more distance to cover a target area. A decrease in flying altitude will result in a *quadratic* increase in the distance to be covered by drone, and thus an altitude drop is costly. However, this may ultimately be necessary if the DNN model suffers from unacceptable degradation in accuracy

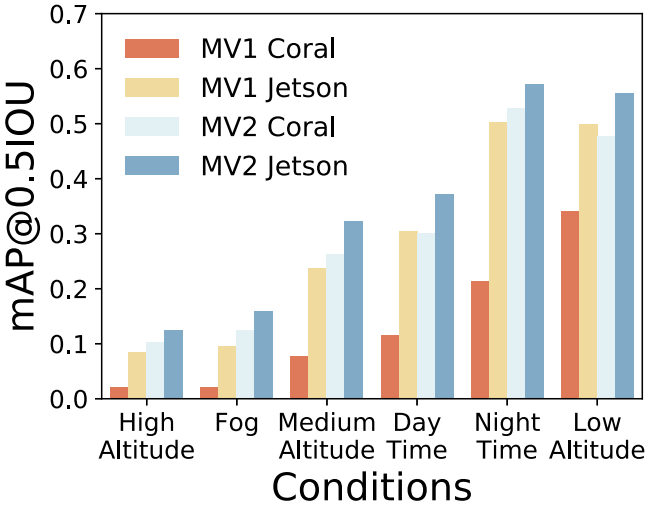


Fig. 6. Detection Accuracy across various conditions.

from using the more distant imagery obtained at higher altitudes. Furthermore, changes in weather conditions, such as fog, could reduce visibility and necessitate flying at a lower altitude. We do not consider the direct power consumption attributable to computation, which could consume ≈ 5-10W (E.g., Jetson Nano, etc.)—equivalent to less than 2% of the energy consumed by the drone used in this study.

Figure 6 shows detection accuracy across various altitudes and lighting conditions for the different hardware and model combinations. As expected, increasing altitude of the drone results in a drastic drop in accuracy, due to the smaller size of the objects in view. It is interesting to note that night time conditions yield better accuracy than day time conditions. This is possibly due to models learning to identify vehicles based on their headlights and taillights being on during the night. Also, foggy conditions are especially bad for model performance.

Observation. *Altitude and lighting conditions significantly influence the accuracy of the sensing task.*

2.6 Energy-Aware Inference Modeling

Having discussed the tradeoffs between various on-board edge devices, model quantization and drone altitude, we present our framework for predicting accuracy, given the edge device, model quantization level, the lighting condition and the flying altitude of the drone. Consider altitude as a discretized variable alt (with a bin-size of say b metres), while the variables relevant to a model such as model quantization, lighting, and so on, are denoted by o. Here, we model the performance to be a set of piece-wise linear functions of altitude, conditioned on the other variables. We sample the value set for performance and altitude from the experiments we have conducted earlier,

$$perf_o(alt) = m * alt + c \quad alt_b < alt < alt_{b+1}$$

where m and c are computed from the accuracy values of the task after flying the drone at the specified altitude. Given a desired accuracy level, we can now figure out the highest altitude to fly our drone at. At this altitude, our sensing will be accomplished at above the expected accuracy level with the largest field of view, thus minimizing the energy consumed.

3 ENERGY-AWARE SCHEDULING

In this section, we leverage the preceding insights to design an energy-aware drone scheduling system. We first summarize the key characteristics of drone operation that guide our design and then explain the key components of our system.

First, to survey a designated area, the drone is flown in straight lines called transects to cover the ground below the flown region with measurements. Turns are only made to move from one transect to another and do not contribute to the survey. Thus, minimizing the number of these turns is integral to optimizing energy usage as these are wasteful manoeuvres. Second, we earlier observed the impact of speed on energy consumption. These characteristics imply that the transects should be as long as possible to maintain sustained high speeds to save energy. Thus, the orientation of generated transects should be such that they are as long as possible, covering the maximum portion of the area of interest, called the optimum sweep direction [Huang 2001]. Third, we can restrict the charging only up to the inflexion point as the battery capacity increase beyond this stage is much slower. Fourth, if in a fleet there are drones with old batteries, it would be beneficial to allocate fewer manoeuvres to them as charging older batteries takes a longer time. Finally, the drone has to reach the charging station for recharging, hence some amount of energy needs to be reserved for that.

3.1 Path Planning

Concave Partitioning: The shape of the area to be surveyed could either be convex or concave. In case of a concave polygonal shape, optimum sweep direction may not guarantee coverage in minimum time due to redundant turns. Therefore, a set of convex sub-regions ensures that for each sub-region, we traverse in the optimum sweep direction to minimize energy consumption. Thus, we partition the concave space into a set of convex sub-regions [Keil and Snoeyink 1998].

Finding Transects: In this step, we determine the optimum sweep direction for each convex sub-region. We utilize the algorithm described in [Araújo et al. 2013], which determines the direction of the maximum width of the polygon obtained by rotating it at every orientation (iterated at 1 degree in our implementation). Furthermore, if two adjacent sub-regions have similar optimum sweep direction, we merge them into a single sub-region, via a threshold θ_{min}. Transects are formed along the optimum sweep direction. The other factor determining the number of transects is the altitude (which defines the field of view) and it is estimated from the desired accuracy level of the sensing task (as described in Section 2.6). For a given sub-region, and given sweep direction and altitude, there are four ways in which the entire polygon can be surveyed. Our energy model requires acceleration data to estimate energy usage to cover various transects. Thus, we simulated acceleration data for different transects.

Connecting Sub-Regions: After obtaining the transects, we need to traverse between the sub-regions in such a way that energy consumption is minimized. We call the inter-region traversal distance as transition distance. For connecting the sub-regions, we should minimize the sum of transition distance from one sub-region to another. A brute-force approach compares the total transition distance for all the possible orderings of the sub-regions and for each permutation of four possible traversals of a sub-region. This approach is computationally expensive as run-time is exponential to number of sub-regions. Thus, we first decide the order of visiting every sub-region. We model the sub-region ordering problem as a graph problem. Each sub-region is considered as a vertex, and the edge weights are the inter-centroid Euclidean distance between them. The proposed formulation is equivalent to the travelling salesman problem (or the source-t min cost Hamiltonian Path problem) in a *metric graph* (due to the definition of edge weights). However, the formulation is NP-Hard and a polynomial time approximation heuristic exists, the *Christofides algorithm* [Christofides 1976; Hoogeveen 1991]. The upper bound is 3/2 for the general case and

Table 3. Task Scheduling Optimization

minimize	$\mathbb{M}^{time}$	
s.t.		
(1)	$V_{\tau,d} \in \{0,1\}, \quad \hat{V}_{\tau,\tau+1,d} \in \{0,1\}$	$\forall \tau \in \Gamma, d \in D$
(2)	$\hat{V}_{\tau,\tau+1,d} <= V_{\tau,d}, \quad \hat{V}_{\tau,\tau+1,d} <= V_{\tau+1,d}$	$\forall \tau \in \Gamma, d \in D$
(3)	$\sum_{d \in D} V_{\tau,d} = 1$	$\forall \tau \in \Gamma$
(4)	$\sum_{\tau=1}^{\lvert\Gamma\rvert-1} \sum_{d \in D} \hat{V}_{\tau,\tau+1,d} = \lvert\Gamma\rvert - \lvert D\rvert$	
(5)	$E_d^{total} = \sum_{\tau \in \Gamma} V_{\tau,d} \cdot E_{\tau,d} + \sum_{\tau=1}^{\lvert\Gamma\rvert-1} \hat{V}_{\tau,\tau+1,d} \cdot \hat{E}_{\tau,\tau+1,d}$	$\forall d \in D$
(6)	$N_d = E_d^{total}/B_d$	$\forall d \in D$
(7)	$T_d^{total} = \sum_{\tau \in \Gamma} V_{\tau,d} \cdot T_{\tau,d} + \sum_{\tau=1}^{\lvert\Gamma\rvert-1} \hat{V}_{\tau,\tau+1,d} \cdot \hat{T}_{\tau,\tau+1,d} + N_d \cdot C_d$	$\forall d \in D$
(8)	$T_d^{total} <= \mathbb{M}^{time}$	$\forall d \in D$

5/3 for the s-t case. These are currently the best known bounds for the problem. This algorithm gives us the order of traversal of the sub-regions.

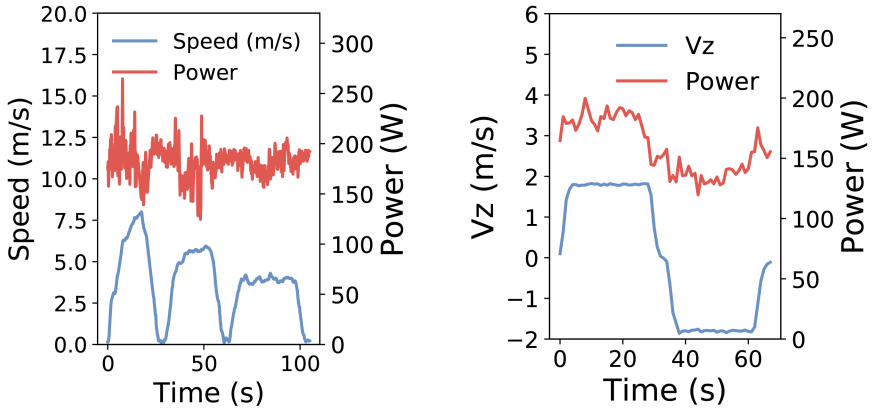
We have obtained the ordering of traversal between the sub-regions, however, there are four possibilities while traversing each sub-region. The traversed path can be found by comparing the distance between the four possible end points of the first convex sub-region to the four possible end-points of the second sub-region. After connecting the points with minimum distance, the two sub-region are combined and there are only two possible end-points of the combined sub-region. Then, compare the distance between the two end points of the combined sub-region with the four end points of the third sub-region and so on to get the final path.

3.2 Task Scheduling

We formulate an ILPprogram (described in Table 3) to solve task division among the available drones. The assumptions in the formulation of the optimization problem are as follows:

(1) All calculations are based on the assumption that the transect length is less than the maximum distance to be covered by the lowest endurance time drone. Therefore, the first transect would be completed.
(2) Whenever charging is required, the drone has access to a charging station with negligible travel time and energy.

Let $\tau \in \Gamma$ be an ordered list of transects and $d \in D$ be the set of available drones. Each drone d takes time $T_{\tau,d}$ to cover transect τ and consumes energy $E_{\tau,d}$. As we know the traversed path and can simulate the drone dynamics, $E_{\tau,d}$ can be estimated by calculating the instantaneous power from simulated acceleration using the model described in Section 2.4, and then integrating the predicted power values. The drone d has to travel from the ending of transect τ to the start of transect $\tau + 1$ (take a turn), and for this it has to waste time $\hat{T}_{\tau,\tau+1,d}$ and spend energy $\hat{E}_{\tau,\tau+1,d}$. $\hat{E}_{\tau,\tau+1,d}$ is again estimated using the model described in Section 2.4, similar to how $E_{\tau,d}$ is estimated. Each drone is allocated a contiguous list of transects to decrease the number of turns (which are wasteful as noted earlier). Let the battery capacity of drone d be denoted by B_d and time taken to charge the exhausted battery till the constant current charging mode be C_d. We can reliably estimate the battery profiles (B_d and C_d) using the models described in Sections 2.2 and 2.4 as



(a) Horizontal flights. (b) Vertical flights

Fig. 7. Survey area.

the battery charging process is linear (CC mode). Furthermore, we want to know if the drone has visited the transect τ. Let the binary indicator variable $V_{\tau,d}$ denote if d has visited τ, while another binary indicator variable $\hat{V}_{\tau,\tau+1,d}$ denote if d has covered the distance between τ and $\tau+1$. In our formulation shown in Table 3, we would want to minimize the time ($\mathbb{M}^{time}$) in which the given task can be completed. To minimize $\mathbb{M}^{time}$, we only need to look at the drone that takes the maximum time to complete its task. Let E_d^{total} be the total energy and T_d^{total} be the total time taken by d to cover the task assigned to it. These two quantities can be defined in terms of the indicator variables,$V_{\tau,d}$, $\hat{V}_{\tau,\tau+1,d}$, the time $\hat{T}_{\tau,\tau+1,d}$ and energy $\hat{E}_{\tau,\tau+1,d}$.

In the formulation, constraint (1) ensures that the variables $V_{\tau,d}$ and $\hat{V}_{\tau,\tau+1,d}$ are binary, i.e., a transect or transition between transects is either completed or not completed. Constraint (2) ensures that the intermediary distance between τ and $\tau+1$ is considered as visited by drone d, if and only if the same drone visits both the transect τ and $\tau+1$. Such a constraint ensures that a drone covers contiguous transects. Also, any transect must be visited only once in total (see constraint (3)). As we handle cases where all drones are used for the task, the total number of transitions between transects is constrained by the number of transects Γ subtracting the number of drones D, and is represented by (4). (6) denotes the number the times that drone d has to charge, to complete its task. Constraints (5) and (7) represent the total energy consumed and time taken by a drone d to complete the task allocated to it. Constraint (8) denotes that the total time taken to finish all the tasks (i.e., $\mathbb{M}^{time}$) is at most equal to the maximum time taken by any drone in $d \in D$.

Verification

We verify the task optimization formulation soon after assigning transects to individual drones. For input, we have time and energy taken by each drone to complete a transect. For verification, the output after execution provides a set of contiguous transects to each drone.

We verify the Task Scheduling Optimisation problem, as shown in Table 3, over a survey area shown in Figure 7, which generates 70 transects with 69 inter-transects that requires to be distributed among three identical class of drones. The input Table 4 shows the time and energy input required by a drone on transects and inter-transects. In the table, the even value of k represents transects while the odd value represents inter-transects.

Table 5 shows the results of the formulation. The first column of results Table 5 shows a set of transects assigned to each drone, while the second and third columns, respectively, depict the total

Table 4. Input Table

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Time	7.14	22.88	25.06	13.09	38.84	22.88	52.63	13.09	66.41	22.88	80.2	13.09	93.98	22.88	107.77	13.09	121.55	22.88	135.34	13.09
Energy	0.23	0.9	1	0.48	1.68	0.89	2.39	0.48	3.09	0.89	3.78	0.48	4.49	0.89	5.18	0.48	5.88	0.89	6.58	0.48
k	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Time	149.12	22.88	162.91	13.09	176.69	22.88	190.48	13.09	204.26	22.88	218.05	13.09	231.83	22.88	245.62	13.09	259.4	22.88	273.19	13.09
Energy	7.27	0.89	7.97	0.48	8.68	0.89	9.36	0.48	10.07	0.9	10.76	0.48	11.48	0.89	12.16	0.48	12.86	0.88	13.55	0.48
k	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Time	286.97	22.88	300.76	13.09	75.36	232.11	126.28	13.26	150.92	36.77	175.56	13.26	200.21	36.77	224.85	13.26	249.49	36.77	274.14	13.26
Energy	14.26	0.89	14.95	0.48	3.54	11.46	6.12	0.48	7.37	1.58	8.62	0.48	9.87	1.57	11.12	0.48	12.37	1.57	13.61	0.48
k	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Time	298.78	36.77	323.42	13.26	348.07	36.77	372.71	13.26	389.59	15.27	383.51	13.26	377.44	15.27	371.37	13.26	365.3	15.27	359.22	13.26
Energy	14.87	1.59	16.1	0.48	17.35	1.57	18.6	0.48	19.45	0.54	19.14	0.48	18.83	0.53	18.53	0.48	18.23	0.54	17.92	0.48
k	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
Time	353.15	15.27	347.08	13.26	341	15.27	334.93	13.26	328.86	15.27	322.79	13.26	316.71	15.27	310.64	13.26	304.57	15.27	298.5	13.26
Energy	17.62	0.53	17.3	0.48	16.99	0.54	16.69	0.48	16.37	0.53	16.07	0.48	15.76	0.53	15.45	0.48	15.15	0.53	14.84	0.48
k	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
Time	292.42	15.27	286.35	13.26	280.28	23.45	265.6	13.26	249.89	24.44	234.18	13.26	218.48	24.44	202.77	13.26	187.06	24.44	171.35	13.26
Energy	14.54	0.53	14.23	0.48	13.92	0.93	13.17	0.48	12.37	0.96	11.57	0.48	10.78	0.95	10	0.48	9.19	0.96	8.41	0.48
k	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	
Time	155.64	24.44	139.93	13.26	124.22	24.44	108.51	13.26	92.81	24.44	77.1	13.26	61.39	24.44	45.68	13.26	29.97	24.44	13.06	
Energy	7.62	0.96	6.82	0.48	6.02	0.96	5.22	0.48	4.42	0.96	3.63	0.48	2.84	0.96	2.05	0.48	1.25	0.96	0.45	

Table 5. Results of Optimisation Problem

DroneNo	k	Time	Energy	N_d
Drone0	58–86	16930.41	270.43	1.59
Drone1	0–56	16,316.00	257.76	1.52
Drone2	88–138	16,933.93	268.65	1.58

Table 6. Verification of Output

Time for Drone 0 =	274.14+13.26+298.78+36.77+323.42+13.26+348.07+36.77+372.71+13.26+389.59+15.27+383.51+13.26+377.44+15.27+ 371.37+13.26+365.3+15.27+359.22+13.26+353.15+15.27+347.08+13.26+341+15.27+334.93 + 7200*__1.59__ = __16930.42__
Time for Drone 1 =	7.14+22.88+25.06+13.09+38.84+22.88+52.63+13.09+66.41+22.88+80.2+13.09+93.98+22.88+107.77+13.09+121.55+22.88+ 135.34+13.09+149.12+22.88+162.91+13.09+176.69+22.88+190.48+13.09+204.26+22.88+218.05+13.09+231.83+22.88+245.62+ 13.09+259.4+22.88+273.19+13.09+286.97+22.88+300.76+13.09+75.36+232.11+126.28+13.26+150.92+36.77+175.56+13.26+200.21+ 36.77+224.85+13.26+249.49+7200*__1.52__ = __16315.97__
Time for Drone 2 =	328.86+15.27+322.79+13.26+316.71+15.27+310.64+13.26+304.57+15.27+298.5+13.26+292.42+ 15.27+286.35+13.26+280.28+23.45+265.6+13.26+249.89+24.44+234.18+13.26+218.48+24.44+202.77+13.26+187.06+24.44+171.35+ 13.26+155.64+24.44+139.93+13.26+124.22+24.44+108.51+13.26+92.81+24.44+77.1+13.26+61.39+24.44+45.68+13.26+29.97+ 24.44+13.06+ 7200*__1.58__ = __16933.93__
	Time Calculation
Energy for Drone 0 =	13.61+0.48+14.87+1.59+16.1+0.48+17.35+1.57+18.6+0.48+19.45+0.54+19.14+0.48+18.83+0.53+18.53+0.48+18.23+0.54+17.92+0.48+17.62+ 0.53+17.3+0.48+16.99+0.54+16.69 = __270.43__
Energy for Drone 1 =	0.23+0.9+1+0.48+1.68+0.89+2.39+0.48+3.09+0.89+3.78+0.48+4.49+0.89+5.18+0.48+5.88+0.89+6.58+0.48+7.27+0.89+7.97 +0.48+8.68+0.89+9.36+0.48+10.07+0.9+10.76+0.48+11.48+0.89+12.16+0.48+12.86+0.88+13.55+0.48+14.26+0.89+14.95+0.48 +3.54+11.46+6.12+0.48+7.37+1.58+8.62+0.48+9.87+1.57+11.12+0.48+12.37=__257.81__
Energy for Drone 2 =	16.37+0.53+16.07+0.48+15.76+0.53+15.45+0.48+15.15+0.53+14.84+0.48+14.54+0.53+14.23+0.48+13.92+0.93+13.17+0.48+12.37+ 0.96+11.57+0.48+10.78+0.95+10+0.48+9.19+0.96+8.41+0.48+7.62+0.96+6.82+0.48+6.02+0.96+5.22+0.48+4.42+0.96+3.63+0.48+ 2.84+0.96+2.05+0.48+1.25+0.96+0.45=__268.62__
	Energy Calculation

time and energy taken by each drone. The fourth column shows the number of times a drone needs a recharge. The value $N_d > 1$ for each drone implies that each drone needs recharging at least once. The total time is the sum of Transect time, inter-transect time and charging Time($N_d * C_d$ where $C_d = 7200$). Hence, we can now verify the time and energy calculations for each drone. These calculations are presented in Table 6.

By calculation shown in Table 6a, we get the total time as 16930.42s, 16315.97s, and 16933.93s for the three drones. Similarly, from Table 6b, for the three drones, we get the total energy as 270.13,

Table 7. Task Scheduling Optimization with Charging Station Consideration

minimize	$\mathbb{M}^{time}$
s.t.	
(1)	$V_{\tau,d} \in \{0,1\}, \quad \hat{V}_{\tau,\tau+1,d} \in \{0,1\} \qquad \forall \tau \in \Gamma, d \in D$
(2)	$X^c_{\tau,d} \in Z^+, \qquad \forall c \in C, \forall \tau \in \Gamma, d \in D$
(3)	$\hat{V}_{\tau,\tau+1,d} <= V_{\tau,d}, \quad \hat{V}_{\tau,\tau+1,d} <= V_{\tau+1,d} \qquad \forall \tau \in \Gamma, d \in D$
(4)	$\sum_{d \in D} V_{\tau,d} = 1 \qquad \forall \tau \in \Gamma$
(5)	$\sum_{\tau=1}^{|\Gamma|-1} \sum_{d \in D} \hat{V}_{\tau,\tau+1,d} = |\Gamma| - |D|$
(6)	$X^c_{\tau-1,d} \leq X^c_{\tau,d} \qquad \forall c \in C, \forall \tau \in \Gamma, d \in D$
(7)	$\sum_{c \in C} (X^c_{\tau,d} - X^c_{\tau-1,d}) \leq 1 \qquad \forall \tau \in \Gamma, \forall d \in D$
(8)	$(\sum_{c \in C} X^c_{\tau,d}) E^*_{th,d} \leq \sum_{i=1}^{\tau} V_{i,d} \cdot E_{i,d} + \sum_{i=1}^{\tau-1} \hat{V}_{i,i+1,d} \cdot \hat{E}_{i,i+1,d} + \sum_{c \in C} \sum_{i=2}^{\tau} (X^c_{i,d} - X^c_{i-1,d}) \cdot (E^c_{i,d}/2) \leq (1 + \sum_{c \in C} X^c_{\tau,d}) E^*_{th,d} \qquad \forall \tau \in \Gamma, d \in D$
(9)	$E^{total}_d = \sum_{\tau \in \Gamma} V_{\tau,d} \cdot E_{\tau,d} + \sum_{\tau=1}^{|\Gamma|-1} \hat{V}_{\tau,\tau+1,d} \cdot \hat{E}_{\tau,\tau+1,d} + \sum_{c \in C} \sum_{\tau=1}^{|\Gamma|-1} (X^c_{\tau+1,d} - X^c_{\tau,d}) \cdot E^c_{\tau,d} \quad \forall d \in D$
(10)	$N_d = \lfloor E^{total}_d / B_d \rfloor \qquad \forall d \in D$
(11)	$\sum_{c \in C} X^c_{\Gamma,d} = N_d \qquad \forall d \in D$
(12)	$T^{total}_d = \sum_{\tau \in \Gamma} V_{\tau,d} \cdot T_{\tau,d} + \sum_{\tau=1}^{|\Gamma|-1} \hat{V}_{\tau,\tau+1,d} \cdot \hat{T}_{\tau,\tau+1,d} + \sum_{c \in C} \sum_{\tau=1}^{|\Gamma|-1} (X^c_{\tau+1,d} - X^c_{\tau,d}) \cdot T^c_{\tau,d} + N_d \cdot C_d \quad \forall d \in D$
(13)	$T^{total}_d <= \mathbb{M}^{time} \qquad \forall d \in D$

257.81 and 268.62 after calculation. Compared with Table 5, we get a maximum error of less than 0.001% for both time and energy calculations.

3.3 Iterative Greedy Adjustment

Our current task scheduling formulation, shown in Table 3, minimizes the maximum time taken by a drone to complete its assigned transects. A situation may arise where the longest transect is significantly longer than other transects. Therefore, the time and energy taken to cover this transect are significantly high compared with the time and energy taken to cover other transects. We propose a further optimization to our algorithm and consider the possibility of partial transects, where the longest transect is split into smaller partial transects. These new partial transects and previous transects, together form a new set Γ', are then reassigned to the drones by applying our formulation again, with Γ changed to Γ'. This process of splitting the longest transect and reassigning it to drones is iteratively repeated, to further reduce M^{time}, until a stopping criterion is met. The stopping criteria can be a finite number of iterations. Too many iterations can increase transects, which means more transition points. Drones would waste time and energy due to these unwanted transition points. This has been sown in Section 7.3

3.4 Charging Station

In this section, we show our new formulation where we consider both time and energy required to reach the charging station (unlike in Table 3 Assumption 2). Consequently, our goal is to provide feasible solutions by placing several charging stations at appropriate locations. The formulation considers the drone's number of visits to charging stations as well as its residual battery energy. For larger survey areas, the drone requires several recharges to complete the task. A visit to a charging station is mandated whenever the drone's battery level falls below a specified threshold value, denoted by E_{th}. For a trip to a charging station, Table 7 captures the required formulation as follows:

(1) An integer non-decreasing variable, $\mathbf{X^c_{\tau,d}}$, represents number of times a drone d has visited charging station $c \in C$ before it encounters transect τ.
(2) Two parameters, $\mathbf{E^c_{\tau,d}}$ and $\mathbf{T^c_{\tau,d}}$ are the energy and time, respectively, taken by drone d for a trip to charging station c from transect τ. These are used in Equations (9) and (12).

(3) The Equations (6,7 and 11) in the new formulation are required for defining constraints for $X^c_{\tau,d}$.

In the new formulation of the task scheduling optimization shown in Table 7, while Constraint (6) defines the variable $X^c_{\tau,d}$ as non-decreasing with respect to τ, Constraint (7) ensures that a drone can visit only one charging station from a transect. Constraint (8) ensures that the drone goes for charging only after it has expended $E^*_{th,d} = B_d - E_{th,d}$ amount of energy, which includes the energy required to visit a charging station ($E^c_{\tau,d}/2$). Constraint (11) ensures that the total number of times a drone requires charging is equal to the sum of trips to the charging stations.

The formulation presented in Table 7 not only allocates a contiguous list of transects to a drone but also chooses a transect which offers the least energy required to reach a charging station. The output of the formulation depends upon the number of charging stations, their location and E_{th} (or $E_{th,d}$).

Another point of consideration about the charging station is that there is neither delay nor contention in obtaining the service from the charging station. The charging station is expected to support fast charging to minimize service time. The Task Scheduling formulation shown in Table 7 has the ability to handle the allocation of the nearest charging station to a drone and assumes there is no contention.

3.4.1 Impact of Number and Location of Charging Stations. We calculate the average distance to a charging station using the two endpoints of a transect. From the optimization problem formulation shown in Table 7, we must have prior knowledge of this distance in order to compute the energy ($E^c_{\tau,d}$) and the time taken ($T^c_{\tau,d}$) to reach the charging station from transect τ.

As stated earlier, the allocation of transects among the drones not only would depend upon the capabilities of drones but also upon the number and location of the charging stations. In general, we can assume for a survey area that the drone launching site will have a charging station. The energy spent on a trip to a charging station should be less than the average energy needed to cover a transect in a survey. Otherwise, significant energy is wasted on these charging trips than on covering the surveying task. Hence, the location of charging stations is essential in planning a surveying task. There are two possible scenarios for placing a charging station.

In the **planned location**, there is flexibility to place the charging station. To accomplish this, we use Table 7, with the following changes:

(1) Replace Equations (9) and (12) of problem in Table 7 with Equations (5) and (7) of original formulation presented in Table 3. This change will not consider the time and energy required for a trip to a charging station.
(2) In Equation 8 of Table 7 remove the term $\sum_{c\in C}\sum_{i=2}^{\tau}(X^c_{i,d} - X^c_{i-1,d}) \cdot (E^c_{i,d}/2)$. This change does not consider a visit to a charging station on a transect τ.

With these changes, the solution to formulation will report the transects on which charging is required. Furthermore, geometrical methods can be used to find the optimum location of charging stations.

In the **fixed location**, the drones have to use existing charging stations. The formulation presented in Table 7 can be used to check the feasibility of the task of surveying the area with the given charging stations.

3.4.2 Impact of E_{th}. Constraint (8) in the optimization formulation shown in Table 7 ensures that a drone will visit a charging station only after the battery level has reached the threshold value E_{th}. Its value can impact the number of drone trips to a charging station. Choosing a higher value of E_{th} means that drones will need to visit charging stations more frequently and at earlier points in their flights, thereby increasing the overall mission time. Conversely, setting a low value

of E_{th} (such as 0) can also be problematic, as drones may not have enough energy left to actually reach a charging station. Thus, an optimal value for E_{th} must be carefully chosen. One possible method is to set E_{th} based on the energy required to travel from the drone's farthest transect to a charging station. Other factors that should be considered include the battery profile and its charge-discharge cycles. As discussed in the Impact of ageing in Section 2.1 on charging, an older battery will discharge faster than a new battery for the same amount of energy. Therefore, the E_{th} value must be chosen according to the drone's battery discharge model and battery profile, without increasing the mission time.

4 ROBUSTNESS IN TASK COMPLETION

The Task scheduling optimisation problem, as indicated in Table 3, assumes robustness of all wireless links and drone system components. Therefore, once the **Ground Control Station** (**GCS**) (henceforth called as **Base Station** (**BS**)), assigns tasks to the drones, it is assumed that the drones receive the information reliably and successfully complete their assigned tasks. However, real-world systems are not entirely reliable, and may encounter off-normal cases such as unreliable wireless links and failure of drone equipment. Equipment failure can be attributed to events such as strong gust of wind, collision with an object like a bird, tree, another stray drone that might come in the drone's path. To handle drone failure, one approach for the BS is to alert neighboring drones to complete the transect of the affected drone. Link failures might occur due to deep signal fading. Due to the nature of wireless links in a high-mobility application, a temporary degradation of signal strength may occur. A signal outage at the BS is associated with an ambiguity of either a link failure or a drone component failure. The link failure can create a catastrophic situation because the BS might issue a task recomputation which will lead to a wasteful expenditure of system resources and can be potentially dangerous due to the possibility of collision. We consider three specific scenarios detailed below.

Scenario 1 - Reactive: In this scenario, the BS recomputes the tasks for the remaining drones only after it does not receive a signal from the drone. Since there is a significant delay due to algorithm recomputation, there is a wasteful energy expenditure, particularly when drones are reassigned the transacts already covered during this delay and requested to start afresh.

Scenario 2 - Out of range Scenario: In this scenario, a drone temporarily goes out of range while covering a transect and hence, temporarily disconnects from the BS. The BS waits for a specific time TR_{th} (explained in a later section) before triggering a task recomputation. The time the drone remains disconnected may be more than TR_{th} which results in an unnecessary and potentially dangerous recomputation.

Scenario 3 - Return To Home: In this scenario, drones are internally programmed to abandon a task and return to the launch site if it fails to get heartbeat signals from the BS.

We refer to the affected drone (shaded in red) to be drone D_1. Figure 8 illustrates both cases where: (i) only the link between D_1 and BS is unreliable and suffers a temporary degradation and (ii) failure of D_1 components, which leads to all links to D_1 being down. We propose a reliable communication mechanism to make the BS proactive in assessing the requirement of a task recomputation according to system state. We describe the sequence of steps to recover from the event of a node/link failure. Above scenarios are addressed as it reduces the delay in Scenario 1, avoids an unnecessary task recomputation in Scenario 2 and prevents a pre-mature return-to-launch decision in Scenario 3.

4.1 System Architecture

In this section, we present the system architecture consisting of drones and the BS. We also mention a few readily available data from this system, which we exploit to provide robustness to our

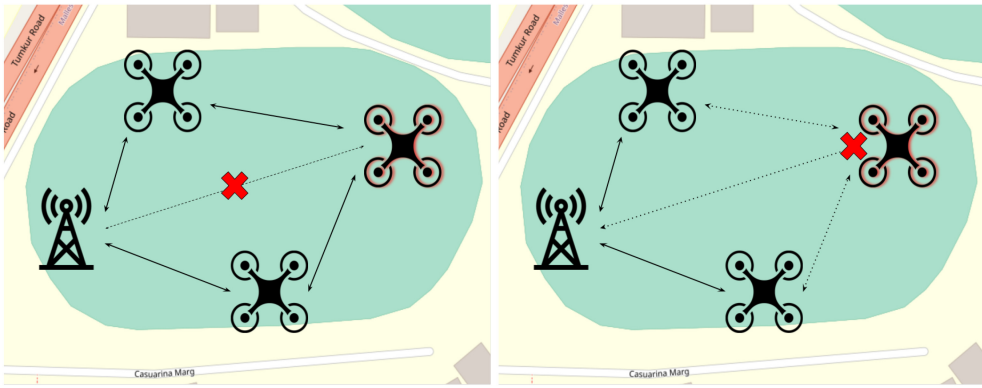


Fig. 8. Communication loss between drone D_1 (shaded in red) and BS.

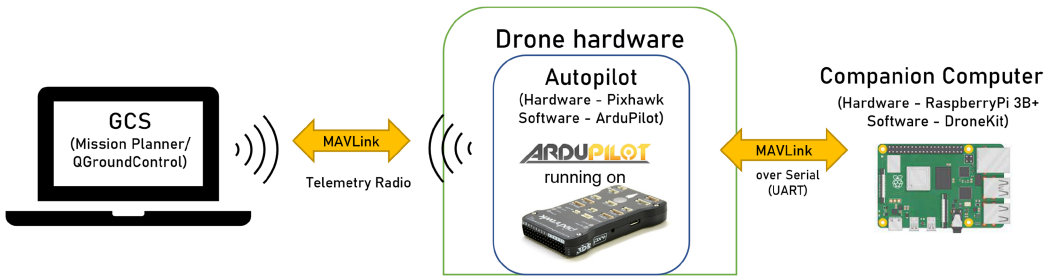


Fig. 9. System Architecture to acquire drone state using MAVLink protocol.

system. We explain our system using Figure 9, which represents our system architecture and is explained here. Each node fetches sensor state from connected sub-modules such as IMU, GPS, and Telemetry Radio. We choose our autopilot as the open-source ArduPilot firmware running on Pixhawk autopilot board. The autopilot features MAVLink, drone-specific protocol based on publish-subscribe model, to communicate between various external sub-systems such as BS and companion computers such as RPi. The drone communicates with the BS via a Telemetry Radio (SikRadio) wireless serial link at 433 Mhz using MAVLink protocol. The drones and BS form a wireless sensor network. Each drone acts as a wireless sensor node (henceforth drones are referred to as nodes) with the capability to transmit current state to other nodes either directly or through multi-hop transmission, as seen in Figure 8.

4.1.1 Telemetry Data. The autopilot sends different categories of MAVLink messages to the GCS and Companion Computer using pre-configured rates defined by the user. Examples of MAVLink messages are IMU data, GPS data, heartbeat packets. We set the rate for Heartbeat Packets to 4 Hz. We configure the BS to get this data via wireless telemetry link, sent by each node at regular intervals. Among these, nodes transmit heartbeat packets, along with RSSI, position, velocity data. The BS monitors the received heartbeat packets and computes the **Packet Delivery Ratio** (**PDR**) each second, i.e., no. of packets received by BS over number of packets sent by node.

We propose to use such readily available telemetry data to determine a failure. The next three subsections describe the following:

— Step 1 - As the link strength starts to degrade, BS triggers a Reliable Communication request (ReqCom) to the affected node (D_1).

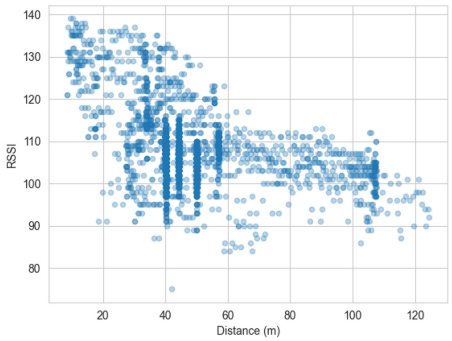


Fig. 10. Variation of RSSI with distance.

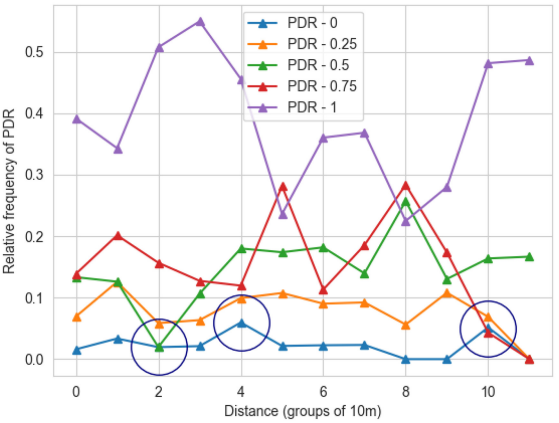


Fig. 11. Variation of relative frequency of PDR with distance.

— Step 2 - Using our proposed method, the node establishes a reliable link.
— Step 3 - If no signal is received in time TR_{th}, a decision of Task Replanning is taken.

4.2 Step 1 - Reliable Communication Trigger

This section describes our methodology which is executed at the BS when a link failure is detected. We present our observations on the variation of link strength with the distance between nodes, obtained from our flight log dataset. The trigger point is determined using these observations.

4.2.1 Link Strength. The BS monitors link strength information at each interval. Link strength can be denoted by RSSI or PDR. Figure 10 illustrates that RSSI overall decreases with distance. However, high fluctuation is observed due to the mobile nature of drones, a swift change in orientation, the quality of drone wireless hardware components, and interference from environmental factors. Therefore, it is not an ideal choice for making a decision. Drones transmit four heartbeat packets each second to the BS. The community has not exploited heartbeat signals for this purpose. We plot PDR i.e., the number of packets received each second at the sink node over that sent by source node. Figure 11 illustrates the variation of PDR with distance. The distances are grouped in bins of 10 m for easier representation. The relative frequency of occurrence of PDR values of 0, 0.25, 0.5, 0.75, and 1 are plotted. A PDR of zero indicates packet loss of 100%. We observe the trend of zero PDR and use this correlation to determine when to transmit a ReqCom.

4.2.2 Trigger Algorithm: Let N be the set of nodes = n_{D1}, n_{D2},..,n_{Dk}, n_{BS} where n_{BS} denotes the BS. Let $PDR_{a,b}$ be the PDR between node n_{Da} and node n_{Db}. Let relative frequency of $PDR_{a,b} = k$ be $F_{a,b,k}$. If $F_{a,BS,0}$ i.e., relative frequency of zero PDR between affected node D_1 and BS, > 0.025 or $F_{a,BS,0.5} < F_{a,BS,0}$, trigger a ReqCom to node n_Da. Such points are marked with blue circles in Figure 11.

4.3 Step 2 - Proposed Robust Communication Mechanism

When the communication link starts to degrade, the node requires alternate communication links via which it can send packets reliably to the BS i.e., a reliable communication link needs to be established between the affected node and BS. One way to accomplish this is for the BS to compute alternate routes based on PDR and get a disjoint set of routes called *RouteList*. It employs the wireless nature of the link to broadcast a ReqCom including the *RouteList*.

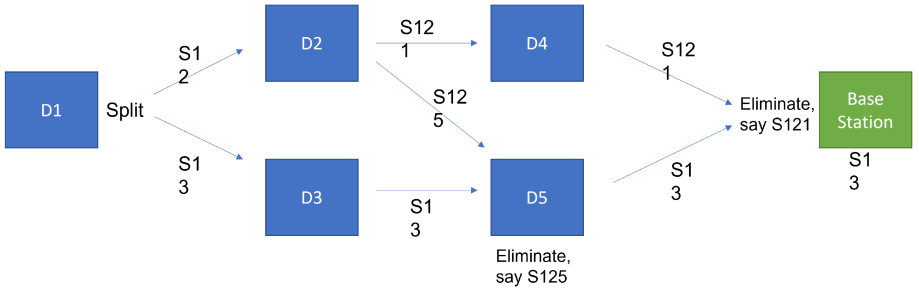


Fig. 12. Frame replication and elimination (FRER).

4.3.1 RouteList. The BS uses the knowledge of $PDR_{a,b}$ to compute all possible disjoint routes and then ranks them according to hop count and PDR of end-to-end route. PDR of end-to-end route is calculated as follows: If a route from n_a to n_{BS} is via n_i and n_j, then the PDR of this end-to-end route is

$$PDR_{ab} = PDR_{a,n_i} * PDR_{n_i,n_j} * PDR_{n_j,n_BS}$$

The rank is assigned to each route k using the following equation:

$$RouteRank(k) = HopCount(k)/PDR(k)$$

Therefore, the route with the fewest hops and a higher PDR gets a higher rank. If the node is alive, it uses the routes to transmit heartbeat packets to the BS until a direct link to BS is re-established. The BS resumes receiving packets and does not issue a task recomputation. The node cannot transmit any heartbeat packets if it is down and moves to the next step - Step 3.

Next, we consider the route that packets take from node D_1 to BS. The discussion is as follows. The route with the highest rank can be chosen. However, it may so happen that one of the hops in the route may have a low link strength. For example, a 3-hop route with PDR = 100 is ranked higher than that with PDR = 75. However, the PDR of the hops in those routes could be 10,1,10 and 4,6,6. Clearly, there is a high probability of packet loss in the higher ranked route because of a hop with very low PDR. As an alternative, to increase the reliability of the packet delivery, the packets can be replicated and transmitted on all routes. However, this leads to overloading of the network and is not an efficient solution. We propose an algorithm that uses a combination of multiple routes and the concept of Frame Replication and Elimination for Reliability (FRER). The following section explains our algorithm based on FRER.

4.3.2 Frame Replication and Elimination (FRER). The IEEE 802.1CB FRER, proposed in **Time-Sensitive Networking** (**TSN**) standards by the IEEE 802.1 Working Group, addresses reliability (reduce packet loss) for a stream of packets using redundant paths in the event of link or node failures. It involves replication of a packet in the source node and relay nodes, sending it via redundant paths, and elimination of duplicate packets received in the destination node and relay nodes. Figure 12 illustrates how FRER works. The standard does not talk about which nodes perform replication and elimination functions, and we propose an approach using PDR to select packet replication decisions for reliable communication.

When a node receives a packet, it reads the packet metadata, and uses our decision-making algorithm to take one of the following three decisions:

— 1 Elimination: The node identifies the packet number and the stream it belongs to using the sequence number defined above. We use a naive approach wherein the first packet is

ALGORITHM 1: Packet Processing Algorithm

Packet Structure: ReqCom packet = | RouteList header | Hop header
RouteList header = | Ether | IP | Route1 | Route2 |
Route header = | Hop1 | Hop2 |
Hop header = | node.mac() | PDR_{node} |
Decision: forward or eliminate packet

```
pkt.receive()
/* check if packet is a duplicate */
if pkt is duplicate() then
    eliminate()
else
    /* check PDR and forward to neighbour */
    if pkt.hdr.hop(n).PDR(n + 1) < PDR_th then
        neighbours = scan(station dump)
        if neighbour in neighbours(RouteList) then
            forwardToNeigh = neighbour with max(signal)
            pkt.clone(forwardToNeigh)
        else
            pkt.simpleforward(nextHop)
        end if
    end if
end if
```

allowed, and subsequent packets received with the same sequence number are dropped and thus eliminated.

— Simple forwarding: If the next hop's $PDR > PDR_{threshold}$, forward the packet to the next node specified according to the route

— Replication: If the next hop's $PDR < PDR_{threshold}$, replicate packets into two duplicate streams and forward it to a node in the neighboring route with the best PDR. The first packet belonging to this stream is assigned a sequence number identifying the individual packet and the stream, and it is incremented for each subsequent packet from the same stream.

4.4 Step 3 - Task Replanning

The BS broadcasts the ReqCom and waits for time TR_{th} to receive the status of D_1. Meanwhile, it computes the remaining transects during TR_{th} and uses the algorithm in Table 3 with the change in sets of transects from Γ to Γ' and set of drones from D to D' to assign the unfinished transects to the remaining functional drones. For further optimization, drones near the faulty drone should be considered in set D'. What should be the value of TR_{th}? Let the Maximum number of hops between a node and BS be $Mhop$. A ReqCom request requires $Mhop$ time intervals to reach the farthest node, and a node then requires one time interval to start transmitting via an alternate route and the packets require another $Mhop$ time intervals to reach the BS. Hence, We compute TR_{th} to be $Mhop * 2 + 1$. If BS fails to receive any packet from node D_1 during this time, it uploads the re-assigned tasks to the nodes.

5 IMPLEMENTATION

5.1 Open Source GCS

Existing GCS softwares are highly sophisticated and versatile as they provide support for various platforms, such as Pixhawk [Meier et al. 2015]. They are capable of handling detailed telemetry

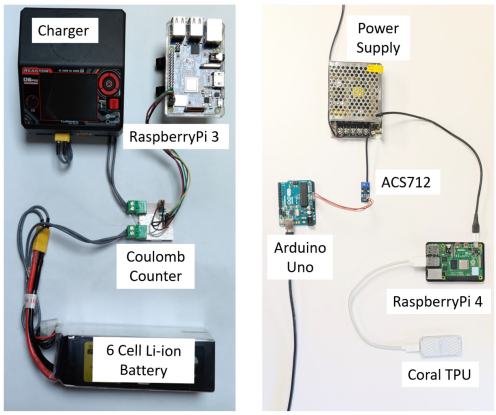


Fig. 13. Energy monitoring setups.

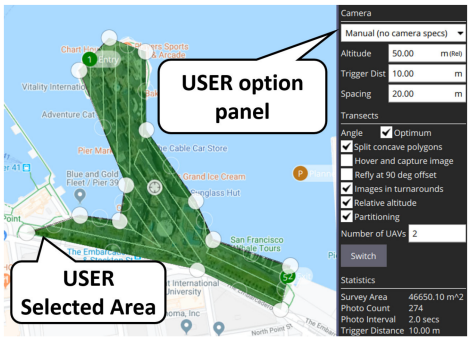


Fig. 14. UI for *Smart Survey* feature in QGC.

and uploading flight plans using multiple connection options such as TCP/UDP, and so on. Thus, we implemented our system as an extension in QGroundControl (QGC). QGC is open-source, and the most widely used GCS with a significant active contributors base. Specifically, we implemented our algorithm as an additional tab called *Smart Survey*, which is an improvement over the existing *Survey* tab that implements a basic path planning algorithm with no energy considerations. Figure 14 shows the UI design for our *Smart Survey* tab. The entire implementation involves around 2000 lines of C++ code and around 400 lines of QML code (our open-source implementation: https://github.com/t-sriyen/qgroundcontrol). For simulating accelerometer values for the energy predictions, we implement Allan variance algorithm [gnss 2021].

5.2 Energy Model Toolkit

To learn the battery charging parameters, we used *PyStan*, a Bayesian modeling library with several MCMC samplers. We use *Sklearn*, a Python-based machine learning library to train our energy discharging models. As part of this work, we release an open-source toolkit consisting of the battery charging and discharging models. Furthermore, the toolkit also contains a framework to calculate the optimal altitude that minimizes energy usage of drones for any UAV-based sensing, given enough domain information. Moreover, we also release our detailed drone flights datasets containing GPS and IMU logs along with flight plans (our open-source toolkit: https://github.com/t-sriyen/DroneEnergyModeling).

5.3 Energy-Aware Model Inference

Our object detection models are trained and fine-tuned on a server equipped with a Nvidia Tesla K80 equipped with 64 GB RAM using Tensorflow 1.12 with the Tensorflow Object Detection API. Our trained models (SSD-MV1 and SSD-MV2) cannot be executed as-is on either edge platforms and thus were converted to their respective proprietary formats—i.e., tflite model for Google Coral and UFF model for Jetson Nano.

5.4 Hardware Implementation

For our experiments, we use TATTU Premium 22.2V 6-cell LiPo batteries with an 8000mAh capacity and discharging rate of 15C. We used a quadcopter with an onboard Pixhawk flight controller along with a RaspberryPi 3B+ as a companion computer to log IMU data, along with energy used to perform different drone maneuvers. For observing charging trends of the battery, we used a

Turnigy Reaktor D6 Pro Duo charger and a coulomb counter, designed using IC *LTC2944*[LTC2944 2017]. Figure 13(a) shows this energy monitoring setup. For experiments involving ML inference on the edge platforms, our power logging setup involved a current sensor (ACS 712) and an SMPS regulated at 5 volts, shown in Figure 13(b).

5.5 Reliable Network

We leverage *P4*[Bosshart et al. 2014] i.e., Programmable Protocol-independent Packet Processing language to implement our reliability algorithm and process the MAVLink messages. We use *Mininet-WiFi*[Fontes et al. 2015], a network emulation tool to emulate three drones and a BS. We assume for our article that the BS has acquired the topology information from MAVLink messages using P4 and computed a RouteList. Additionally, we assume BS has notified the drone D_1 with a ReqCom containing the RouteList. There are two cases: one where D1 has failed and hence, no heartbeat packets can be transmitted. In the other case where it is alive, it generates packets and appends the RouteList header having a defined header structure. The interface replicates the packet to the first hop of the two routes Route1 and Route2.

Each node in the network runs our P4 program. In addition, each node periodically receives broadcast information about neighboring nodes together with signal strength, from the control plane using the OS's wireless interface driver, and is stored in data plane memory in the NIC of the node toward fast lookup using P4 match-action tables. We parse each incoming packet's header in the data plane. We use the IP header's ID field to identify if a packet is a duplicate. If it is an original packet, the next hop's PDR is compared with PDR_{th} and if it is found to be higher, then it is forwarded to the next hop. However, if it is lower, it replicates the packet and sends it to the next hop of a neighboring route. Figure 21 demonstrates that our algorithm ensures reliability due to a node continuously going out of range and still maintaining PDR using replication and elimination in comparison to no replication.

6 EVALUATION METHODOLOGY

6.1 Datasets

ML Inference: We conduct our experiments on the UAVDT [Du et al. 2018] object detection dataset, which consists of 80,000 frames of UAV flights. The frames are labeled with object bounding boxes of vehicles and other attributes—such as weather conditions, flying altitude, and so on—that are useful in characterizing performance of object detection algorithms in various real-world scenarios.

Path Planning: We created a dataset of prominent public places, including parks, piers, and city squares, and so on spread across four cities—New York, San Francisco, Bangalore, and London. These areas vary greatly in shapes and sizes, ranging from 0.2 to 5.7 sq. km. These sites form the set of shapes given as input to the algorithms. We will release this dataset as part of the toolkit to aid benchmarking.

6.2 Metrics

Energy Modeling: For this, we use **Mean Absolute Percentage Error** (**MAPE**) as our performance metric.

ML Inference: We measure the accuracy of inference using Mean Average Precision with $IOU \geq 0.5$ (mAP@0.5) [Everingham et al. 2010].

Path Planning: We characterize the input shapes using convexity, which is defined as the area of the shape divided by the area of convex hull of the shape [Sonka et al. 1993]. We characterize the performance by measuring both energy and the time taken for a planned path.

6.3 Baselines

Energy Modeling: We compare the performance of our energy discharge model with the start of the art [Tseng et al. 2017], which modeled the drone energy consumption as a linear combination of several factors that include the drone weight, wind speed, velocity, and so on.

Path Planning: We compare the performance of our path planning approach with three existing algorithms - DroneDeploy (East-to-west algorithm) [DroneDeploy 2020], QGC [Zurich 2013] and FarmBeats [Vasisht et al. 2017] using energy consumption as the metric. DroneDeploy generates sweeping patterns from east to west or west-to-east regardless of the shape of the surveying area. FarmBeats generates the path that minimizes number of waypoints. In QGC, the sweep direction is left to the user to select. In our evaluation, we kept the sweep direction fixed from north-to-south.

Task Scheduling: We consider the equal transect division algorithm as our baseline, i.e., the transects are equally divided amongst all the drones available to us. This is in contrast to our system wherein a linear programming model is proposed. We also test the iterative greedy adjustment on one of the largest survey area. We ran the iterative approach for 15 iterations without any threshold.

Charging Station: We consider our formulation presented in Table 3 as baseline. We use M^{time} as a metric to compare the impact of the location and the number of charging stations.

7 EXPERIMENTAL RESULTS

7.1 Energy Modeling

Using our energy model, we tried different regression algorithms: RF, KNN and SVR. Our model consists of several components (i.e., $P_x y$, P_z, P_{wind}, P_{drag}) as features. *Our model outperforms baseline model* as shown in Figure 22. We used over 100 different drone flight data and categorized them into four groups. The first two groups are called *vertical* and *horizontal* flight datasets. As the name suggests, these flights only involve drones either moving in horizontal or vertical directions. The *vertical triangle* group is similar to our workload discussed in Section 2.4. In the dataset group called *full*, every flight path was included.

Within each group, we performed a leave-one-out cross-validation approach to characterize the performance of our model (Proposed) compared with Tseng et al. [Tseng et al. 2017] (see Figure 22). As stated earlier, we model the energy consumption using a RF regression as it out performs the other (KNN and SVR) algorithms. Also, with more data points, as is the case in the *full* dataset group, the performance improves to an average MAPE of 5.6% for the predicted energy consumption across all manoeuvres. Clearly, our model significantly out performs the model presented by Tseng et al. [Tseng et al. 2017], as our average MAPE values are 2.5x lower than theirs in the average case. Our proposed method performs even better in horizontal and vertical manoeuvres, where we report 3 to 3.5x improvements in MAPE. We conducted a detailed evaluation of our charging models and observed an average MAPE of around 1.94% in predicting the transition point between the CC and the CV mode. Furthermore, the R^2 fit of the charging models for voltage and current over 100 sessions never dropped below 0.98.

Summary: *Our energy model demonstrates an average MAPE of 5.6% — a 2.5x improvement over the state-of-the-art.*

7.2 Path Planning Evaluation

To evaluate our system's performance on path planning tasks, we compare it with three baseline algorithms (DroneDeploy, QGC, Farmbeats) on a dataset discussed in Section 6. In many different real-world surveying tasks, such as mapping districts, river deltas, and so on, the underlying shape is not convex. Thus, in Figure 15 we show the graph of the relationship between relative energy

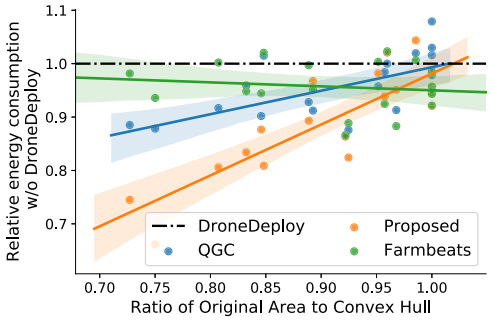


Fig. 15. Relative energy improvement of proposed method for shapes with different convexity.

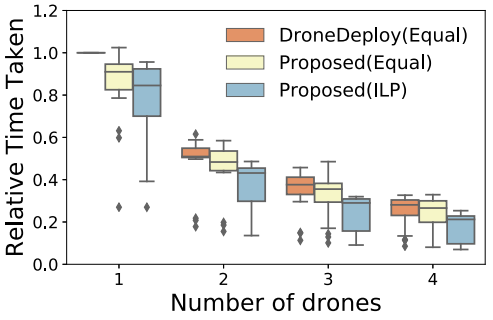


Fig. 16. Relative time taken with increasing number of drones.

consumption w.r.t DroneDeploy and Area Convexity. Area Convexity is defined as the ratio of the survey area to the area of its convex hull. When the convexity is less than 0.95, the proposed algorithm shows significant improvement in energy consumption. The percentage improvement in energy when the convexity lies between 0.85 to 0.95 is 11.29%. The percentage improvement in energy consumption when the convexity is between 0.7 and 0.85 is 21.14%. The percentage improvement in the energy consumption when the convexity is greater than 0.95 is 3.09%, which is similar to the Farmbeats algorithm. Hence, when the shape is not convex, it is beneficial to partition the shape. When the shape is nearly convex, the paths generated by FarmBeats and our algorithm are very similar. Hence the energy consumption does not change too much.

Summary: *Our system outperforms the three baseline algorithms in energy consumption. The improvement is more pronounced when the area convexity is lower.*

7.3 Task Scheduling Evaluation

To evaluate the performance of our proposed system for multi-drone scheduling, we use the same dataset used in the above path planning evaluation. We vary the number of drones and generate the path using our algorithm and the equal transect division algorithm. Figure 16 shows the relative time taken by three approaches—(i) *DroneDeploy (Equal)*, where the path is generated using DroneDeploy and task division is done using equal transect splitting algorithm, (ii) *Proposed (Equal)*, where our system does path planning and task division is done using equal transect splitting algorithm and (iii) *Proposed (ILP)*, where the path planning and task division is done by our system using the proposed linear programming approach. As expected, the time taken decreases with an increase in the number of drones. In all the cases, the *Proposed (ILP)* based task division algorithm results in the least time taken. Specifically, with just two drones, the average reduction in time taken to complete surveying tasks in our dataset by *Proposed (ILP)* compared with *DroneDeploy (Equal)* the 35.97%. With four drones, the reduction in time is 46.91%. This is because the *Proposed (ILP)* algorithm considers various factors—such as charging profile, energy consumption for various maneuvers, time taken per transect, and so on—to divide the task among available drones optimally.

We use one of the survey areas (Figure 7) from the dataset, used above in path planning evaluation, for evaluating the greedy adjustment. We used 4 drones and used our formulation shown in 3 for task division. From our experimentation, the advantage of the iterative greedy approach depends on the survey area in consideration. The results can also be affected by the class of drones used. Figure 17 shows the values for 15 iterations for time and energy consumed by each drone on each iteration. There is no considerable improvement in this case, where all the

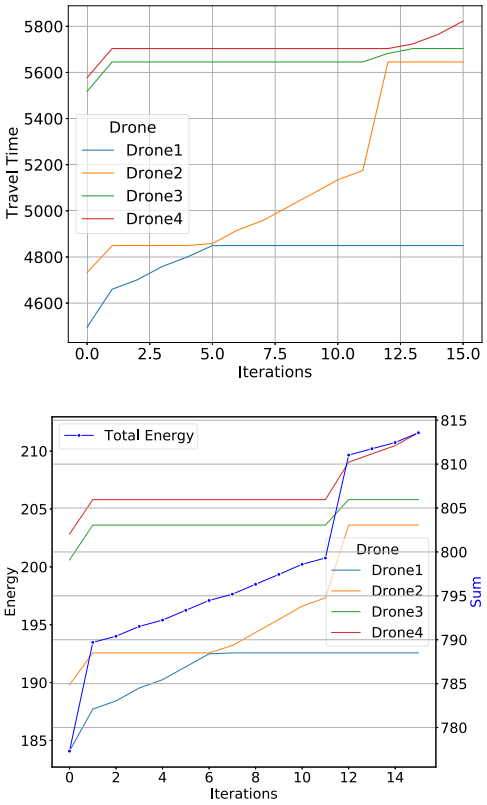


Fig. 17. Time and Energy plot for 4 same capability drones after greedy algorithm.

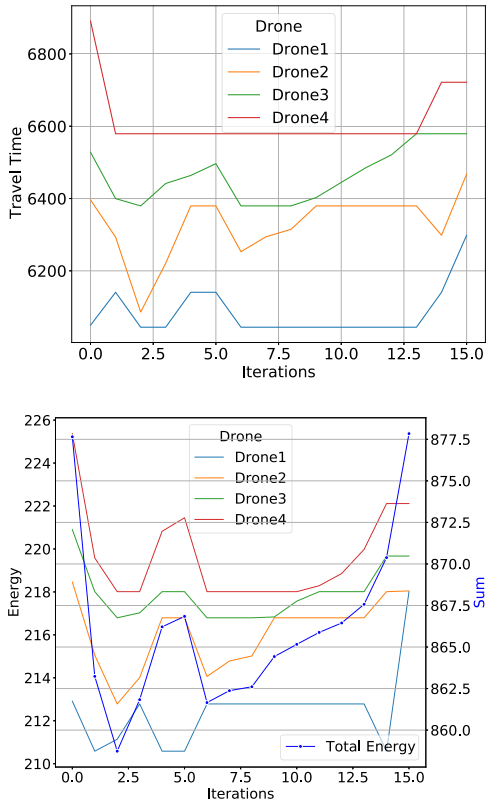


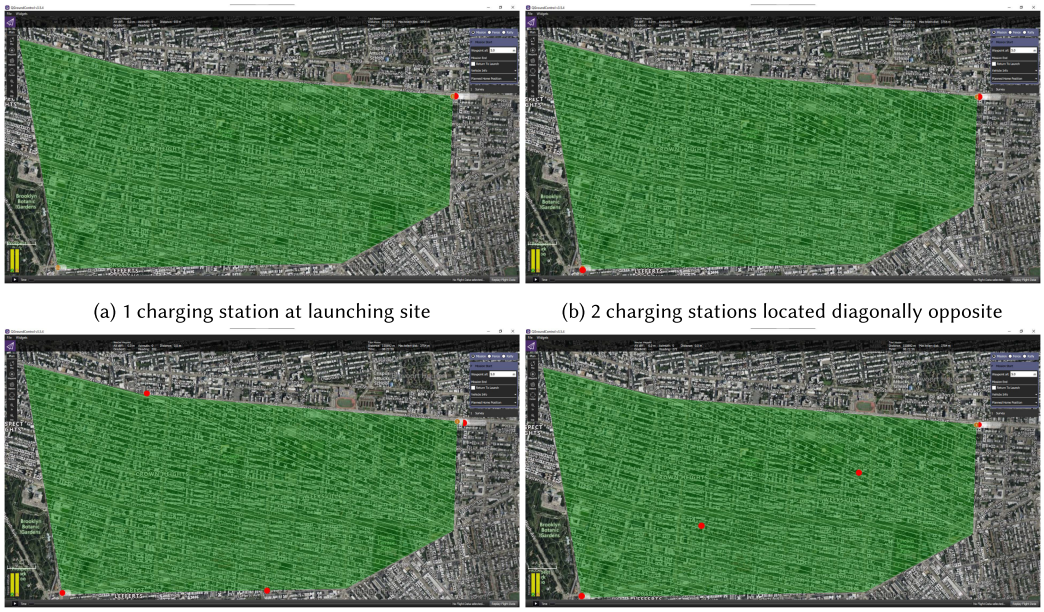
Fig. 18. Time and Energy plot for 4 different capability drones after greedy algorithm.

drones are of the same class. This is because the splitting of the transect will be distributed to the same class of drones(maximum velocity is $10m/s$). This distribution will cause more energy consumed and time used as the number of transects increases. However, Figure 18 shows about a 5% reduction in maximum travel time and 3% in maximum energy consumed after the second iteration. This is because the redistribution of transects happens among 4 drones of different classes(maximum velocities are 10 m/s, 5 m/s, 10 m/s, and 12 m/s, respectively). The new transects are redistributed among better-capable drones, causing an overall decrease in time and energy usage. However, as we increase the iterations, more transects cause an increase in time and energy usage.

Summary: *Our proposed system achieves about 47% reduction in mission times with multiple drones than the baselines. Our Iterative greedy adjustment can further provide a reduction of 5% in some cases.*

7.4 Charging Station Evaluation

We evaluated the new formulation presented in Table 7 on a survey area shown in Figure 7 of 570 Hectares. In Figure 19(a), we have 1 charging station at the launching site. In Figure 19(b), we have 2 charging stations on diagonally opposite corners. In Figure 19(c), 4 charging stations are located on the perimeter, while in Figure 19(d), the charging stations are placed in the middle of the survey area. The charging stations are marked in red circles.



(a) 1 charging station at launching site

(b) 2 charging stations located diagonally opposite

(c) 4 charging stations on the perimeter

(d) 4 charging stations in the middle

Fig. 19. Survey area with different configurations for charging stations.

Figure 20 shows the maximum time (M^{time}) increases with respect to the previous formulation (baseline) because the time for a trip to a charging station is included in the current formulation. Also, for more than 1 drone, increasing the number of charging stations from 1 to 4 reduces the time to complete the task by 1.7%. This is because the charging time is significant in comparison to the travelling time. A slight improvement in M^{time} is also observed on changing the configuration from 4 charging stations on the perimeter(Figure 19(c)) to the middle(Figure 19(d)) of the survey area. For a single drone covering the survey area, the appropriate location and number of charging stations can significantly reduce M^{time} by 15%.

As observed in Figure 20, a high value of E_{th} such as $0.2 * B_d$, can increase visits to the charging stations, thereby increasing M^{time} significantly. For the cases, when the Number of Drones is 2 and 3, M^{time} has increased by 35% and 60%, respectively.

Summary: Our new proposed formulation considers the presence of charging stations. The location and the number of charging stations significantly impact the mission time of a single drone. A high value of E_{th} can significantly impact mission time.

7.5 Impact of Altitude

To study the effect of the altitude on the accuracy of object detection models and the consequent impact on task-level energy consumption, we use UAVDT [Du et al. 2018] dataset for identifying vehicles. We choose three object detection models YOLO [Redmon et al. 2016], Tiny YOLO [Redmon et al. 2016] and SSD-MV2 [Liu et al. 2016; Sandler et al. 2018] to compare their accuracy at different altitudes. Figure 23 shows the variation of mAP@0.5 of different models and task-level energy consumption with the altitude of drone. At higher altitudes, the field of view of the camera has increased coverage. Hence, the flight time needed to complete a surveying task could reduce. We observe that object detection models have lower accuracies at higher altitudes due to reducing object size (in terms of the number of pixels). YOLO has maximum accuracy out of the three

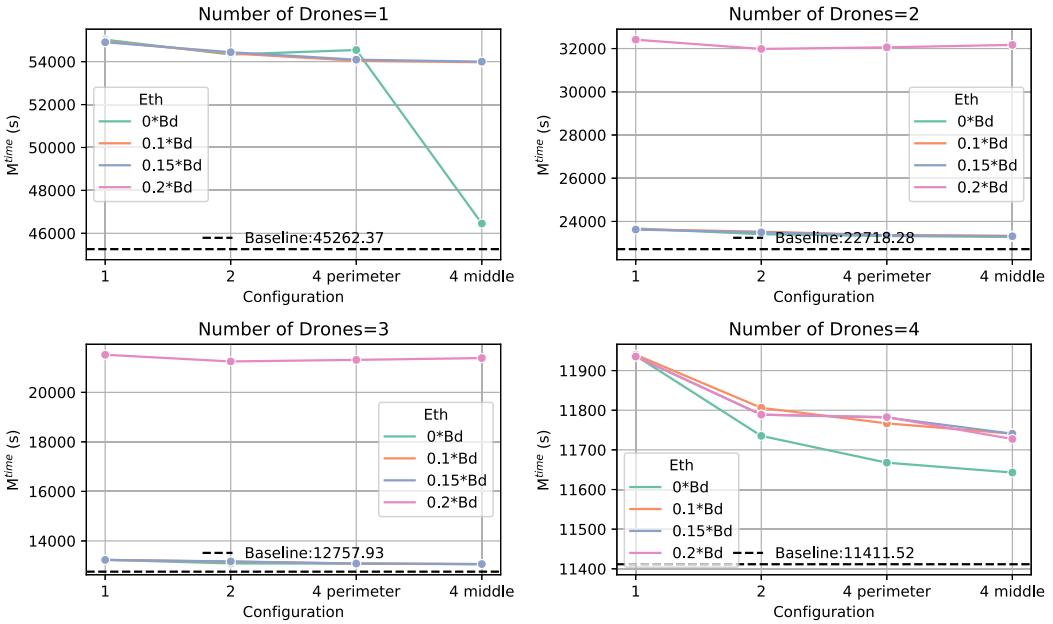


Fig. 20. Results from new Task Scheduling optimisation. The x-axis represents the configurations presented in Figure 19

models used at all altitudes. Thus, if an embedded device can run an expensive model like YOLO, it will yield greater accuracy. Furthermore, we see that the energy consumption decreases with the increase in altitude, but the decrease is not uniform. Instead, the energy decreases in a step-wise manner. This behavior is because the the flight time and hence the energy consumption depends on the cumulative length of the transects, and an increase in altitude does not necessarily mean a decrease in the number of transects. In some cases, the number of transects decreases with an increase in altitude, while in other cases, it remains unchanged, and so the energy consumption also remains essentially unchanged.

Summary: *Increasing altitude hurts object detection accuracy. However, energy consumption decreases in a step-wise manner.*

7.6 Impact of Reliability

Using our Mininet-WiFi emulation of mobile drones and a stationary BS as wireless nodes, we demonstrate the effect of our reliability algorithm on the PDR even though drone D1 moves away from BS and experiences loss of connection. We run our experiments once with a direct drone-to-BS link, and then with our replication algorithm where other nodes in the vicinity are used as alternate routes to BS. The drones moved around randomly in the survey area. Figure 21 demonstrates that PDR remains greater than 0.75 for 90% vs 60% of the time and greater than 0.5 for 100% of the time for our replication strategy and direct link respectively. However, a drastic comparison is observed for PDR greater than 0.5, where it remains for 100% of the time vs 66% for replication vs direct link, respectively. We also observe zero PDR in the case of direct link when the drone goes more than 80m away from drone whereas it still maintains a link to the BS in the case of replication strategy.

Summary: *Our algorithm outperforms the direct link in PDR. The improvement is more pronounced for longer distances.*

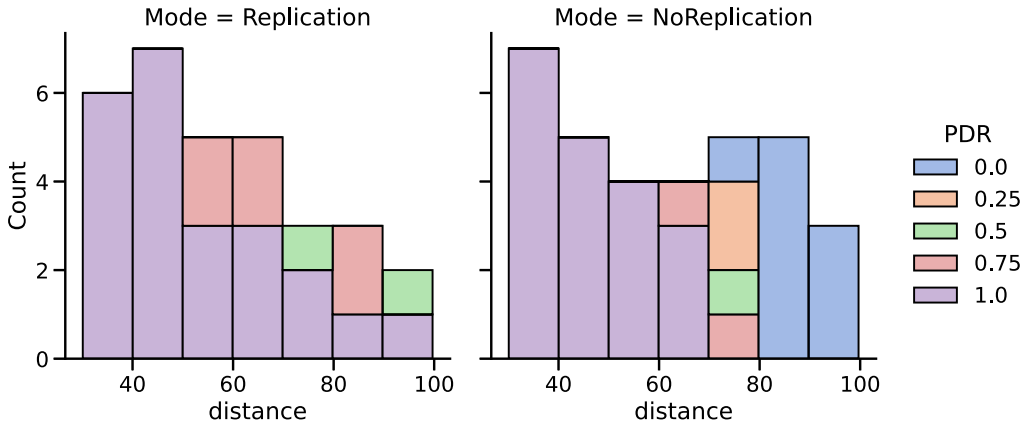


Fig. 21. PDR occurrence frequency vs Distance for Replication and no Replication scenarios.

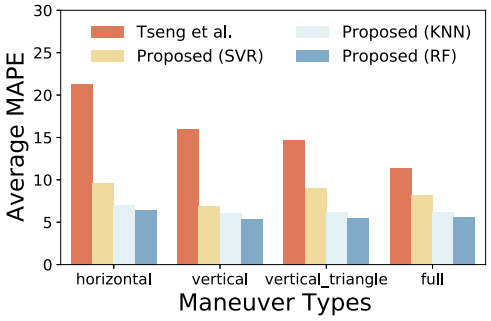


Fig. 22. MAPE of our proposed model with different regression algorithms.

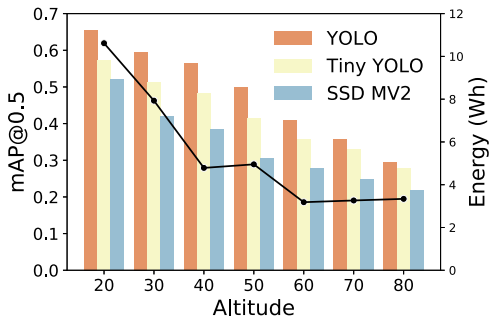


Fig. 23. Performance of the object detection models at different altitudes.

8 CASE STUDY: PEOPLE COUNTING

This section describes a people counting case study with our system.

Setup: We conducted this experiment using two drones mounted with a camera and an onboard computer (Jetson Nano). The application objective was to detect people walking on two adjacent sports grounds of area 106,563 and 112,601 *sq.ft.* in a university (anonymized). Our open-source ground control software (QGC with energy-aware path planning and scheduling) allocated flight paths to the drones to repeatedly cover each ground until the batteries are exhausted. Several students were asked to walk on the ground so that the drones could detect them. The application entailed continuous monitoring of the sports grounds until two batteries are exhausted by each of the two drones. In this case study, we had two new batteries (<10 charge-discharge cycles) and two older ones (>100 cycles). Each drone started the mission with a newer battery, and then we swapped it with older ones. The onboard computers have a pre-trained tinyYOLO object detector on the previously collected people detection dataset. As discussed in Section 2.6, we selected an altitude of 15 meters for the flight to have a high object detection accuracy (*F1 score* >0.8 in the training set).

Observations: The two drones covered a total distance of 10.358 km in 44.54 minutes and 9.73 km in 41.1 minutes, respectively. The first ground was covered 14.2 times, while the second was covered 15.75 times. Overall, the object detector identified people with a precision of 0.8 and a recall of 0.89. Overall, the mean average precision (mAP@0.5) of 83.65%.

9 RELATED WORK

9.1 Energy Modeling in Drones

Optimal usage of a drone's battery can significantly increase its flight time. Past works have looked at creating a theoretical or regression energy model for the drone as studied by [Zhang et al. 2021]. One of them, [Tseng et al. 2017], proposes a regression model of energy consumption for drones. [Di Franco and Buttazzo 2016] presents an energy model derived from real-world measurements and then uses it to derive the path planning algorithm. A recent work [Kim et al. 2020] proposes a methodology for SOC estimation based operation policy for a drone flight. However, these systems do not talk about energy-efficient task division among multiple drones across multiple charging events—a fundamental requirement in large-scale surveying tasks. Moreover, none of the existing work considers the interplay between energy consumption and sensing quality.

9.2 Drone Path Planning and Scheduling

Most approaches to solve drone path planning problem involve dividing the survey area into cells. Following which, a traveling salesman algorithm is applied to find the coverage path [Xie et al. 2019]. Furthermore, several algorithms find optimal sweep direction to minimize the number of turns while surveying an area [Araújo et al. 2013; Huang 2001]. All these works restrict themselves to only the geometrical aspects of the survey area and do not consider factors such as battery capacity, weight, and so on. Recently, an approach to energy-aware path planning was introduced [Franco and Buttazzo 2015]. However, the treatment was quite preliminary and did not consider factors such as wind, payload weight, and so on. Neither did it look at the impact of the plan on sensing accuarcy. There are also many powerful mission planning and GCS having various capabilities. QGround Control [Zurich 2013] is the most used open-source GCS having an ample amount of contributors on GitHub. UGCS [UGCS 2020] is closed-source but has some great features compared with open-source counterparts like multiple drone support even in simulation, 3D surveying, and so on. None of these have implemented energy-focused path planning and leave the choice of the mission flight altitude to the user.

9.3 Drone Orchestration Systems

Work in several areas (computer systems, robotics, etc.) has looked at developing drone orchestration, i.e., a system to manage a fleet of drones for a set of applications [He et al. 2020; Mahmoud et al. 2015; Mottola et al. 2014; Van't Hof and Nieh 2019]. AnDrone [Van't Hof and Nieh 2019] is a virtual drone computing platform that attempts to use drone-as-a-service to make it accessible like a cloud resource. Through this, a single physical drone can run multiple *virtual drones* instances for different applications simultaneously. Voltron [Mottola et al. 2014] presents a programming model to manage a fleet of drones using programming constructs. More recently, BeeCluster [He et al. 2020] looked at providing APIs to abstract complex drone tasks. However, none of these systems consider battery energy modeling (charging and discharging) and its implications in reducing mission times. Moreover, none of these focuses on the interplay between application objectives — i.e., model inference accuracy in surveying tasks— and the choice of altitude, with the resulting impact on the flight time for a mission and the battery drain. However, an interesting future direction would be integrating our energy-aware path planning and task scheduling into these drone orchestration systems.

10 CONCLUSION

Realizing the full potential of drones necessitates managing the limited energy resources carefully while maintaining sensing accuracy. In this article, we presented a holistic view on

drone scheduling by considering tradeoffs across several factors, such as battery characteristics, variation in drone imagery, and so on. We conducted an extensive empirical study and explored the impact of altitude and lighting on sensing accuracy. While uncovering several real-world battery characteristics, we built accurate models for learning charge and discharge profiles. For example, our energy model could predict energy consumed by various manoeuvres with an error of 5.6%—comfortably beating earlier state-of-the-art baseline. We use the insights from our empirical studies and the learned models to design our holistic, energy-aware, multi-drone scheduling system that reduces the energy consumed by 21.14% and time taken by 46.91%. We implemented our system as an open-source extension to a popular drone ground control software. We presented a people counting case study that performed with an accuracy of 83.65%.

REFERENCES

Federal Aviation Administration. 2016. Summary of Small Unmanned Aircraft Rule (Part 107). Retrieved from https://www.faa.gov/uas/media/Part_107_Summary.pdf. [Online; accessed 10-August-2019].

European Union Aviation Safety Agency. 2019. Civil Drones (Unmanned Aircraft). Retrieved from https://www.easa.europa.eu/easa-and-you/civil-drones-rpas. [Online; accessed 10-August-2019].

J. F. Araújo, P. B. Sujit, and J. B. Sousa. 2013. Multiple UAV area decomposition and coverage. In *2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. 30–37. DOI: https://doi.org/10.1109/CISDA.2013.6595424

Glenn Reesearch Center at National Aeronautics and Space Administration. 2019. Retrieved from https://www.grc.nasa.gov/www/k-12/airplane/drageq.htmls. [Online; accessed 10-August-2019].

Anirudh Badam, Ranveer Chandra, Jon Dutra, Anthony Ferrese, Steve Hodges, Pan Hu, Julia Meinershagen, Thomas Moscibroda, Bodhi Priyantha, and Evangelia Skiani. 2015. Software defined batteries. In *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 215–229.

Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.

Nicos Christofides. 1976. Worst-case analysis of a new heuristic for the travelling salesman problem.

Carmelo Di Franco and Giorgio Buttazzo. 2016. Coverage path planning for UAVs photogrammetry with energy and resolution constraints. *J. Intell. Robotics Syst.* 83, 3–4 (Sept. 2016), 44–462. DOI: https://doi.org/10.1007/s10846-016-0348-x

DJI. 2020. PHANTOM 4 PRO Specs. Retrieved from https://www.dji.com/sg/phantom-4-pro/info. [Online; accessed 10-January-2020].

DroneDeploy. 2020. Retrieved from https://www.dronedeploy.com.

Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. 2018. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 370–386.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.

Ramon Fontes, Samira Afzal, Samuel Brito, Mateus Santos, and Christian Esteve Rothenberg. 2015. Mininet-WiFi: Emulating software-defined wireless networks. In *2nd International Workshop on Management of SDN and NFV Systems, 2015(ManSDN/NFV 2015)*. Barcelona, Spain.

C. D. Franco and G. Buttazzo. 2015. Energy-aware coverage path planning of UAVs. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*. 111–117. DOI: https://doi.org/10.1109/ICARSC.2015.17

Andrew Gelman. 2006. Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian Analysis* 1, 3 (2006), 515–534.

gnss. 2021. "gnss". Retrieved from https://github.com/Aceinna/gnss-ins-sim. [gnss3].

Songtao He, Favyen Bastani, Arjun Balasingam, Karthik Gopalakrishna, Ziwen Jiang, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. 2020. BeeCluster: Drone orchestration via predictive optimization. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 299–311.

J. A. Hoogeveen. 1991. Analysis of christofides' heuristic: Some paths are more difficult than cycles. *Oper. Res. Lett.* 10, 5 (July 1991), 291–295. DOI: https://doi.org/10.1016/0167-6377(91)90016-I

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. Retrieved from https://arxiv.org/abs/cs/1704.04861

W. H. Huang. 2001. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, Vol. 1. 27–32 vol.1. DOI: https://doi.org/10.1109/ROBOT.2001.932525

J. Mark Keil and Jack Snoeyink. 1998. On the time bound for convex decomposition of simple polygons. Citeseer.

Majid Khonji, Mohammed Alshehhi, Chien-Ming Tseng, and Chi-Kin Chau. 2017. Autonomous inductive charging system for battery-operated electric drones. In *Proceedings of the Eighth International Conference on Future Energy Systems*. ACM, 322–327.

Jiwon Kim, Yonghun Choi, Seunghyeok Jeon, Jaeyun Kang, and Hojung Cha. 2020. Optrone: Maximizing performance and energy resources of drone batteries. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3931–3943.

Qi Li, Keyang Yu, and Dong Chen. 2020. Automatic damage detection on rooftop solar photovoltaic arrays. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 332–333.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*. Springer, 740–755.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*. Springer, 21–37.

LTC2944. 2017. LTC2944. Retrieved from https://www.analog.com/media/en/technical-documentation/data-sheets/2944fa.pdf.

Sara Mahmoud, Nader Mohamed, and Jameela Al-Jaroodi. 2015. Integrating uavs into the cloud using the concept of the web of things. *Journal of Robotics* 2015 (2015), 10–10.

Lorenz Meier, Dominik Honegger, and Marc Pollefeys. 2015. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 6235–6240.

Luca Mottola, Mattia Moretta, Kamin Whitehouse, and Carlo Ghezzi. 2014. Team-level programming of drone sensor networks. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. 177–190.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.

Angélica F. Resende, Maria T. F. Piedade, Yuri O. Feitosa, Victor Hugo F. Andrade, Susan E. Trumbore, Flávia M. Durgante, Maíra O Macedo, and Jochen Schöngart. 2020. Flood-pulse disturbances as a threat for long-living amazonian trees. *New Phytologist* 227, 6 (2020), 1790–1803.

GOLDMAN Sachs. 2017. Drones Reporting for Work. https://www.goldmansachs.com/intelligence/technology-driving-innovation/drones/

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4510–4520.

Richard Schiffman. 2014. *Drones Flying High as New Tool for Field Biologists*. American Association for the Advancement of Science, 459–459.

Mario Silvagni, Andrea Tonoli, Enrico Zenerino, and Marcello Chiaberge. 2017. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk* 8, 1 (2017), 18–33.

Milan Sonka, Vaclav Hlavac, and Roger Boyle. 1993. *Image Processing, Analysis, and Machine Vision*. Chapman and Hall.

Chien-Ming Tseng, Chi-Kin Chau, Khaled M. Elbassioni, and Majid Khonji. 2017. Flight tour planning with recharging optimization for battery-operated autonomous drones. *CoRR, abs/1703.10049*.

UGCS. 2020. Retrieved from https://www.ugcs.com/

Alexander Van't Hof and Jason Nieh. 2019. AnDrone: Virtual drone computing in the cloud. In *Proceedings of the Fourteenth EuroSys Conference 2019*. ACM, 6.

Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. 2017. Farmbeats: An iot platform for data-driven agriculture. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 515–529.

J. Xie, L. R. G. Carrillo, and L. Jin. 2019. An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems. *IEEE Control Systems Letters* 3, 1 (Jan 2019), 67–72.

Juan Zhang, James F. Campbell, Donald C. Sweeney I. I., and Andrea C. Hupman. 2021. Energy consumption models for delivery drones: A comparison and assessment. *Transportation Research Part D: Transport and Environment* 90, 90 (2021), 102668.

E. Zurich. 2013. Qgroundcontrol: Ground Control Station for Small Air Land Water Autonomous Unmanned Systems.

Received 3 May 2022; revised 13 April 2023; accepted 6 January 2024