EagleEye: Nanosatellite constellation design for high-coverage, high-resolution sensing

Zhuo Cheng, Bradley Denby, Kyle McCleary, Brandon Lucia

Carnegie Mellon University
{zhuoc2,bdenby,kmcclear,blucia}@andrew.cmu.edu

Abstract

Advances in nanosatellite technology and low launch costs have led to more Earth-observation satellites in low-Earth orbit. Prior work shows that satellite images are useful for geospatial analysis applications (e.g., ship detection, lake monitoring, and oil tank volume estimation). To maximize its value, a satellite constellation should achieve *high coverage* and provide *high-resolution* images of the targets. Existing homogeneous constellation designs cannot meet both requirements: a constellation with low-resolution cameras provides high coverage but only delivers low-resolution images; a constellation with high-resolution cameras images smaller geographic areas. We develop EagleEye, a novel mixed-resolution, leader-follower constellation design. The leader satellite has a low-resolution, high-coverage camera to detect targets with onboard image processing. The follower satellite(s), equipped with a high-resolution camera, receive commands from the leader and take high-resolution images of the targets. The leader must consider actuation time constraints when scheduling follower target acquisitions. Additionally, the leader must complete both target detection and follower scheduling in a limited time. We propose an ILP-based algorithm to schedule follower satellite target acquisition, based on positions identified by a leader satellite. We evaluate on four datasets and show that EagleEye achieves 11–194% more coverage compared to existing solutions.

CCS Concepts: • **Computer systems organization → Embedded systems**.

Keywords: orbital edge computing, nanosatellites, constellation design

ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA

ACM ISBN 979-8-4007-0372-0/24/04…$15.00
https://doi.org/10.1145/3617232.3624851

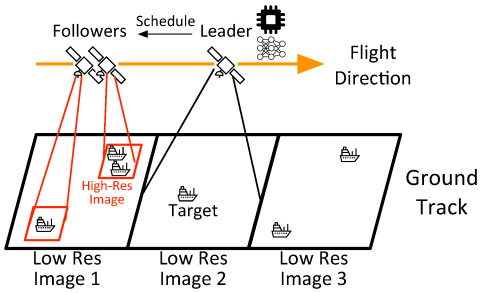
ACM Reference Format:

Zhuo Cheng, Bradley Denby, Kyle McCleary, Brandon Lucia. 2024. EagleEye: Nanosatellite constellation design for high-coverage, high-resolution sensing . In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3617232.3624851

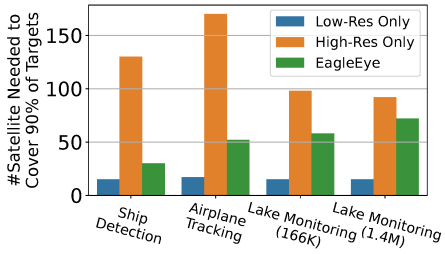
1 Introduction

The emergence of small, cheap nanosatellites — e.g., chipsats, pocketqubes, and cubesats [31, 46, 54, 64] — and the maturation of commercial space launch services [34] bring space within reach for a wide range of valuable, new space-based cyberphysical systems applications. These applications leverage the unique vantage point of a low-Earth orbit (LEO) satellite to capture spectral (i.e., visual, infrared) sensor data that are inaccessible on Earth. Low-cost, high-cadence launches allow the deployment of *constellations*: groups of satellites that work together to implement an application. Applications transform this data into valuable insights, such as optimizing transport [20, 26], enhancing agriculture [59], and supporting disaster relief efforts [36]. Applications typically sense a target area on Earth and process the sensor data using machine learning. Historically, all computation happens on Earth. Data centers process information downlinked by satellites that were individually and manually tasked with capturing and downlinking particular observations. This outdated operational model is a fundamental barrier to increasing the capability of future satellite systems, posing physical limitations and limiting operational autonomy.

LEO nanosatellites are limited by physical and operational constraints. Satellites are physically limited by the low communication bandwidth to the ground and the limited energy. LEO satellites are also constrained in their acquisition of sensor data. Applications want sensor coverage in large geographic areas to maximize the reach of their applications. Applications also want high-resolution data to maximize the quality of data delivered to analysts. Satellite cameras meet only one of these constraints: wide-area sensors capture low-resolution data (e.g., 100km per pixel) and high-resolution sensors capture data with a narrow-area view (e.g., a few square kilometers per image). These constraints present an



(a) *An EagleEye mixed-resolution leader-follower constellation.*



(b) *Across four applications, existing solutions fail to achieve either high coverage (90% coverage) or high resolution at a reasonable constellation size, while EagleEye achieves all. Note that a "Low-Res Only" constellation does not deliver high-resolution data.*

Figure 1. *The EagleEye system model and its benefits.*

unsatisfying design choice for satellite application developers, forcing them to choose coverage or resolution, but not both. Today's satellite constellations are also limited by a lack of autonomy and are relying on manual tasking. A human operator identifies the targets and sends pointing commands. The inability to detect and prioritize tasks precludes dynamic and reactive applications.

Getting the human out of the loop and autonomously, dynamically identifying sensing targets from orbit is challenging. An autonomous constellation of satellites must detect targets using orbital edge computing [28–30]. Onboard computing requires careful system design balancing energy consumption with computational capability to avoid being bottlenecked by compute or energy collection time. Second, satellites in the constellation must collectively identify events of sufficient interest with low time and energy overhead. Today's constellations operate with little or no autonomy making collective decision-making about points of interest infeasible. Third, after identifying interesting events, satellites in the constellation must plan actuation actions to point their sensors at targets.

We develop EagleEye, a new nanosatellite constellation operating model that enables capturing *high-resolution data* with *high spatial coverage* and a *high degree of autonomy*. The goal of EagleEye is to identify points of interest with high coverage and to sense those points of interest with high resolution.

EagleEye organizes a constellation of satellites into a *mixed-resolution, leader-follower* as Fig. 1a illustrates. In the leader-follower model, a low-resolution, high-coverage leader satellite identifies points of interest by continuously processing each image using orbital edge computing [28, 30]. A co-orbital follower satellite trails the leader by a small distance and points its high-resolution (but low-coverage) sensor at the identified points of interest and captures them. EagleEye resolves the tension between sensor coverage and image resolution: instead of choosing between high coverage and high-resolution, the leader-follower model provides both with little increase in cost. Fig. 1b shows the impact of EagleEye's design, with *both* high coverage and high-resolution with far fewer satellites than existing high-resolution solutions.

The design of EagleEye requires solving two main problems in computational nanosatellite constellation design. The first problem is designing a leader satellite that uses orbital edge computing to identify targets in low-resolution data without introducing a new time or energy bottleneck compared to existing low-resolution systems. Moreover, the leader must perform inference on low-resolution data with high precision to avoid sending false sensing cues to followers. The second problem is designing a constellation in which leaders and followers collectively perform *actuation-aware scheduling* to point and capture targets. Actuation-aware scheduling is challenging because the leader must calculate a feasible actuation plan for followers that covers targets on a pointing trajectory. The scheduler must have a low overhead and account for followers' time and energy.

We develop and evaluate a prototype of a EagleEye constellation design. The system identifies targets using several ML object detection models. The leader runs actuation-aware scheduling for followers using a low-cost integer linear programming (ILP) formulation that models pointing time. We evaluate EagleEye for several applications — airplane tracking, ship tracking, and lake algal bloom detection — using an orbital edge computing simulator from prior work [30], and four publicly available datasets. Our evaluation shows that EagleEye achieves high-coverage and high-resolution with as much as 4.3× reduction in required constellations size.

To summarize, the main contributions of this work are:

- EagleEye, a new leader-follower mixed-resolution constellation organization and operating model for computational nanosatellite constellations.
- Actuation-aware scheduling and target clustering, which leverage onboard computing on low-resolution data to coordinate satellites to point at targets autonomously to capture high-resolution target data.
- An EagleEye prototype implementation including an ILP formulation of scheduling and several ML target detectors.

- A comprehensive evaluation showing that for four real-world Earth observation tasks, EagleEye improves coverage and image resolution, while reducing constellation cost and complexity.

2 Background and Motivation

We provide background on tasking in nanosatellite constellations and highlight key shortcomings of today's systems in dynamically capturing geo-distributed targets with high coverage and high resolution. We then quantitatively motivate EagleEye by showing the promise of autonomous, mixed-resolution, leader-follower constellations.

2.1 Achieving high-coverage, high-resolution sensing with a satellite constellation

More launches to low Earth orbit and the decreasing costs of nanosatellites foster a "new space race:" many satellite constellations monitor the planet for a multitude of Earth-observation tasks. Satellites collect Earth images for geospatial analysis, such as environmental and ecological monitoring, meteorology, and agriculture. Use cases for Earth observation abound. For example, images containing ships could be used to detect illegal fishing [8, 16], oil spills and bilge dumping [42, 50]. High-resolution images containing lakes could help detect algae blooms [20, 52]. Images containing oil storage tanks could be used to estimate total oil reserve volumes [10].

A constellation is a collection of satellites that work together to support an application. The scope of this work is nanosatellite constellations that have development and launch costs that are orders of magnitude lower than larger, "exquisite" [45] satellite designs. Designing a constellation to support an Earth-observation application requires defining an organization and operating model. A constellation organization entails defining the hardware and software composition of each satellite, the number of satellites in the constellation, and the mix of capabilities across a heterogeneous constellation. The operating model involves when and how satellites sense and process Earth-observation signals, and how satellites communicate with one another and with internet-connected receivers in the "ground segment." The organization also involves defining the orbit altitude and inclination into which satellites deploy.

Constellation Organization. Nanosatellite constellations provide a low-cost, low-complexity option for deploying large numbers of satellites. A cubesat uses commercial, off-the-shelf (*e.g.,* Planet [23, 32], NASA [4]) components for electronics and structure and has a small size (*e.g.,* 10 cm × 10 cm × 10 cm for a "1U" cubesat) with masses around 1 – 10 kg. Cubesats often deploy to LEO. Several commercial operators have deployed numerous LEO cubesat constellations [12, 15]. LEO spans an altitude less than 2,000 km and often around 400 – 700 km. Building and launching a cubesat is relatively inexpensive, with a cost of tens of thousands dollars [6]. The low cost of nanosatellites enables launching constellations of tens or hundreds with a similar capability while costing far less than monolithic satellites [45].

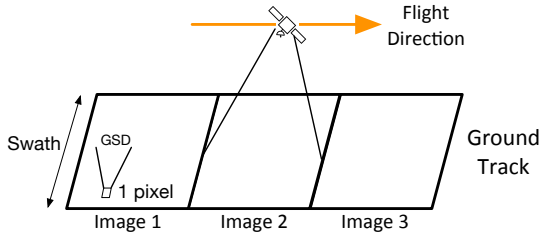
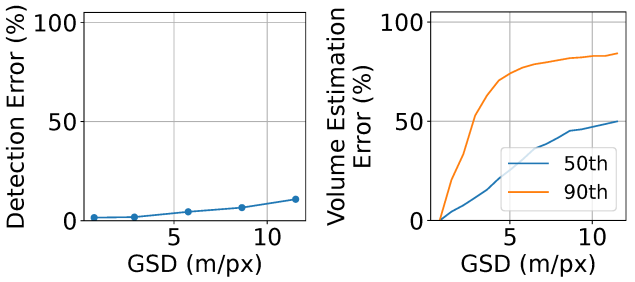


Figure 2. *A satellite's* ground track, swath *width (meters), and* ground sample distance *or GSD (meters/pixel).*

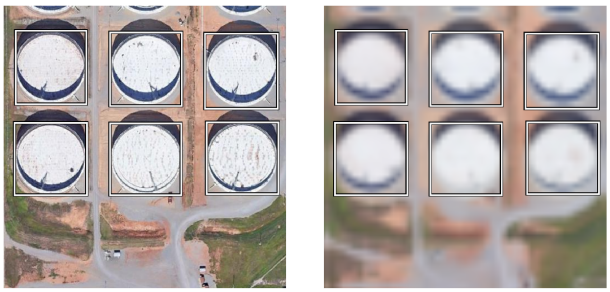
Satellite hardware includes several components. Satellites include an attitude determination and control system (ADACS) with actuators (*e.g.,* reaction wheels) to enable precise pointing at rates between 1 and 10 degrees per second. Onboard imagers capture electromagnetic spectral data inclusive of the visual domain and possibly other spectra, such as RF, near-infrared (NIR), and short-wave infrared (SWIR). In this work, we primarily assume visual spectrum data sensing, the sensors for which are common and have low cost; EagleEye applies generally to arbitrary spectrum data. The ground coverage area and ground sample distance or "GSD" (meters per pixel) of an image produced by a sensor are intrinsically defined by the camera system and the orbital altitude. A fundamental tension between coverage and GSD makes a key trade-off in constellation design at the heart of EagleEye. Existing LEO satellites with COTS imagers capture images of Earth with GSD of tens of centimeters to tens of kilometers per pixel. Fig. 2 illustrates the relationship between a nanosatellite's ground track, its camera's swath width, and its camera's GSD. A GPS/GNSS receiver [40] provides Earth-relative position information, allowing a satellite to perform precise geo-registration of captured sensor data. Recent work [30] showed that it is possible to deploy commodity computing devices (such as the NVidia Jetson/Orin mobile GPU) in a cubesat. High-performance computing hardware equips a nanosatellite to run sophisticated computations, such as image classification, object detection, and pixel segmentation.

Operating Model. Today, a vast majority of satellites operate with *no autonomy*. In this operating model, a human operator sends commands to each satellite in the constellation from a ground terminal. On receiving a command, a satellite senses based on GPS coordinates, may process data using orbital edge computing, and may transmit interesting data to Earth.

Recent constellation research proposes on-orbit computing to improve autonomy [28, 30]. In these operating models, a



(a) *Oil Tank Detection (stage 1) Accuracy* **(b)** *Oil Tank Volume Estimation (stage 2) Error*



(c) *High-Resolution image* **(d)** *10x lower resolution image*

Figure 3. *Oil Tank Volume Estimation Task*

constellation aims to cover its complete ground track. Covering the ground track is challenging with high-resolution data because an individual satellite may be unable to process an entire high-resolution frame before observing the next frame. Nanosatellite pipelining [30] covers the ground track by statically distributing the work among satellites and processing frame data in parallel. Kodan [28] uses ML model specialization, frame tiling, and tile processing elision to reduce the number of satellites required to cover a ground track (often to one satellite). These operating models improve over the command-oriented, human-in-the-loop model, but they do not address heterogeneous camera constellations.

2.2 Requirements

A satellite constellation design needs to meet several requirements: high image resolution (i.e., low GSD), high revisit rate (i.e., a small time interval between consecutive views of the same location), high coverage (i.e., a constellation covers a high fraction of earth area), and low cost (i.e., the total satellite count and orbit count is reasonable). Many of these design factors are coupled, and some are naturally negatively correlated. They all influence the cost and feasibility of a constellation deployment.

Data analytics have image resolution thresholds. Some applications require high-resolution images to produce accurate results. We characterize the impact of image resolution with a visual oil volume estimation task [11, 61]. The task consists of two stages: (1) detecting the oil tanks from the images; (2) estimating oil volumes based on the shadow size

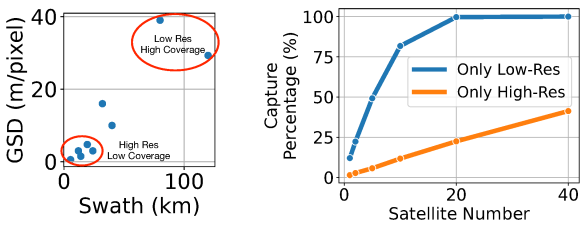


Figure 4. *Geospatial coverage and data quality are determined by a satellite's camera's swath width and image resolution. Left: GSD vs. Swath for nine real nanosatellite cameras. Right: Existing systems cannot provide high coverage with high-resolution data.*

on the tank lids. Fig. 3 shows the result of running the task on image data at different GSD levels ranging from 0.7 to 11.5 m/px. The results show that the low-resolution image is sufficient to correctly detect the oil tank, but does not contain enough detail to accurately estimate the volume of the oil tank. The key point is that higher resolution images are needed for some applications.

Geo-distributed targets require high geospatial coverage. An application's sensing targets may be highly geographically distributed, and in motion (e.g., airplanes or ships). To detect such targets, a fixed-size constellation's satellites' sensors must capture images of large geographic regions. With a fixed image sensor size (i.e., total sensed pixel count) a larger area per image entails a larger geographic area per pixel, and higher resolution images cover a smaller geographic area. Camera focal length and orbit altitude largely determine these parameters and cameras that can feasibly be deployed to space at low cost are restricted to a single operating point. An operator must then choose: high-resolution data or high-coverage?

We characterized this unsatisfying tradeoff between data quality and coverage. Fig. 2 shows image GSD versus swath (i.e., ground track width). A large swath operating point uses a short focal length or high altitude to cover a larger area, but at low resolution. A high resolution operating point uses a long focal length or low altitude that captures low-GSD (high-resolution) images, but only at a narrow swath. Fig. 4 (Left) contrasts several existing cubesat cameras (Planet [12], Dragonfly [3], Simera Sense [13]). Fig. 4 (Right) shows the fraction of targets that a constellation captures over a one-day period using different camera swath-widths (we describe our experimental setup in detail in §5.2). Using a large 100 km swath requires only 20 satellites to capture all targets, although only at an unacceptably low resolution. Using a small 10 km swath provides acceptably high-resolution images, but unfortunately, even 40 satellites capture only 41% of the targets.

Constellation size defines total cost. The material cost of a computational nanosatellite constellation scales directly

with the satellite count. Satellite cost has several components. Material cost for a nanosatellite is low, especially if equipped with only COTS components (*e.g.,* a COTS GPU is around $2k). Launch costs are by far the largest cost associated with a nanosatellite (*e.g.,* around $50-100k for a 3U cubesat) and must be amortized across many satellites. Moreover, launch costs manifest as a non-linearity as cost varies with constellation size: if the addition of a satellite requires an additional launch, the cost of the second launch amortizes poorly unless still more satellites are added. Operations costs are moderate and scale with satellite count. The primary operation cost is the ground station receiver operation cost, which is being commoditized [1, 21], but still scales with constellation size and data payload.

2.3 Existing Solutions

Tip and Cue. One existing solution uses a "tip and cue" operating model [7, 18, 58], where satellites from different missions with different cameras are used. This approach utilizes a low-resolution camera satellite for target detection and a high-resolution camera satellite for capturing high-resolution images of the targets.

However, this solution has several limitations. First, it requires operators from different missions (entities) to share compute, communication, and sensing resources, posing practical challenges with respect to economics and regulations. Second, the satellites fly in different orbits, leading to long delays (around 12 hours for one deployment [18]) between target identification and high-resolution imaging, which prevents imaging moving targets (*e.g.,* airplanes and ships). Third, in our best reading of the somewhat scant details of these systems, they lack operational autonomy: the target detection satellite sends images to Earth for processing, and the ground segment relays imaging commands to the high-resolution satellite.

AB&B [27] solves the limitations by proposing a bi-satellite cluster where two satellites (a low-resolution camera leader satellite and a high-resolution camera follower satellite) fly in the same orbit, with a separation of 100 s. This addresses the challenges related to resource sharing and image capture delays. To achieve autonomous operation, it runs the target identification and the high-resolution image capture schedule on the leader satellite.

However, AB&B still has several limitations. First, they use a custom branch-and-bound algorithm to schedule high-resolution image capture, which exhibits a high runtime. As shown in §6, AB&B takes more than 15 s to schedule just 19 targets. This leads to difficulties in meeting frame deadlines and result in lower coverage. Second, they neglect satellite energy constraints, whereas cubesats have very limited energy. Their scheduler's extended runtime exacerbates the energy insufficiency concerns. Third, they only design a bi-satellite cluster and evaluate the coverage over a 500 km × 2000 km area, without considering how the constellation size affects the coverage in a larger area (*e.g.,* the entire Earth area is around 510 million km^2). Fourth, they only consider a single follower satellite. Although this suffices for some target densities, we show in §6 that multiple followers provide higher coverage for a high target density.

3 Design Overview

EagleEye is a new constellation organization and operating model that leverages orbital edge computing and constellation design to provide high-coverage, high-resolution data. The viability of EagleEye hinges on the recent maturation of orbital edge computing, effective crosslinking between LEO nanosatellites, and robotics advances that support agile pointing. The key ideas in EagleEye are (i) a mixed-resolution leader-follower constellation organization, (ii) actuation-aware scheduling for follower pointing, and (iii) target clustering to increase coverage.

3.1 Mixed-Resolution Leader-Follower Constellations

EagleEye leverages a heterogeneous, *mixed-resolution, leader-follower* constellation organization. Leader and follower satellites contain different compute and sensing hardware. Fig. 1a illustrates the EagleEye constellation organization. A leader satellite has a low-resolution imager and compute hardware that enables processing low-resolution frames with high performance. A follower satellite has a high-resolution imager and may or may not include high-performance, computational hardware. Fig. 5 shows how EagleEye constellations differ from existing work, allocating satellites in a constellation into heterogeneous leader-follower groups instead of tasking them homogeneously. Leaders and followers have radio equipment for cross-link [65, 66] and downlink communication. All satellites have ADACS and GPS/GNSS for precise pointing and attitude/orbit determination.

EagleEye defines an operating model for a leader-follower constellation. Operationally, a leader satellite images its entire ground track, geo-registering each image with GPS coordinates. The leader processes each image using a pretrained ML model that identifies the targets in the image. A constellation has one or more followers that collectively capture high-resolution images of all targets identified by the leader. The leader distributes the target imaging tasks to the followers. The leader first uses a crosslink to query the position and attitude of each follower. The leader then computes an *actuation-aware schedule* of image captures. The schedule is a series of pointing and capture actions that a follower should perform to image targets at high-resolution. The actuation-aware schedule involves all followers and, when feasible, covers all targets. The leader distributes to each follower its

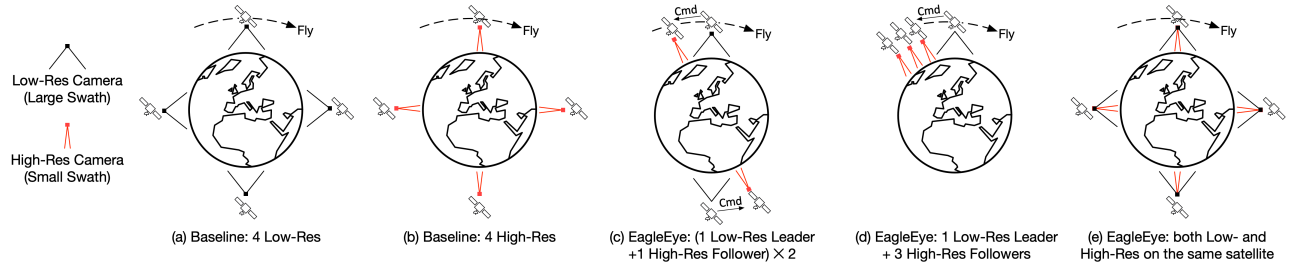


Figure 5. *Configuration comparison of a 4-satellite constellation.*

schedule of pointing and capture actions and each follower executes the schedule, capturing and storing the data. Eventually, follower satellites may perform additional onboard processing of the frames or may transmit the captured, high-resolution frames to Earth for consumption by a downstream application. The primary benefit of the EagleEye operating model is higher coverage at high resolution, because the high-resolution followers focus their sensing on areas with targets identified by the leader using low-resolution, high-coverage data.

3.2 Challenges of Target Detection & Sensor Scheduling

The goal of the constellation is to efficiently identify targets without exceeding time or energy limits on the leader, and at the same time maximize the number of targets captured by followers. At a high level, the scheduling algorithm achieves this goal by assigning a priority score to each target, based on the confidence with which it was detected, as reported in the output of the target identification ML model. The scheduling's optimization function is to maximize the sum of priority scores of targets captured by followers in a schedule. As for target detection by the leader, the targets must be large enough for the leader's low-resolution camera to observe.

Challenge 1: Actuation-aware scheduling. The pointing and imaging schedule that the leader produces for each follower must take into account the follower's position, attitude, and actuation constraints. For each target assigned to a follower, there is a window of time during which that target is available for imaging. The window is defined by the maximum "off-nadir" pointing angle, as shown in Fig. 6 (Left). Once the satellite's pointing angle exceeds this maximum threshold, the captured images become excessively distorted, rendering them unusable. Also, the schedule should consider the pointing actuation time as shown in Fig. 6 (Right), which limits the number of targets a follower can capture.

Challenge 2: Limited target detection and scheduling time. Time and energy are the primary limiting factors in a computational nanosatellite [30]. For full ground track coverage, the leader must capture each completely new frame that it observes, which implies a capture cadence of *e.g.,* 15s

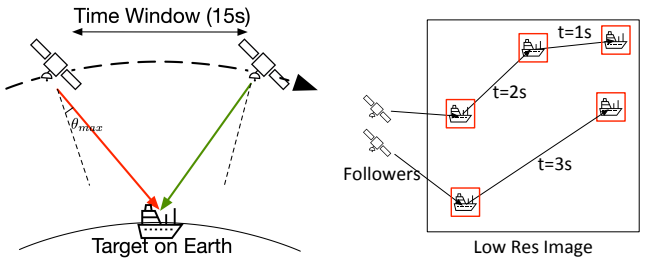


Figure 6. *Actuation-aware scheduling needs to consider: 1. (Left) each target's imaging time window, depending on each satellite's maximum off-nadir imaging angle θ_{max}; 2. (Right) the actuation time required to point between targets.*

at 500km with a 100km swath. The leader has 15s to complete target object detection and scheduling. Amplifying the challenge, cubesats usually have limited computing power in embedded CPUs and GPUs. Besides, cubesats get limited energy through solar panel and a single solar panel could only support the satellite computer to run for a portion of its orbit.

While high-accuracy object detection within these time and energy constraints is challenging, recent work [28] provides software solutions to reduce ML execution time in an accuracy-aware manner. These advances make it possible for EagleEye to support actuation scheduling based on ML detection results.

3.3 Problem Formulation

This section describes our optimization problem formulation in more detail.

We consider that a leader identifies M targets in a low-resolution image and schedules N follower satellites to capture the targets. The input contains all follower satellites' initial locations, the camera pointing direction, and all targets' locations.

$$Input : (sloc_i(t=0), sp_i(t=0)) \quad \forall i \in \{1, 2, \ldots, N\}$$

$$(tloc_j, tval_j) \quad \forall j \in \{1, 2, \ldots, M\}$$

Variable	Description
$N \in \mathbb{N}$	number of follower satellites
$M \in \mathbb{N}$	number of targets
$sloc_i(t) \in \mathbb{R}^3$	location of follower satellite i at time t
$sp_i(t) \in \mathbb{R}^3$	pointing direction of follower satellite i at time t
$tloc_j \in \mathbb{R}^2$	location of target j on Earth surface
$tval_j \in \mathbb{R}$	value of target j
$q_i \in \mathbb{N}$	number of targets captured by follower satellite i
$(c_{(i,k)}, t_{(i,k)})$	index & time of k^{th} target capture of i^{th} follower

Table 1. *Variables used in Problem Formulation*

The output is the schedule for all follower satellites

$$Output : S_i = \{(c_{(i,1)}, t_{(i,1)}), \ldots, (c_{(i,q_i)}, t_{(i,q_i)})\}$$

where each tuple is the target index ($c_{(i,k)} \in \{1, 2, \ldots, M\},\ \forall k \in \{1, 2, \ldots, q_i\}$) and the corresponding capture time.

The goal is to maximize the sum value of all captured targets

$$Goal : \max \sum_{c \in Hit} tval_c$$

where $Hit = \bigcup_{i=1}^{N} \{c_{(i,1)}, \ldots, c_{(i,q_i)}\}$ is the index set of all captured targets. The union operation is used to remove duplicate targets.

There are three constraints. The first is the actuation constraint: for each follower satellite, the difference in the pointing angle between two consecutive captures should be less than the maximum rotation angle in the time interval

$$C1 : ||sp_i(t_{(i,k-1)}) - sp_i(t_{(i,k)})|| \leq MaxAng(t_{(i,k)} - t_{(i,k-1)})$$

where $||sp_i(t_1) - sp_i(t_2)||$ calculates the difference between two pointing angles, $MaxAng(t)$ calculates the maximum angle that a satellite can rotate in time interval t.

The second is the off-nadir angle (time window) constraint: the off-nadir angle for each capture should be less than the maximum off-nadir angle

$$C2 : OffNadir(sloc_i(t_{(i,k)}), sp_i(t_{(i,k)})) \leq \theta_{max}$$

where $OffNadir(sloc_i(t), sp_i(t))$ calculates the off-nadir angle based on the satellite's location and pointing direction.

The third is to ensure that the target is in the captured images

$$C3 : tloc_{c_{(i,k)}} \in Image(sloc_i(t_{(i,k)}), sp_i(t_{(i,k)}))$$

where $Image(sloc_i(t), sp_i(t))$ calculates the region of the Earth's surface covered by the satellite image based on the satellite's location and pointing direction.

4 System Design

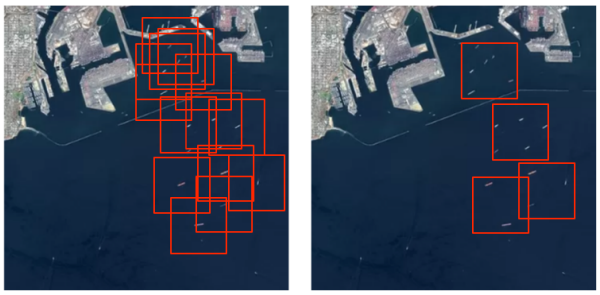
EagleEye improves the number of high-resolution targets downlinked with a mixed-resolution, leader-follower constellation. The target identification module finds targets by processing low-resolution imagery. The actuation-aware scheduler first analyzes targets, leverages actuation constraints, and produces a schedule for each follower to capture a sequence of targets. Each follower then adheres to its actuation schedule to capture the sequence of targets it is assigned.

4.1 Target Identification

The leader's target identification module leverages onboard ML inference. The input to the inference model is a low-resolution frame. For EagleEye, we assume that an object detection ML model identifies targets of interest. Thus, the model must be trained for the target application using previously collected and labeled orbital imagery. The targets of interest must be of appropriate size in the input tiles, and the total number of tiles per frame must not exceed to energy or time budget of the leader. The output of the model is a set of target bounding boxes associated with latitude/longitude GPS coordinates.

Frame tiling and scaling. As with prior work on orbital edge computing [28, 30], the target identification module first decomposes a (large) frame into tiles that fit the input dimension of the ML model, and then processes each tile. To identify small objects in low-resolution data with high accuracy, EagleEye decomposes a frame into the appropriate tile size (that may be smaller than the ML model's input), and then scaling the tile to the ML input size [28, 29]. The tile size with optimal accuracy is application- and system-dependent [29, 30] and recent work shows execution time benefits to operating away from the empirically-optimal tiling for accuracy [28]. Tile size is an important parameter because a decrease in tile size increases tile count, which increases the number of invocations of the ML model required to process a frame, ultimately increasing frame processing time. Changes to tile size also change the size (in pixel count) of the features of interest when input into the ML network. To avoid missing some data, a leader must finish processing each frame before it sees an entirely new frame, which imposes a hard deadline on frame processing. The number of tiles per frame must also avoid exhausting the leader's energy store. To validate EagleEye's use of frame tiling, §6 shows that across a wide range of tile sizes, frame processing time remains below the energy constraint and processing deadline.

Target clustering After identifying targets, the leader optimizes the set of targets by clustering targets that are close together, enabling them to be captured in a single, high-resolution image instead of requiring multiple captures. Fig. 7 shows the clustering problem schematically, with the center points of identified targets enclosed by a single box representing a high-resolution image capture.



(a) *w/o clustering* **(b)** *w/ clustering*

Figure 7. *Target clustering: Cover multiple targets with one high-resolution image. (Camera swath is changed for illustration purposes.)*

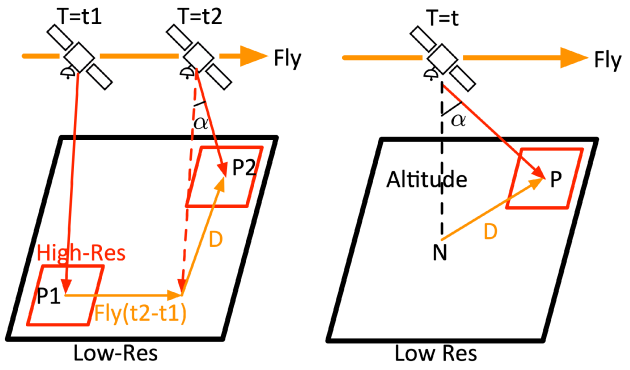


Figure 8. *Left: Calculate Actuation Time Between Targets; Right: Calculate Time Window.*

We formulate target clustering as a planar point cover problem: given several points on a 2D surface, find the minimal set of rectangles that cover all points, searching across all rectangle positions and dimensions. We simplify the problem by assuming that the sides of high-resolution images are in parallel with the sides of low-resolution images. Note that allowing off-parallel, high-resolution images may further reduce the total number of high-resolution images, but we leave this extension for future work.

We use Integer Linear Programming to solve the problem and the solution is a set of boxes enclosing clustered targets that can be imaged together in a single high-resolution image. We find that the solver could find the optimal rectangle cover solution in 1 ms for 500 targets (figure not shown). After clustering, the scheduler treats the priority score of a cluster's high-resolution frame to be the sum of the priority scores of the targets enclosed by the frame.

4.2 Computing Actuation Time and Target Time Window

EagleEye's actuation-aware scheduler computes an actuation plan for each follower to point at and capture each target. The plan must consider the time to point from one target to the next, and must consider the optimal path along which to capture all targets. The scheduler computes the plan as the solution to an Integer Linear Programming (ILP), which includes constraints corresponding to the physics of pointing and orbital motion, as well as the cost of visting targets in a particular order.

Computing Actuation Time As shown in Fig. 8 (Left), assume that the satellite points at P_1 when $T = t_1$, we want to compute the minimum time $T = t_2$ when the satellite points at P_2. A satellite's pointing speed is defined by its ADACS and actuators, limiting the angle that a satellite may rotate in a fixed time interval t to some number of degrees, $MaxAng(t)$. The motion planner finds the minimum t_2 that satisfies the following constraint:

$$\alpha = \frac{|\vec{D}|}{Altitude} = \frac{|\vec{P_2} - (\vec{P_1} + \overrightarrow{Fly(t_2 - t_1)})|}{Altitude} <= MaxAng(t2 - t1) \quad (1)$$

where $Fly(t_2 - t_1)$ is the motion vector of the satellite projected onto the ground. A solution gives a value to t_2, defining $t_2 - t_1$, which is the time over which to rotate by α to point at the target.

Setting time window constraints. As shown in Fig. 8 (Right), we want to find the interval of t that satisfies

$$\alpha = \frac{|\vec{D}|}{Altitude} = \frac{|\vec{P} - \overrightarrow{N(t)}|}{Altitude} <= \theta_{max} \quad (2)$$

where $\overrightarrow{N(t)}$ is the position of the satellite projected onto the ground at time t and θ_{max} is the maximum off-nadir angle.

4.3 Actuation scheduling

The scheduler solves the actuation scheduling problem to produce a sequence of pointing actuations for each follower. The leader adds constraints and optimization goals to the ILP formulation, framing it as a generalized traveling salesman problem that plans routes for multiple salesmen to cover multiple cities optimally. One set of ILP constraints define the actuation time as distance between targets. Another set of constraints relate these actuation times to the time window for each target, for each follower. The ILP optimization goal is to maximize the value of the sum of priority scores of captured targets. The solution to the ILP is a set of pointing actuations (*i.e.*, rotations) for each follower satellite to perform to cover its assigned set of targets.

Alternative formulations As a point of comparison, we construct another solution that uses a greedy algorithm to choose the nearest unimaged target. The greedy algorithm does not achieve an optimal result, with 4.3–14.4% less coverage compared to our ILP-based solution, as shown in §6.

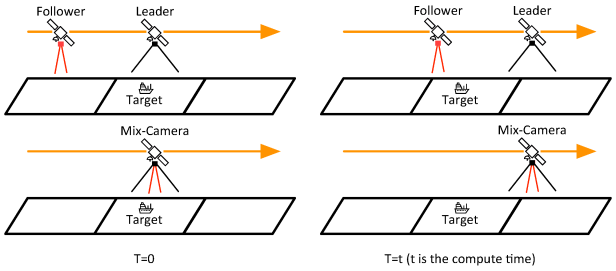


Figure 9. *Computation delay does not affect leader-follower, but affect mix-camera.*

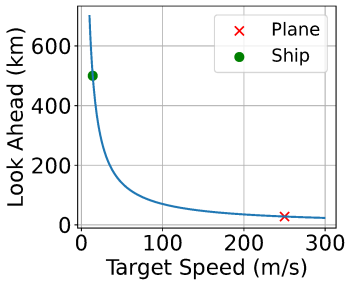


Figure 10. *Maximum look ahead distance for different target speeds.*

We also consider the AB&B approach used in prior work [27]. This approach exhibits high runtime, which limits its applicability (§6).

4.4 Number of followers

One configuration for EagleEye is the number of followers in a group. Fig. 5 shows several configurations of a constellation with 4 satellites. We can choose to have (i) 2 groups with one leader and one follower in each group (Fig. 5c) or (ii) 1 group with one leader and three followers in each group (Fig. 5d). With more groups, the total area covered by all leaders increases, detecting more targets. With more followers in a group, more targets in each low-resolution image would be captured by the followers.

An extreme configuration is to have both a high- and a low-resolution camera in the same satellite group (Fig. 5e). For a fixed number of satellites, this configuration would have the maximum number of groups. However, it has two limitations. First, the satellite might miss some targets due to computation delay. The satellite has only about 15 seconds to capture, process, point to take high-resolution images of targets, and point back to take another low-resolution image of the ground track. As shown in Fig. 9 bottom, the mix-camera satellite might already fly away from the target when the computation is completed. We show how the compute time would affect the coverage in §6. On the other hand, the computation delay does not affect the leader-follower configuration (Fig. 9 top), as long as the computation is finished before the next low-resolution image comes in, because the follower satellites are behind the leader satellite. Second, a cubesat is highly volume constrained [29], mostly due to the high-resolution camera's need for long focal length. It is likely infeasible to add a second high-coverage camera without increasing volume (and hence launch cost).

4.5 ML accuracy

We consider how the accuracy (precision, recall) of the object detection model running in the leader would affect coverage. A false positive increases the complexity of scheduling and potentially wastes time and energy of followers that could be used to image real targets. A false negative target is omitted from the scheduling and remains unimaged by any followers, thereby resulting in reduced coverage.

4.6 Scheduling for Moving Targets.

For a moving target (*e.g.,* a ship or a plane), EagleEye needs that the target does not move out of the targeted image during the time interval between the leader and the follower taking the images. Consider that the satellite ground speed is V_{sat}, the target speed is V_{target}, the high-resolution image swath is $swath$, and the distance along the ground track between when the leader captures an image and when the follower does is D (which we call the "lookahead distance"). γ is a slack parameter indicating the fraction of a high-resolution image swath over which a target may move and still be considered in the frame. We relate the lookahead distance to target and satellite velocity, considering the slack: $\frac{D}{V_{sat}} * V_{target} \leq \gamma * swath$. Fig. 10 shows the maximum look ahead distance D for different target speeds, assuming a satellite at 500 km altitude with $V_{sat} = 7.5$ km/s, $swath = 10$ km, $\gamma = 0.1$. We show that the maximum lookahead distance is $500km$ for a ship with ground speed 14 m/s (50 km/h) and 28 km for a plane with ground speed 250 m/s (900 km/h). The data shows that a reasonable lookahead distance accommodates reasonable target speeds. Note that we assume V_{target} represents the maximum target speed. EagleEye will have more time slack if the target moves slower than the expected target speed.

4.7 Discuss

Heterogeneous configurations overhead There is a non-recurring engineering (NRE) cost associated with supporting two camera types. Integrating different cameras does not require substantial changes to highly-coupled subsystems, like communications hardware or ADACS, and we expect the time cost to be surmountable.

Reliability There could be satellite failure or network partition between satellites. If a leader fails, the followers in

the group could fall back and simply capture nadir high-resolution images. Alternatively, the followers might be reassigned to another group under a new leader. If a follower fails, the leader can adjust by scheduling only with operational followers.

Recapture One future work is to explore target recapture (re-identification) between satellites. If the leader satellite identifies a target that has already been imaged by another follower previously, the leader can deprioritize the target and schedule the follower to capture other targets first. Alternatively, if the leader satellite identifies a target that has changed over time (*e.g.,* a ship has moved), the leader can prioritize the target when scheduling.

Orbit Design As the constellation grows in size, there will be more overlapping regions captured by different satellites in a short period. To further improve coverage, we need to strategically distribute the constellation to reduce these overlapping areas. This requires adjustments to the constellation's orbital parameters, such as orbit inclination and the number of orbital planes. We consider this to be future work.

5 Implementation and Methodology

5.1 Prototype

We implement a prototype of EagleEye using state-of-the-art orbital edge computing modeling tools (`cote` [30]) to model orbital dynamics, image capture, and ADACS pointing. We use Google-ORTools [53] to define our ILP formulations for target clustering and follower scheduling and find the solution. We use our prototype to evaluate EagleEye for a range of real-world use cases, taking advantage of publicly available Earth imagery and remote sensing datasets.

5.2 Use Cases

We study four real-world, remote-sensing applications. Labeled datasets for Earth observation are challenging to obtain. In some cases we are unable to obtain location data and usable imagery. When one type of data was unavailable, we evaluated EagleEye using the data to the degree possible, given availability; we call out such cases explicitly below.

Ship Detection We evaluate an application that identifies ships to detect illegal fishing and oil spills. We obtain the ship locations from prior work [5], which maps 19,119 ships around the world. The dataset does not model ship movement, so we evaluate on a snapshot. We use ship imagery from prior work [26], which has 3,896 images containing 3,219 ships, with 16m GSD, from the GaoFen-1 and GaoFen-6 satellites. We downsample the images to get a GSD of 30m. We train Yolov8 [41] on these images for object detection, achieving a mAP@50 (mean average precision calculated at IOU threshold 0.5) of 77.6%.

Airplane Tracking We evaluate an application that tracks airplanes for airspace safety. We use plane location data from Spire [15] that tracks the location of 55,196 planes across the world over 24 hours. The dataset models the plane movement with time. We are unable to obtain a usable corresponding imagery dataset, but the location dataset suffices to evaluate EagleEye's scheduler.

Lake Monitoring We evaluate an application that captures images of small lakes to detect emergent algae blooms. We use an existing lake location dataset [47]. We consider two scenarios that omit some lakes based on size: (1) 166,588 lakes with a size between $1 - 10$ km^2; (2) 1,410,999 lakes with a size between $0.1 - 10$ km^2. The lake dataset lacks a usable imagery companion, and we omit evaluation of ML inference on these data.

Oil Tank Volume Estimation We evaluate an application that estimates oil tank contents by measuring the size of shadows on their covers that vary with level. We get oil tank image data from [10], which has 10,000 images from Google Earth, with GSD (0.72 m/pixel). We train Yolov8 to detect the location of the oil tank and run the code from [11] on the images to estimate the oil volume. The dataset does not provide ground truth data for the fill-level estimation, so we assume that volume estimates based on high-resolution images are 100% accurate, which is consistent with other prior work [61] that used a similar method to achieve 97.2% accuracy. The dataset does not include the geographic distribution of oil tanks in each image. We thus use the oil tank dataset to evaluate the accuracy of ML inference for tank detection and fill level estimation, but we are unable to use this dataset for scheduling evaluation.

5.3 Satellite Parameters

We use low-cost nanosatellites for both the leader and the follower. The high-resolution camera has a 10 km swath at 3m GSD, and the low-resolution camera has a 100 km swath at 30m GSD. We consider a 11° maximum "off-nadir" pointing angle (θ_{max}). The ADACS can rotate the satellite 3 deg/s [14, 17, 35, 55]. We add a pointing acceleration/deceleration of 9 deg/s^2 [35], modeled by adding 0.67 s overhead to each point action *i.e.,*, $MaxAng(t) = 3 * (t - 0.67)$ deg/s. We also model a high-end reaction wheel [19] with a rotation rate of 10 deg/s (see §6). We use satellite "two-line elements" (TLEs) from Celestrak [2] to model a polar orbit, with an inclination of 97.2°, altitude of 475 km, and an orbit period of 94 minutes. All satellites fly in the same orbit; deploying to multiple orbits incurs additional launch costs. We model the energy consumption of a 3U CubeSat using the same parameters as in prior work [30]. We assume that all cubesat data processing occurs using a NVidia Jetson AGX Orin [9] running in its low-power (15W) operating mode. Each satellite can transmit to ground stations for six minutes each period to downlink data. For cross-links, the leader sends schedule data to each follower, conveying time and pointing direction for each high-resolution image capture. Each schedule result

is under 2KB and the leader sends around 400 schedule results every period; cross-link data volume is negligible, totaling under 1MB / orbit, which is easily accommodated by an S-Band radio's 0.4 MB/s. The distance between a leader and its followers is 100km, which is the low-resolution swath width, and we assume that leader-follower groups are evenly spaced in an orbit. For all experiments, we model 24h of activity.

5.4 Baseline Systems

We consider two baselines that represent the state-of-the-art in orbital edge computing constellation organizations: Low-Res Only and High-Res Only. These constellations consist of satellites that capture data at a single resolution only.

6 Results

We evaluate EagleEye to show several key results:

1. With the same number of satellites, EagleEye produces high-resolution data while achieving 11–194% more coverage than a High-Res Only constellation.
2. EagleEye's target clustering and follower scheduling is fast, finishing in 10 ms on an embedded CPU.
3. EagleEye achieves higher coverage with a faster slew rate or with more followers.
4. EagleEye has better energy consumption than either Low-Res Only or High-Res Only, avoiding the introduction of any new energy limitation on the system.

6.1 End-to-end results

The main end-to-end results in our evaluation are that EagleEye improves coverage substantially and EagleEye imposes a low run time overhead.

Coverage We consider coverage as the percentage of targets captured in high-resolution images. EagleEye substantially improves coverage (11–194%) for all applications compared to High-Res Only as shown in Fig. 11a. ILP-based scheduling achieves 4.3–14.4% higher coverage than greedy scheduling algorithm. We also include the results of Low-Res Only to show the maximum potential coverage (physical limits) of using low-resolution cameras. Note that for Airplane Tracking, Low-Res Only converges to 80% coverage. This is because the targets are moving and some targets only appear in the later period of the simulation, making them impossible to capture.

Across four use cases, EagleEye's improvement over High-Res Only is higher in applications with a lower target density (*i.e.,* Ship Detection, Airplane Tracking) than in applications with a higher target density (*i.e.,* Lake Monitoring (1.4M)). The trend exists because the single follower used in these experiments is unable to capture all detected targets.

Runtime Fig. 12a shows that ILP based scheduling has negligible runtime overhead and is scalable, while prior solution (AB&B) is not. Finding a solution for the ILP scheduler takes about 1ms even with a high target count. On the other hand, the runtime of the AB&B based scheduling fails to meet the frame capture deadline with more than 19 targets in a low-resolution images. Fig. 12b shows that up to 32% images contain more than 19 targets, so AB&B is infeasible for a real-world deployment. We also measure the runtime of target clustering and find that the 90th percentile runtime is less than 1ms (figure not shown).

6.2 Sensitivity and Characterization

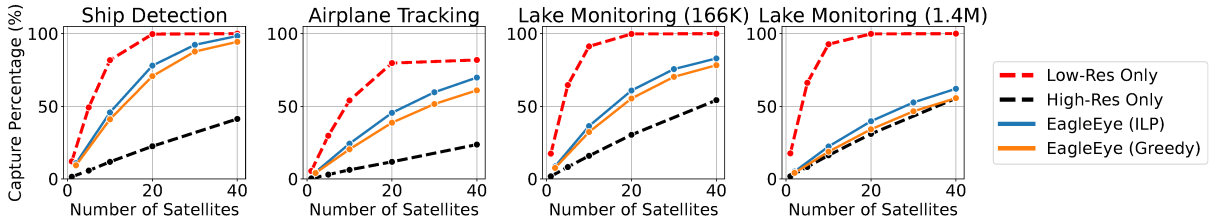
We study EagleEye's sensitivity to key design parameters.

Slew rate Fig. 11b shows that a faster slew rate improves coverage. With low target density (Ship Detection, Airplane Tracking), increasing the slew rate from 3 deg/s to 10 deg/s only marginally improves coverage. The small change indicates that a constellation designed for low target density may safely use a lower-cost ADACS with a lower slew rate. For Lake Monitoring (1.4M), EagleEye with slew rate = 1 deg/s achieves lower coverage than High-Res Only because the density of the targets is such that nadir pointing a high-resolution imager effectively captures the targets, and off-nadir pointing imposes a time cost that reduces the number of targets captured.

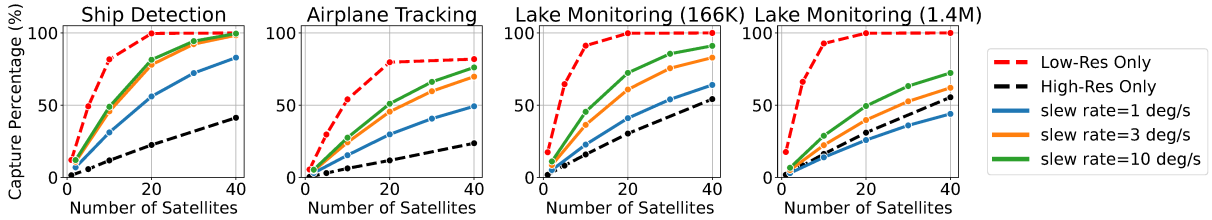
Number of Followers Fig. 11c shows how EagleEye' coverage varies with different numbers of followers. With low target density (*i.e.,* Ship Detection, Airplane Tracking), EagleEye with 1 follower achieves higher coverage than EagleEye with 3 follower. This apparent inversion is because one follower covers all targets detected by each group's leader. Given a fixed number of satellites in the constellation, it would be more beneficial to increase the number of groups than to increase the number of followers per group. For targets with high density (*i.e.,* Lake Monitoring (1.4M)), covering all targets requires more followers.

Fig. 13 shows that mix camera configuration achieves a lower coverage compared to the leader follower configuration due to the compute time delay. The coverage becomes lower as the compute time increases (with larger model) because there is less time left for target pointing and capture. Mix Camera with Yolo_x achieves 0% coverage because all targets are no longer in the time window when compute is finished.

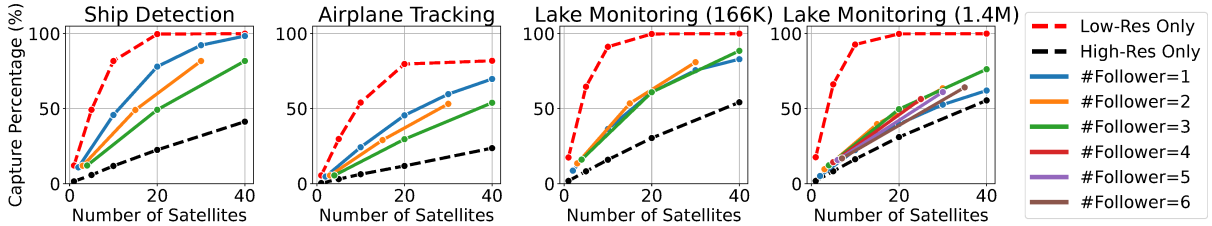
Target Miss Ratio We examine how many targets in a low-resolution image can be covered by one follower. As shown in Fig. 14a, when there are fewer than 10 targets in a low-resolution image, one follower could cover all of them. This also explains why EagleEye (#follower=1) has highest coverage on low target density applications, while more followers are required for high target density applications. Note that there would be at most 100 targets in one low-resolution image after target clustering, because the ratio of low-resolution and high-resolution camera swath is 10.



(a) EagleEye scheduling improves coverage. *EagleEye uses 1 follower and slew rate = 3* deg/s *configuration.*

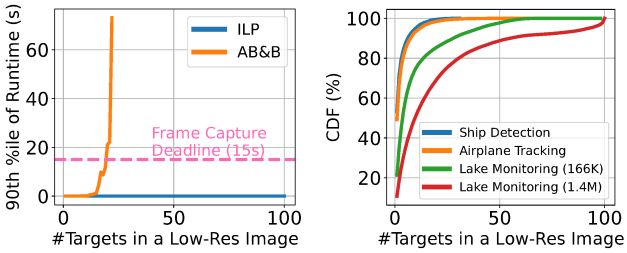


(b) Faster slewing further improves coverage. *(EagleEye uses 1 follower and ILP scheduling.) With a high target density (Lake Monitoring (1.4M)), the slew rate = 1* deg/s *is too slow and EagleEye covers fewer targets than High-Res Only.*



(c) Adding followers improves coverage with high target density. *(Lake Monitoring (1.4M)). For applications with low target density (Ship Detection, Airplane Tracking), using one follower is the most efficient way to improve coverage.*

Figure 11. EagleEye improves coverage compared to several baseline systems across all applications. *System improvements to slewing and follower count increase EagleEye's benefits.*



(a) **(b)**

Figure 12. *(a) EagleEye's runtime is low and insensitive to target count. (b) # Targets per image.*

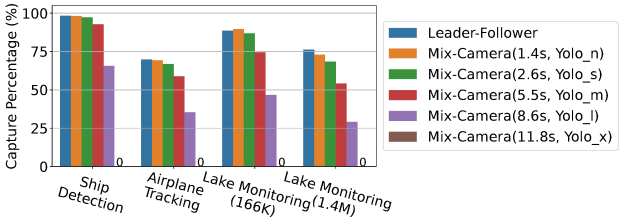
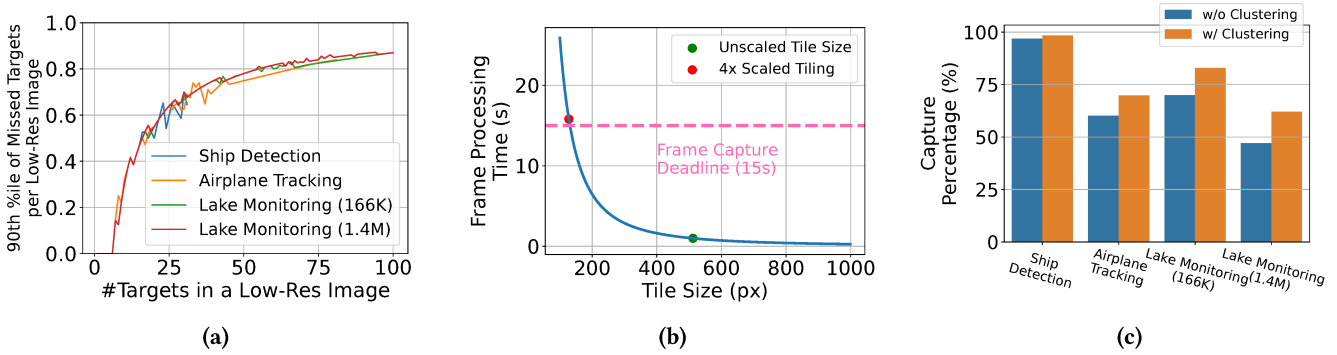


Figure 13. *Mix camera configuration achieves a lower coverage compared to the leader follower configuration due to the compute time delay. The numbers in parentheses are the compute time.*

Frame processing time Fig. 14b shows that in a wide range of tile sizes, frame processing time remains below the processing deadline. Low-resolution data in our orbit nets a longer frame deadline than prior work [29, 30]. We assume that fewer tiles per frame suffice to produce accurate targets for the followers.

ML Accuracy We investigate the impact of the object detection model's recall rate on coverage. Depending on the recall rate, a subset of targets are randomly chosen and labeled as False Negative. We do not analyze the effect of precision on coverage, as determining the location of False Positiveinstances requires running the model on the images at different locations, which we leave for future work.

Fig. 15 shows that coverage decreases in a slower rate when recall decreases, *e.g.*,, coverage is more than 0.5 when the recall rate decreases to 0.2, which means that some targets are not detected by the leader but are still captured by the follower. This is because the follower image swath is large



(a) (b) (c)

Figure 14. *(a) 1 follower could not cover all targets when there are more than 10 targets in a low-resolution image. (using 1 follower and slew rate = 3deg/s configuration). (b) Frame processing time with different tile sizes. (c) Target clustering improves coverage by* 1.5–31.7%.

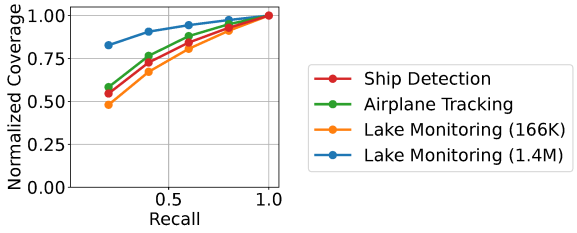


Figure 15. *Coverage decreases in a slower rate when recall decreases.*

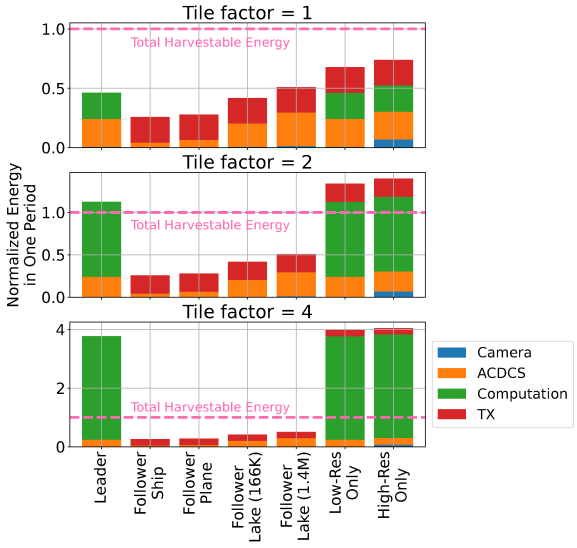


Figure 16. *Energy Analysis*

and might contain multiple targets. When a follower takes a high-resolution image, the image may contain some false negative targets.

Target clustering improves coverage Fig. 14c shows that by using target clustering, EagleEye achieves a higher (1.5–31.7%) coverage. Target clustering provides more gains with high target density (*e.g.*,, Lake Monitoring).

Energy Analysis Fig. 16 shows the energy usage of EagleEye and the baselines. For different applications, Leader and baselines would consume the same amount of energy, because they take images along the ground track and process all images, no matter the target density. Followers would consume different amounts of energy for different applications, depending on how many targets they are scheduled to capture.

The total harvestable energy per orbit supports around 2× tiling for EagleEye. For 4× tiling, the leader has the same problem as the baselines: the excess compute energy requirement precludes full ground track coverage. The leader uses a slightly less energy than the baselines use, because it does not send images to Earth, offloading the energy cost onto the followers, which are less energy constrained. The figure also shows that for all tiling factors and applications, energy is not a bottleneck for followers. Combining the energy and slew rate results, we provide guidance for constellation designers: add solar panels to the leader, providing more compute energy and improve the follower's ADACS slew rate to cover more targets.

7 Related Work

Satellites Scheduling: Many works [27, 38, 39, 49, 62, 63] consider scheduling satellites to point their sensors and maximize the number or value of targets observed. However, these works assume that the target locations are known beforehand and focus on offline scheduling. AB&B [27] considers online scheduling but has several limitations as mentioned in §2.3. Other work [24, 25, 57] focus on a simplified problem, excluding the modeling of actuation time between targets due to their assumption that the sensor is a radar with instantaneous electronic movement capabilities.

Heterogeneous Satellite Constellations: Existing leader-follower designs [7, 18, 58] require offline scheduling and cannot be applied to moving targets, as mentioned in §2.3. Another work [56] aims at a different problem: decreasing the image capture latency for rapid response to disasters.

This work uses two constellations in two different orbit planes: one for imaging and the other as a communication relay between the imaging satellites and Earth.

Superresolution. Superresolution techniques [33, 44] offer a promising means of enhancing image resolution through statistical refinement and data synthesis. However, applying them directly to low-resolution satellite images may not align with our objectives, as it could introduce misleading artifacts for analysts requiring high-fidelity data. On the other hand, super-resolution running on leaders (if made cheap enough) may improve target identification accuracy.

Scheduling in other domains: Recent work has studied architectural and system design techniques for robots [22, 48, 51] and autonomous drones [37, 43]. This work is related to EagleEye because we also consider physical and operational constraints in our system design. However, these efforts are distinct in purpose and mechanism because they aim to optimize hardware architecture for better performance and energy efficiency. Also, satellites differ from robots and drones because they have less freedom of movement: a satellite trajectory is fixed (along the orbit) after launching, and the only thing that can be changed is the satellite orientation.

Other aspects of satellite research: Kodan [28] solves the in-orbit computation bottleneck for image processing by training specialized ML models for different geospatial contexts. L2D2 [60] reduces satellite-ground communication latency by adding more low-cost commodity hardware-based ground stations to increase the density of ground stations.

8 Conclusion

EagleEye introduces the leader-follower, mixed-resolution constellation organization and operating model for computational nanosatellite constellations. Leveraging heterogeneity enables an EagleEye constellation to provide coverage similar to a wide-angle, low-resolution imaging constellation, while providing high-resolution images. The novel target clustering technique and scheduling algorithm make it possible for a leader satellite to task followers with a set of actuations, providing an increase in constellation autonomy and eliminating the need for human tasking interventions. Overall, EagleEye provides a reduction of up to 4.3× in the constellation size required to provide high-resolution images, with gains that translate across several application domains.

Acknowledgments We thank the anonymous reviewers and our shepherd Vijay Janapa Reddi for their valuable feedback. We thank Yoshiki Takashima and Atul Bansal for their comments on earlier drafts of this paper. This work was supported in part by National Science Foundation Cyber-Physical Systems Frontier Award #2111751.

References

[1] Azure Space - cloud-powered innovation on and off the planet - The Official Microsoft Blog. https://blogs.microsoft.com/blog/2020/10/20/azure-space-cloud-powered-innovation-on-and-off-the-planet.
[2] Celestrak: Current gp element sets. https://celestrak.org/NORAD/elements/.
[3] Cubesat cameras - dragonfly aerospace. https://dragonflyaerospace.com/products/.
[4] First-ever mars flight | snapdragon insiders | qualcomm. https://www.qualcomm.com/snapdragoninsiders/news-forums/snapdragon-insiders-get-access-qualcomm-expert-kim-koro-learn-about.
[5] Global fishing watch | data download portal. https://globalfishingwatch.org/data-download/.
[6] HOW MUCH DOES IT COST TO LAUNCH A CUBESAT? https://brightascension.com/how-much-does-it-cost-to-launch-a-cubesat/.
[7] Issue 60: Tip & cue. january 26, 2023 | by planet snapshots | medium. https://medium.com/@planetsnapshots/issue-60-tip-cue-4d9deeaa7988.
[8] Iuu | illegal, unreported, unregulated fishing | global fishing watch. https://globalfishingwatch.org/fisheries/iuu-illegal-unreported-unregulated-fishing/.
[9] Jetson orin for next-gen robotics | nvidia. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.
[10] Oil storage tanks | kaggle. https://www.kaggle.com/datasets/towardsentropy/oil-storage-tanks.
[11] Oil tank volume estimation | kaggle. https://www.kaggle.com/code/towardsentropy/oil-tank-volume-estimation/notebook.
[12] Planet satellite. https://www.planet.com/.
[13] Simera sense cubesat cameras. https://simera-sense.com/cubesat-imager/.
[14] Small spacecraft avionics and control. https://sbir.nasa.gov/printpdf/58413.
[15] Spire : Global data and analytics. https://spire.com/.
[16] Study utilizes satellite data to expose illegal fishing in north korean waters. https://www.planet.com/pulse/study-utilizes-satellite-data-to-expose-illegal-fishing-in-north-korean-waters/.
[17] A suitable cubesat adcs – tensor tech. https://tensortech.com.tw/how-to-pick-up-a-suitable-cubesat-adcs-attitude-determination-and-control-system-components-or-an-integrated-system-for-your-mission/.
[18] Tip and cue technique for satellite monitoring of moving objects. https://www.iceye.com/blog/tip-and-cue-technique-for-efficient-near-real-time-satellite-monitoring-of-moving-objects.
[19] Xact-50 attitude control system (acs) from blue canyon technologies. https://satsearch.co/products/bluecanyontech-xact-50.
[20] E. Alcantara, K. Coimbra, I. Ogashawara, T. Rodrigues, J. Mantovani, L. H. Rotta, E. Park, and D. G. Fernandes Cunha. A satellite-based investigation into the algae bloom variability in large water supply urban reservoirs during covid-19 lockdown. *Remote Sensing Applications: Society and Environment*, 23:100555, 2021.
[21] Amazon Web Services, Inc. Amazon Ground Station. https://aws.amazon.com/ground-station/, 2018.
[22] M. Bakhshalipour, S. B. Ehsani, M. Qadri, D. Guri, M. Likhachev, and P. B. Gibbons. Racod: Algorithm/hardware co-design for mobile robot path planning. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 597–609, New York, NY, USA, 2022. Association for Computing Machinery.
[23] C. R. Boshuizen, J. Mason, P. Klupar, and S. Spanhake. Results from the planet labs flock constellation. 2014.
[24] A. Candela, J. Swope, and S. Chien. Dynamic targeting for cloud avoidance to improve science of space missions. In *16th Symposium on Advanced Space Technologies in Robotics and Automation*, June 2022.

[25] A. Candela, J. Swope, S. Chien, H. Su, and P. Tavallali. Dynamic targeting for improved tracking of storm features. In *International Geoscience and Remote Sensing Symposium (IGARSS 2022)*, Kuala Lumpur, Malaysia, July 2022.

[26] J. Chen, K. Chen, H. Chen, Z. Zou, and Z. Shi. A degraded reconstruction enhancement-based method for tiny ship detection in remote sensing images with a new large-scale dataset. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2022.

[27] X. Chu, Y. Chen, and Y. Tan. An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling. *Advances in Space Research*, 60(9):2077–2090, 2017.

[28] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi. Kodan: Addressing the computational bottleneck in space. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, page 392–403, New York, NY, USA, 2023. Association for Computing Machinery.

[29] B. Denby and B. Lucia. Orbital edge computing: Machine inference in space. *IEEE Computer Architecture Letters*, 2019.

[30] B. Denby and B. Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 939–954, New York, NY, USA, 2020. Association for Computing Machinery.

[31] B. Denby, E. Ruppel, V. Singh, S. Che, C. Taylor, F. Zaidi, S. Kumar, Z. Manchester, and B. Lucia. Tartan artibeus: A batteryless, computational satellite research platform. 2022.

[32] K. Devaraj, M. Ligon, E. Blossom, J. Breu, B. Klofas, K. Colton, and R. W. Kingsbury. Planet high speed radio: Crossing gbps from a 3u cubesat. 2019.

[33] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *CoRR*, abs/1501.00092, 2015.

[34] L. Dreyer. Latest developments on spacex's falcon 1 and falcon 9 launch vehicles and dragon spacecraft. In *2009 IEEE Aerospace conference*. IEEE, 2009.

[35] A. Gaude and V. Lappas. Design and structural analysis of a control moment gyroscope (cmg) actuator for cubesats. *Aerospace*, 7:55, 05 2020.

[36] R. Gupta, R. Hosfelt, S. Sajeev, N. Patel, B. Goodman, J. Doshi, E. Heim, H. Choset, and M. Gaston. xbd: A dataset for assessing building damage from satellite imagery. *arXiv preprint arXiv:1911.09296*, 2019.

[37] R. Hadidi, B. Asgari, S. Jijina, A. Amyette, N. Shoghi, and H. Kim. Quantifying the design-space tradeoffs in autonomous drones. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '21, page 661–673, New York, NY, USA, 2021. Association for Computing Machinery.

[38] Z. Hasnain, J. Mason, J. Swope, J. V. Hook, and S. A. Chien. Agile spacecraft imaging algorithm comparison for earth science. In *International Workshop on Planning & Scheduling for Space (IWPSS)*, July 2021.

[39] L. He, X. Liu, L. Xing, and Y. Chen. Cloud avoidance scheduling algorithm for agile optical satellites. In M. Gong, P. Linqiang, S. Tao, K. Tang, and X. Zhang, editors, *Bio-Inspired Computing – Theories and Applications*, pages 161–172, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[40] Hyperion Technologies. Gnss200. https://space-for-space.com/wp-content/uploads/2020/04/HT_GNSS200_v2.1-flyer.pdf, 2019.

[41] G. Jocher, A. Chaurasia, and J. Qiu. YOLO by Ultralytics, 1 2023.

[42] M. Krestenitis, G. Orfanidis, K. Ioannidis, K. Avgerinakis, S. Vrochidis, and I. Kompatsiaris. Oil spill identification from satellite images using deep neural networks. *Remote Sensing*, 11(15), 2019.

[43] S. Krishnan, Z. Wan, K. Bhardwaj, P. Whatmough, A. Faust, S. Neuman, G.-Y. Wei, D. Brooks, and V. J. Reddi. Automatic domain-specific soc design for autonomous unmanned aerial vehicles. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 300–317, 2022.

[44] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[45] T. R. Loveland and J. R. Irons. Landsat 8: The plans, the reality, and the legacy. *Remote Sensing of Environment*, 2016.

[46] A. Mehrparvar, D. Pignatelli, J. Carnahan, R. Munakat, W. Lan, A. Toorian, A. Hutputanasin, and S. Lee. Cubesat design specification rev. 13. Technical report, California Polytechnic State University, San Luis Obispo, 2014.

[47] M. L. Messager, B. Lehner, G. Grill, I. Nedeva, and O. Schmitt. Estimating the volume and age of water stored in global lakes using a geo-statistical approach. *Nature Communications*, 7(1):13603, Dec 2016.

[48] S. Murray, W. Floyd-Jones, Y. Qi, G. Konidaris, and D. J. Sorin. The microarchitecture of a real-time robot motion planning accelerator. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, 2016.

[49] S. Nag, A. S. Li, and J. H. Merrick. Scheduling algorithms for rapid imaging using agile cubesat constellations. *Advances in Space Research*, 61(3):891–913, 2018.

[50] Nasa - nasa imagery of oil spill. https://www.nasa.gov/topics/earth/features/oilspill/index.html.

[51] S. M. Neuman, B. Plancher, T. Bourgeat, T. Tambe, S. Devadas, and V. J. Reddi. Robomorphic computing: A design methodology for domain-specific accelerators parameterized by robot morphology. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '21, page 674–686, New York, NY, USA, 2021. Association for Computing Machinery.

[52] M. Niroumand-Jadidi and F. Bovolo. Water quality retrieval and algal bloom detection using high-resolution cubesat imagery. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-3-2021:191–195, 2021.

[53] L. Perron and V. Furnon. Or-tools.

[54] S. Radu, M. Uludag, S. Speretta, J. Bouwmeester, A. Dunn, T. Walkinshaw, P. Kaled Da Cas, and C. Cappelletti. The pocketqube standard issue 1. Technical report, TU Delft, 2018.

[55] T. S. Rose, D. W. Rowen, S. D. LaLumondiere, N. I. Werner, R. Linares, A. C. Faler, J. M. Wicker, C. M. Coffman, G. A. Maul, D. H. Chien, A. C. Utter, R. P. Welle, and S. W. Janson. Optical communications downlink from a low-earth orbiting 1.5u cubesat. *Opt. Express*, 27(17):24382–24392, Aug 2019.

[56] I. Sanad and D. G. Michelson. A framework for heterogeneous satellite constellation design for rapid response earth observations. In *2019 IEEE Aerospace Conference*, pages 1–10, 2019.

[57] J. Swope, S. Chien, X. Bosch-Lluis, Q. Yue, P. Tavallali, M. Ogut, I. Ramos, P. Kangaslahti, W. Deal, and C. Cooke. Using intelligent targeting to increase the science return of a smart ice storm hunting radar. In *International Workshop on Planning & Scheduling for Space (IWPSS)*, July 2021.

[58] Go beyond the tip of the iceberg. with the tip and cue technique. capturing a cloud-free view of the world's third-largest iceberg. https://geosat.space/agriculture/case-study/tip-and-cue-technique/.

[59] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha, A. Kapoor, M. Sudarshan, and S. Stratman. Farmbeats: An iot platform for data-driven agriculture. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017.

[60] D. Vasisht, J. Shenoy, and R. Chandra. L2d2: Low latency distributed downlink for leo satellites. In *Proceedings of the 2021 ACM SIGCOMM*

2021 Conference, SIGCOMM '21, page 151–164, New York, NY, USA, 2021. Association for Computing Machinery.

[61] T. Wang, Y. Li, S. Yu, and Y. Liu. Estimating the volume of oil tanks based on high-resolution remote sensing images. *Remote Sensing*, 11(7), 2019.

[62] X. Wang, G. Wu, L. Xing, and W. Pedrycz. Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. *IEEE Systems Journal*, 15:3881–3892, 2020.

[63] L. Xiaolu, B. Baocun, C. Yingwu, and Y. Feng. Multi satellites scheduling algorithm based on task merging mechanism. *Applied Mathematics and Computation*, 230:687–700, 2014.

[64] Zac Manchester. KickSat. http://zacinaction.github.io/kicksat/, 2015.

[65] I. U. Zaman, A. Eltawil, and O. Boyraz. Wireless communication technologies in omnidirectional cubesat crosslink: Feasibility study and performance analysis. *IEEE Journal on Miniaturization for Air and Space Systems*, 2(3):157–166, 2021.

[66] I. U. Zaman, J. E. Velazco, and O. Boyraz. Realization of omnidirectional cubesat crosslink by wavelength-selective optical transceiver. *IEEE Journal on Miniaturization for Air and Space Systems*, 1(1):47–55, 2020.