Autonomic Computing Challenges in Fully Autonomous Precision Agriculture

Jayson Boubin[1], John Chumley[1], Christopher Stewart[1], and Sami Khanal[2]
Department of Computer Science and Engineering, The Ohio State University[1]
Department of Food, Agricultural and Biological Engineering, The Ohio State University[2]

***Abstract*—Precision agriculture examines crop fields, gathers data, analyzes crop health and informs field management. This data driven approach can reduce fertilizer runoff, prevent crop disease and increase yield. Frequent data collection improves outcomes, but also increases operating costs. Fully autonomous aerial systems (FAAS) can capture detailed images of crop fields without human intervention. They can reduce operating costs significantly. However, FAAS software must embed agricultural expertise to decide where to fly, which images to capture and when to land. This paper explores *fully autonomous precision agriculture* where FAAS map crop fields frequently. We have designed hardware and software architecture. We use unmanned aerial systems, edge computing components and software driven by reinforcement learning and ensemble models. In early results, we have collected data from an Ohio cornfield. We use this data to simulate a FAAS modeling crop yield. Our results (1) show that our approach predicts yield well and (2) can quantify computational demand. Computational costs can be prohibitive. We discuss how research on adaptive systems can reduce costs and enable fully autonomous precision agriculture. We also provide our simulation tools and dataset as part of our open source FAAS middleware, SoftewarePilot.**

I. Introduction

In 2050, the planet will support 9.7B people. People (and the livestock they eat) will demand 1.7X more food [1]. However, rising sea levels and city sprawl will decrease arable land in growing countries. Atmospheric CO_2 will dampen food production as climate change worsens [2], [3]. Producing enough food and providing access to it is a grand challenge.

Food demand will require larger crop yields per arable acre. Today, only 60–80% of planted seeds yield edible crops [4]. Nitrogen deficiency, hydration, crop diseases, pests and fungi degrade yield. *Precision agriculture* uses satellites, aircraft, soil sensors and digital weather stations to sense field conditions, monitor crop health, detect problems early and improve crop yields [5], [6]. For example, Integrated Pest Management uses images collected from aircraft to detect and count insects and vermin in a crop field [7]. Armed with this data, farmers can apply pesticides parsimoniously to reduce costs and sustain the planet. More generally, we broadly define *yield maps* as whole-field characterizations of crop and/or soil health. One goal of precision agriculture is to produce yield maps which farmers may use to manage crops.

Crop health changes over time because nitrogen levels diminish, pests emerge and water needs vary. Yield maps produced frequently can detect changes promptly. Problems detected early afford cost effective solutions. However, frequent mapping is costly and can exceed savings. Figure 1 plots costs to map a 250-acre corn field frequently (weekly) for 12 weeks. In comparison, current practice creates a field map once per growing season [8]. Corn fields in Ohio USA profit $148 per acre on average [9]. Figure 1 plots costs relative to profit. Airplane pilots charge $5–$10 per acre to produce maps [10], [11]. At $5 per acre, 12 whole-field maps would cost 40% of profits. Unmanned aerial systems (UAS) use small aircraft and do not carry pilots onboard. Instead, they are piloted remotely. Figure 1 assumes UAS pilots charge $20 per hour and cover 12 acres per hour. Frequent mapping using UAS consumes 14% of profits. Despite remote piloting, labor associated with piloting would account for 12% of profits.

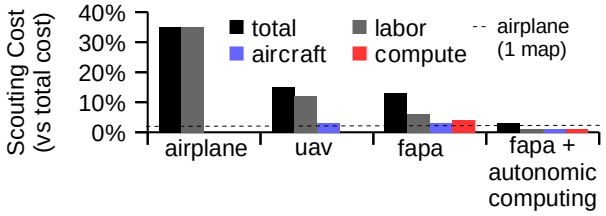

Fig. 1. Fully autonomous precision agriculture reduces and shifts costs.

Fully autonomous aerial systems (FAAS) eschew human piloting. Instead, FAAS software manages flight, captures images and analyzes data. FAAS software can fly more safely and efficiently than human pilots. Experts agree that fully autonomous systems will replace unmanned aerial systems [12], [13]. However, as shown in Figure 1, the economic case for precision agriculture is complicated. We model fully autonomous precision agriculture (FAPA), i.e., FAAS that produce yield map reports. FAPA reduces labor costs significantly from $20 per hour to $10 per hour, but the compute resources needed to execute FAAS software increase total costs. Naive FAPA accounts for 13% of profits.

Autonomic computing systems adapt their execution at runtime, speedup compute workloads and use fewer resources. FAPA systems can use autonomic computing to reduce labor and compute costs. As shown in Figure 1, if autonomic computing techniques can reduce costs by 66%, frequent mapping would be cost competitive with current practices. We contend that autonomic computing is the missing ingredient in FAPA systems.

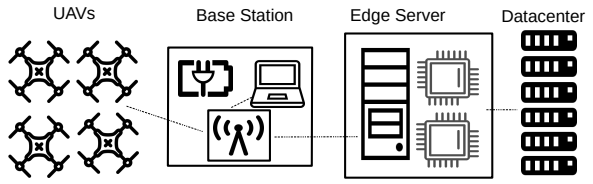

Fig. 2. Core agricultural FAAS system components.

This paper reports design and implementation for an early FAPA prototype. Our FAPA system uses consumer-grade UAV, edge and cloud infrastructure. Reinforcement learning and deep feature extraction are key software components. FAPA must compare recently sensed data to anticipated outcomes, because image analysis alone does not fully capture utility for reinforcement learning. Our prototype uses ensemble models. Each model correlates to expected yield. Variance among models captures utility. When our prototype achieves desired utility, it lands and produces a yield map. Our reinforcement learning approach intelligently samples the field and produces a yield map without exhaustively exploring the field in its entirety.

We evaluated our FAPA system on aerial, cornfield images. Each image represents a GPS location. The FAPA aircraft flies between recorded locations. At each location, the aircraft computes its reinforcement learning workload and improves its yield map. Our FAPA system produced low-error yield maps. Error decreased with sample size, but converged when 40% of the field was sampled. In terms of cost, accelerated computing systems inflate hardware costs but also execute faster and reduce hourly labor. Early analysis of the FAPA workload reveals opportunities for autonomic computing. For example, self-aware integration between FAPA systems could reduce sample size. Edge to cloud bursting could reduce hardware costs.

All software tools used to collect and extract our dataset and simulate FAPA can be found as part of our open source FAAS middleware SoftwarePilot, which is hosted on Github [14].

II. Design

In this section, we outline a hardware and software architecture for FAPA. The hardware design considers agricultural settings and resource constraints. The software design employs reinforcement learning to manage flight actions, e.g., fly north, south, east, west and land.

Hardware Architecture: Figure 2 depicts the main hardware components. Unmanned aerial systems (UAS), base stations, edge gateways, and data centers play unique roles. Below, we describe each.

UAS provide remote sensing. They capture and store images and tag their geographical position (e.g., using GPS). Modern UAS cameras support still images and video across a number of spectra. UAS fly between positions without human pilots, reducing operating costs. However, their battery life is limited. Modern UAS can fly 20–40 minutes but recharge 60–80 minutes. UAS have onboard processors, and can perform lightweight processing on-board. Complex processing is offloaded to support long battery life and reduce execution delay.

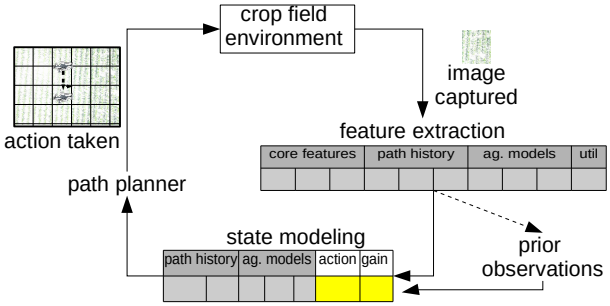

Fig. 3. Reinforcement learning underlies our approach.

Due to power constraints, maximizing the utility of UAS flight is important. The number of images required to map an acre of cropland has a quadratic relationship with the spatial resolution of the images taken. The spatial resolution of the images taken must be chosen thoughtfully depending on the application. Some systems may only need centimeter precision for tasks like yield monitoring, while more complex pest and disease detection systems may need sub-millimeter precision to properly discern features. The average area covered per UAS mission is dependent on the precision of each image, and is a primary metric that designers should account for when considering UAS storage capacity, charge time, classification needs, total scanning time, total number of UAS, and edge compute resource provisioning.

Base stations enable communication between UAS, power edge computing systems and recharge UAS in the field. The electric grid and/or sustainable off-grid power supplies base stations. As a result, the base station houses wireless networks and edge computers. FAPA require long range wireless networks capable of communicating with multiple UAS over many acres. In swarm setups [15], UAS may proxy communication through base stations. The base station can also provide docking and charging for UAS. Route planners can use in-field charging stations to avoid long return flights to the UAS home base. Configuration of the base station varies across applications and fields. Specifications of radio frequency, charging dock, and edge gateway link depend on farm size, distance from the gateway, compute resource power constraints, and the number of UAS supported.

Edge servers pull images from UAS, extract pertinent features, direct high-level flight paths and produce yield maps. They are the computational engine for FAPA workloads. Workload kernels hosted on edge servers include: image analysis, deep learning, path planning, crop modeling and flight API management. These computationally intensive kernels can require powerful processors and accelerators, e.g., GPUs. Edge servers can have significant energy demands. These demands can stress base stations powered by sustainable, off-grid sources, e.g., power cells or renewable energy. Edge servers may also draw expensive consumer electricity from electric grids.

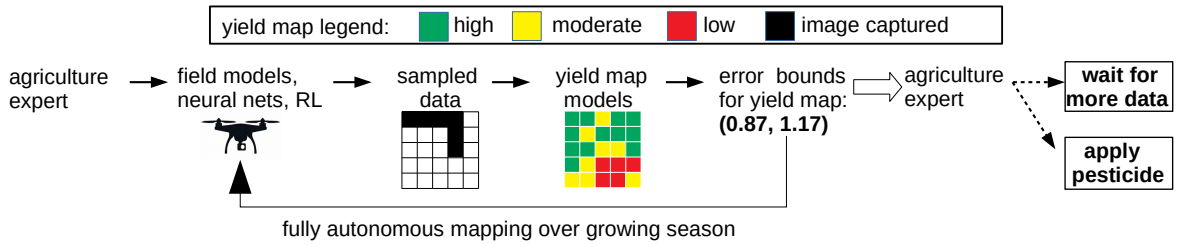

Fig. 4. Fully autonomous precision agriculture

Edge servers are constrained by base station power supplies. Large clusters are not permissible, but at times, edge servers may suffer large compute workloads. For example, a swarm of UAS executing FAPA may simultaneously capture images that require heavy processing. Data centers can offload demands when Internet connections are available. Complex feature extraction, i.e., the execution of deep neural networks, SVM and other classifiers, are attractive offloading workloads. Offloading also helps when many models must be run and may be used adaptively, e.g., only when load is high enough to cause service level violations. Cloud offloading requires bi-directional Internet connection capable of transmitting images and classification results rapidly.

Software Architecture: As shown in Figure 3, FAPA software executes feature extraction, state modeling and reinforcement learning. FAPA software is fully autonomous— humans neither fly nor direct UAS. The workflow is (1) collect data, (2) extract features related to FAPA goals, (3) determine if goals have been met, and (4) create new flight paths such that future sampled data will fulfill modeling goals.

We use the term features extraction broadly. Any image or data processing reduction constitutes feature extraction. Examples include geographic location (data processing), object detection (image processing), pose estimation (image and data processing) and simple brightness and contrast computations. Feature extraction procedures differ greatly in compute needs and their ability to be reused across applications and fields.

Feature extraction produces a *feature vector*, i.e., a vector containing numerical values representing each extracted feature. The feature vector is used to compute *utility*, i.e the usefulness of an image/state toward accomplishing FAPA goals. Utility calculations are customized for each precision agriculture objective. Usefulness clearly depends on purpose. Once a utility value is obtained, the FAPA system uses the utility value along with historical information to model the state of its explored area. This may include updating maps of sensed information, or using high level models to gain meta-information about the collection of feature vectors the FAPA system has already collected. This state model can be used to determine whether the user-provided FAPA system goal has been accomplished and the UAS must land, or whether more data must be collected.

Fully Autonomous Precision Agriculture: This sensing, extracting, modeling, and pathfinding cycle is repeated until the FAPA system runs out of data, a system component runs out of power, or the FAPA system goal is met. Many FAPA systems can be composed using this basic design. In this paper, we implement a FAPA system to model crop yield in corn fields.

Figure 4 describes the design of the yield-modeling FAPA system. The goal of this system is to generate a yield map of the crop field. Due to the size of crop fields and the lack of resources available in them, it is wise to sample the field, taking pictures of only a percentage of the field to predict yield. Using our FAPA system design, we have created a system that can sample a crop field based on the goal of yield modeling and extrapolate sampled ground truth points into a yield map of the entire field with low error.

To do this, the system must have appropriate feature extraction models, state models, and historical training data. Green Excess and Leaf Area Index are strongly correleted with yield. We use them in our feature extraction procedure and to create yield maps. The details of these models are orthoganol to this paper. We refer the reader to [8] for details.

Our FAPA goal targets yield map error. Low error maps bolster confidence for costly management choices. Hence, high utility samples— i.e., useful images— reduce yield map error. However, a problem emerges: how can we compute utility without knowing true yields in advance?

We estimate error by using an ensemble modeling approach. We assume whole-field yield error and aggregate variance converge comparably with sample size. We use multiple models to obtain better predictive performance, i.e., ensemble modeling. Once the ensemble models converge beyond a user defined threshold, we consider our map complete. If the error of the ensemble models is above the user defined threshold, we fly to a new location and sample the field again. Using Green Excess and Leaf Area Index, we can calculate utility gain by estimating the change in error for our ensemble models for each possible path the UAS may take. The direction which decreases ensemble error the most is chosen.

III. Implementation

Using this design, we have implemented a FAPA system that is capable of generating such yield maps in simulation. Our simulation environment relies on data sensed from real UAVs to generate yield maps. Our dataset consists of over 10,000 12 megapixel images captured at 33 feet over a 75 acre Ohio corn field at a spatial resolution of $4mm^2$. Our images

Algorithm 1 FAPA Simulation

```
1: yieldmap ← empty
2: i ← starting image
3: ensembleError ← ∞
4: while ensembleError > threshold do
5:     featVec ← extract(i)
6:     yieldmap[featVec[lat],featVec[lon]] = featVec[GE]
7:     errVector ← findNextFlightAction(featVec)
8:     dir = min(errVector)
9:     i = nextImage(i, dir)
10:    imagesMapped ← error(dir)
11: return extrapolate(yieldmap)
```

were captured at 3 points in the growing season, June, August, and September. The images are all geo-tagged, allowing us to place them in a virtual map. Using this data set, appropriate feature extraction, FAAS control algorithms, and ensemble error models, we fully simulate a FAPA system.

Our simulator takes the series of coarse steps denoted in Algorithm 1 to replicate FAAS action. The simulator described takes three arguments: 1) a set of geo-tagged images representing a crop field, 2) a starting image in the image set, and 3) an ensemble model error threshold. The simulator will use these arguments to output an estimated green excess yield map. Algorithm 1 starts by initializing an empty yield map, a starting image (i), and ensemble error. The yield map is initialized as a 2 dimensional array of zeros. Each cell in the array represents an image in the image set. The goal of the FAPA simulation is to fill this map with real and extrapolated green excess values that can be used to predict yield. Until ensemble error falls below the user provided threshold, it loops over the sampling procedure.

The sampling procedure begins by extracting features from image *i*. The goal of feature extraction is to retrieve from an image a vector of floating point values representing specific, user defined features that are useful for pathfinding, yield estimation, and geo-location. Our simulator extracts 15 features, including GPS location, green excess, leaf area index, and corn detection. These features are represented in the *featVec* variable.

The next step in the sampling procedure is to add the information learned from feature extraction to our yield map. To do this, we add the green excess value of *i* (*GE*) into the position in *yieldmap* corresponding to the latitude and longitude of *i* (*Lat, Lon*) obtained from the feature vector.

Once the yield map has been updated, we must find the next image to sample. To conserve energy and sample efficiently, we choose this image based on the images directly adjacent to *i*. To find the next image, we must find the next flight action (i.e a movement north, south, east, or west) then find the image corresponding to that movement. To find the next flight action, we use a reinforcement learning approach. Our approach uses a series of feature vectors from past FAAS executions to make pathing decisions. For each feature vector, information is also known about the possible flight actions for that position. For our simulation, our feature vector data set includes the green excess of all adjacent images. This information can be used to calculate utility gain, a quantification of improvement of our model based on the addition of that feature vector's information.

Using reinforcement learning, we can estimate the utility gain for each flight action using similar prior execution data without actually sampling each flight action. To find similar execution data to a feature vector, the simulator runs the K-Nearest Neighbors [16], [17] algorithm using our feature vector data set as the reference set and the current feature vector as the query. This returns the K most similar feature vectors in the data set to the current feature set. These feature sets are used to calculate utility gain.

To determine the best direction to choose for sampling, the simulator uses the green excess of each of the K-Nearest Neighbors to minimize the error in its unfinished yield map. Map error is estimated using ensemble models. Ensemble models are very simple formulae that converge when the map is complete. In the case of this FAPA simulation, ensemble models all converge to mean green excess. We use five models to make up our ensemble: mean, median, 95% confidence interval min and max, and coarse mean (i.e $(min + max)/2$). To determine ensemble error, we find the range between the ensemble model values. For each flight action, we calculate the average ensemble error when adding each of the K-nearest neighbors of the current feature vector to the current yield map. The flight action which decreases ensemble error the most is chosen. In Algorithm 1, the *findNextFlightAction* function takes the feature vector, finds its K-Nearest Neighbors using our feature vector data set, calculates ensemble error, and returns a error vector containing the average ensemble errors of each flight action. The direction with the smallest error is chosen.

Once the flight action is chosen, the next image is selected based on the current image and flight direction, and is assigned to *i*. This sampling procedure continues until the user provided amount of images are sampled. Once sampling concludes, empty regions of the yield map must be filled in. This is done using recursive dilation.

The extrapolate function in Algorithm 1 recursively dilates the unfinished yield map until it is full. The dilation procedure begins by creating a copy of the current yield map. The procedure iterates over the original map, looking for empty squares with full indices in their eight-connected neighborhood. Once an index fitting this description is found, the mean of its full eight-connected neighbors is added to the copy map. Once all indices have been checked, the copy map is evaluated. If all indices of the copy map are full, it is returned. If any index in the copy map is empty, it is then extrapolated and its extrapolated version is returned.

IV. Early Results

The simulator supplied was sensed data from 7,000 cornfield images composed into 1350 sample runs. We profiled energy demands for aircraft and compute offline. For aircraft, we

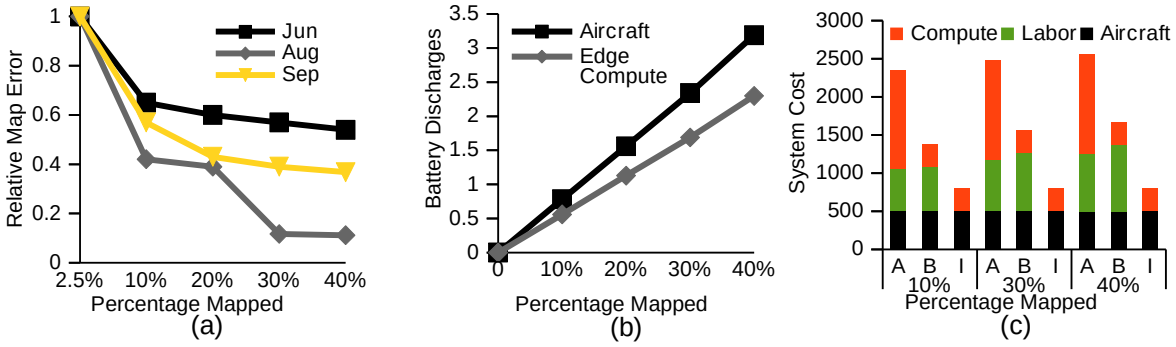

Fig. 5. Early results: (a) Yield error decreases as sample size increases, (b) FAPA aircraft and compute have comparable energy demands and (c) low cost FAPA requires balancing compute and labor costs. Sampling percentages are relative to a 75 acre field.

repeatedly tested flight commands in each direction in isolation and use average energy demand. For FAPA software, we measured delay and power on a Lenovo Thinkpad T470 with Intel i7 7500u CPU and NVIDIA 1080 GPU. We did not use cloud offload for these tests. Figure 5 shows early results pertaining to yield error, energy demand, and cost.

Figure 5 (a) describes the sum of squared error (SSE) between our extrapolated yield maps and ground truth yield maps. We normalized to SSE of a 2.5% sampling size. Figure 5 (a) shows extrapolated yield map error decreases as sampling area increases. The selected sampling points reflect knee-points in our ensemble of models. Variance in the ensemble converges with SSE. This property generalizes our reinforcement learning approach, because utility models can transfer across fields and applications, compute local ensemble variance and identify sample sizes that yield good accuracy. For example, consider accuracy reported across monthly data sets. We are able to reuse ensemble models while still finding good sample sizes. The correlation is imperfect. August data converges more quickly September and June. Future work will explore the factors affecting convergence and look for techniques to further align local model calculations with global objectives. In addition, we observed that our pathfinding algorithm (nearest neighbors) performed poorly. At times, the FAPA system took clearly wrong turns. We believe that redundancy elimination in selected features, i.e., PCA, could improve pathfinding. Algorithms like A* search will help also.

Figure 5 (b) depicts estimated energy consumption. The aircraft is a DJI spark with a stock battery. Aircraft discharge slightly outpaces edge discharge. FAPA composed with battery powered edge servers must include substantial off-grid batteries to avoid frequent recharge. For our 75 acre field, an aircraft can map about 10% of the field before recharge.

While the edge system can be easily provisioned with greater battery capacity, aircraft must land to be recharged, or have their batteries swapped. Highly provisioning edge system batteries may be an important step towards realizing FAPA. A highly provisioned edge system should be able to control multiple UAS, increasing energy demands linearly. For large fields, links from the edge to the cloud may be required to offload computation to meet objectives for all aircraft.

Figure 5 (c) shows the cost of implementing FAPA systems. *System A* is an accelerated FAPA system, equipped with one UAV and an edge device equipped with an accelerator (GPU, FPGA, etc). *System B* is a basic FAPA system, equipped with an edge system with only a simple CPU. *System I* is an ideal FAPA system, including autonomic software components that allow the system to operate labor free.

The cost of each system is broken down between aircraft, compute, and labor. Each system requires the same aircraft and basic compute hardware. Systems A requires accelerators that decrease the runtime of our FAPA algorithms and increase system throughput. Systems A and B also both require labor. Without autonomic software components, workers would be required to set up, execute, and take down the FAPA system. System I eschews this cost by implementing autonomic policies in software. Workers being paid $10 an hour would cost hundreds of dollars over the course of the season performing tasks that can be performed by autonomic software alone. Labor costs increase as sample rate and field size increase. The addition of autonomic software components to intelligently manage aircraft charging, scheduling, and dispatching, as well as data movement and reporting could considerably cut the cost of FAPA system implementation, especially for large farms.

V. Discussion

Table I outlines autonomic software techniques that could greatly reduce FAPA costs. There are opportunities across the software stack. In this section, we highlight a few opportunities to motivate future work.

Self-aware system integration seeks autonomous component reuse, integrating components without human programmer intervention. FAPA could use self-aware integration to expand their feature vector. Features computed by FAPA on nearby fields could influence where FAPA systems fly and when they land. Integrating more features can reduce sample size required for low yield error which reduces labor costs. The challenge is to produce a scalable infrastructure for feature sharing in resource constrained agriculture settings.

Quality-adaptive models [18]–[20] allow multi modal feature extraction. One mode uses computationally intensive but

Topic	Systems Layer	Adaptive Computing Challenges
Self-Aware Integration	Autonomic Management	Seamlessly integrate features extracted by other devices on nearby fields
Quality-Adaptive Models	Autonomic Management	Speedup feature extraction when accuracy does not degrade outcome
Load Balancing	Middleware	In swarms, dynamically re-partition fields to avoid long recharging delays
Adaptive Power Management	OS/Architecture	Duty cycle power hungry devices while keeping execution time low
Edge to Cloud Bursting	OS/Architecture	Use cloud resources to multiplex thin edge clients across multiple fields
Micro-UAV Swarms	Cyber physical	Reuse smaller, cheaper aircraft over multiple mapping missions and fields
Performance Modeling	Pervasive	Model end-to-end execution time of mapping missions

TABLE I
Autonomic computing opportunities in FAPA.

accurate models. Other modes use less intensive models when inaccurate features do not affect FAPA outcomes. Quality-adaptive models seek to switch between models to conserve battery life on edge and aircraft. The ability to choose lower quality models in situations that allow for them should speed up FAPA compute time, decreasing labor costs or allowing for an increase in sampling percentage and relative battery life.

Self integration and quality-aware models necessitate a new system layer between middleware and application. We propose the Autonomic Management Layer, a new system layer between application and middleware responsible for the management of autonomic software components. To maximize the utility of a FAPA system, an ensemble of autonomic components must be added to decrease labor expenses and increase sampling and mapping accuracy. These components are too high level for the middleware, which handles data movement, aircraft flight, and classification. Conversely, they are too low level for the application layer, which handles user interaction and reporting. The new layer is required to interpret user provided data like field maps, utility functions, and classifiers, into middleware actions that implement high level autonomic policies.

Adaptive power management duty cycles power hungry devices including: unused aircraft, GPUs and accelerators and compute processors. This technique prolongs aircraft and edge battery life and avoids long recharging delays. To be sure, aircraft consume energy hovering idly. Computation delays affect energy usage on edge servers and aircraft. Smart power management prolongs both aircraft and edge batteries (or reduces edge energy costs). For example, by autonomically decreasing the number of active aircraft in a swarm or the number of accelerators operating [21], [22], classification and pathing latencies will change and impact system life. When combined with quality-adaptive models, adaptive power management may be able to save power at zero cost to system performance. In addition, lessons from sustainable computing where multiple energy sources power compute (and now devices) are pertinent [23]–[26].

Edge to cloud bursting also requires interaction between the autonomic layer and OS/Architecture layer. The FAPA model described in this paper uses relatively simple models to cover a relatively small area with one aircraft. If the number of aircraft, size of the farm, or model complexity increase within reason, it is plausible that normal edge systems will not be able to process all of the images they receive quickly enough for aircraft. Aircraft will have to hover, wasting power, waiting for instructions from edge systems. The autonomic and OS/Architecture layers can work together to instead move data to the cloud when aircraft hover times are too high.

VI. Conclusion

Precision agriculture is increasing crop yields across the globe in an effort to feed our growing population. Using FAAS in precision agriculture applications can considerably reduce labor costs and increase field mapping frequency. In this paper, we present a design of a FAPA system along with a simulated implementation. We examine trade-offs concerning accuracy, power, and cost of the total system using profile information from real FAAS missions. We demonstrate that our FAPA system can produce low-error yield maps by sampling 40% or less of the total crop field, decreasing the energy and labor costs of a FAPA implementation. We also identify autonomic computing mechanisms that can help decrease labor and hardware costs of FAPA systems further.

: Acknowledgments: This work was funded in part by NSF Grants 1749501 and 1350941 with support from NSF CENTRA collaborations (grant 1550126).

References

[1] H. C. J. Godfray, J. R. Beddington, I. R. Crute, L. Haddad, D. Lawrence, J. F. Muir, J. Pretty, S. Robinson, S. M. Thomas, and C. Toulmin, "Food security: The challenge of feeding 9 billion people," *Science*, 2010.

[2] C. Rosenzweig and M. L. Parry, "Potential impact of climate change on world food supply," *Nature*, vol. 367, 1994.

[3] X. Zhang and X. Cai, "Climate change impacts on global agricultural land availability," *Environmental Research Letters*, vol. 6, 2011.

[4] S. Savary, A. Ficke, J.-N. Aubertot, and C. Hollier, "Crop losses due to disease and their implications for global food production losses and food security," *Food Security*, vol. 4.2, 2012.

[5] C. Zhang and J. M. Korvacs, "The application of small unmanned aerial systems to precision agriculture," *Precision Agriculture*, vol. 13.6, 2012.

[6] A. ur Rehman, A. Z. Abbasi, N. Islam, and Z. A. Shaikh, "A review of wireless sensors and networks' application in agriculture," *Computer Standards and Interfaces*, vol. 36.2, 2014.

[7] L. E Ehler, "Integrated pest management (ipm): Definition, historical development and implementation, and the other ipm," in *Pest management science*, 09 2006.

[8] S. Khanal, J. Fulton, N. Douridas, A. Klopfenstein, and S. Shearer, "Integrating aerial images for in-season nitrogen management in a corn field," *Computers and Electronics in Agriculture*, 2018.

[9] L. Foreman, "Characteristics and Production Costs of U.S. Corn Farms, Including Organic, 2010," 2014.

[10] S. Garvey, "Considering trying out a uav?." http://www.grainnews.ca, 2015.

[11] C. Paterson, "A perspective from above." http://www.agadvance.com/issues/jan-2013/a-perspective-from-above.aspx, 2013.

[12] K. P. Valavanis and G. J. Vachtsevanos, "Future of unmanned aviation," in *Handbook of unmanned aerial vehicles*, 2015.

[13] L. Martin, "The future of autonomy isn't human-less. it's human more." https://www.lockheedmartin.com/en-us/capabilities/autonomous-unmanned-systems.html, 2018.

[14] J. Boubin, C. Stewart, S. Zhang, N. T. Babu, and Z. Zhang, "Softwarepilot." http://github.com/boubinjg/softwarepilot, 2019.

[15] M. Rosalie, M. R. Brust, G. Danoy, S. Chaumette, and P. Bouvry, "Coverage optimization with connectivity preservation for uav swarms applying chaotic dynamics," in *ICAC*, 2017.

[16] C. D. Yu, J. Huang, W. Austin, B. Xiao, and G. Biros, "Performance optimization for the k-nearest neighbors kernel on x86 architectures," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, (New York, NY, USA), pp. 7:1–7:12, ACM, 2015.

[17] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *CoRR*, vol. abs/1702.08734, 2017.

[18] J. Kelley, C. Stewart, N. Morris, D. Tiwari, Y. He, and S. Elnikety, "Measuring and managing answer quality for online data-intensive services," in *ICAC*, 2015.

[19] L. larson, W. Tarneberg, C. Klein, and E. Elmroth, "Quality-elasticity: Improved resource utilization, throughput and response times via adjusting output quality to current operating conditions," in *IEEE ICAC*, 2019.

[20] J. Kelley, C. Stewart, N. Morris, D. Tiwari, Y. He, and S. Elnikety, "Obtaining and managing answer quality for online data-intensive services," in *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 2017.

[21] N. Morris, S. M. Renganathan, C. Stewart, R. Birke, and L. Chen, "Sprint ability: How well does your software exploit bursts in processing capacity?," in *ICAC*, 2016.

[22] C. Wang, B. Urgaonkar, A. Gupta, L. Y. Chen, R. Birke, and G. Kesidis, "Effective capacity modulation as an explicit control knob for public cloud profitability," in *ICAC*, 2016.

[23] H. Makrani, H. Sayadi, D. Motwani, and ..., "Energy-aware and machine learning-based resource provisioning of in-memory analytics on cloud," in *Symposium on Cloud Computing*, 2018.

[24] C. Stewart and K. Shen, "Some joules are more precious than others: Managing renewable energy in the datacenter," in *ACM Workshop on Power Aware Computing and Systems*, 2009.

[25] Z. Xu, N. Deng, C. Stewart, and X. Wang, "Cadre: Carbon-aware data replication for geo-diverse services," in *ICAC*, 2015.

[26] C. Li, R. Wang, T. Li, D. Qian, and J. Yuan, "Managing green datacenters powered by hybrid renewable energy systems," in *ICAC*, 2014.