AdaSens: Adaptive Environment Monitoring by Coordinating Intermittently-Powered Sensors

Shuyue Lan
Northwestern University
shuyuelan2018@u.northwestern.edu

Zhilu Wang
Northwestern University
zhilu.wang@u.northwestern.edu

John Mamish
Northwestern University
JohnMamish2024@u.northwestern.edu

Josiah Hester
Northwestern University
josiah@northwestern.edu

Qi Zhu
Northwestern University
qzhu@northwestern.edu

Abstract— Perceiving the environment for better and more efficient situational awareness is essential in applications such as wildlife surveillance, wildfire detection, crop irrigation, and building management. Energy-harvesting, intermittently-powered sensors have emerged as a zero maintenance solution for long-term environmental perception. However, these devices suffer from intermittent and varying energy supply, which presents three major challenges for executing perceptual tasks: (1) intelligently scaling computation in light of constrained resources and dynamic energy availability, (2) planning communication and sensing tasks, (3) and coordinating sensor nodes to increase the total perceptual range of the network. We propose an adaptive framework, AdaSens, which adapts the operations of intermittently-powered sensor nodes in a coordinated manner to cover as much as possible of the targeted scene, both spatially and temporally, under interruptions and constrained resources. We evaluate AdaSens on a real-world surveillance video dataset, VideoWeb, and show at least 16% improvement on the coverage of the important frames compared with other methods.

I. Introduction

An intermittently-powered sensor network is a group of spatially distributed devices that are powered by intermittent energy sources (e.g., solar) and collect data by measuring environmental conditions. Often, temperature, pressure, moisture, position, lighting and sound are measured and used for further analysis. These intermittently-powered devices harvest energy from the environment, but unlike traditional wireless sensor nodes, do not have a typical battery (other than a small super-capacitor). They instead operate opportunistically from harvested power, which means power failures are common [1]. Moreover, in many application scenarios, along with the aforementioned sensors, cameras are deployed in sensor networks to better perceive the environment, such as wildlife surveillance, smart buildings and crop irrigation. Compared to other sensors like temperature and humidity sensors, cameras consume more energy to perceive the environment, create more data and consume more storage, and need more communication bandwidth if the data needs to be sent back to servers. More specifically, such intermittently-powered cameras are very challenging to deploy because of the following reasons.

The first challenge for applying intermittently-powered devices to capture video is to *overcome the unstable energy supply to better perform perception tasks.* Power failures and variable energy make it difficult to provide reliable performance. Second, the sensor nodes are usually highly constrained in terms of computational and memory resources. As camera sensors are resource-hungry, a challenge is to *make cameras work with limited resources on these intermittently-powered devices for computation, communication and storage*. Third, in a sensor network for monitoring on a target area, there is often redundancy among the information captured by different sensor nodes. For example, two cameras may monitor the same parking lot with different but overlapping views. If a vehicle enters their overlapping area, we may only need one camera to be on in order to cover this interesting event and save the energy of the other camera for capturing future important scenes that have interesting events. The third challenge is thus *how to coordinate and utilize all intermittently-powered nodes to better cover the important scenes.*

Many multi-camera monitoring networks have been developed for various applications, such as vehicle tracking [2], wide-area surveillance [3], complex activity detection [4], etc. However, none of them considers energy-harvesting (EH) powered, intermittent devices. There are works on developing efficient checkpoint schemes and algorithms for low-power/intermittent devices, such as inference on intermittent embedded systems [5], [6], [7] , computational and energy harvesting task scheduling [8], [9], and efficient communication [10]. Other works focus on custom hardware that enables energy management [11] and timekeeping [12]. However, none of them has addressed multi-node collaboration for intermittently-powered sensors. In contrast, our work considers system-level optimization of intermittently-powered sensor networks for adaptive environment monitoring.

More specifically, we address the challenges in environment monitoring with intermittently-powered camera sensors, where multiple sensor nodes with cameras monitor the target area from different points of view with overlapping. These nodes are EH-powered with varying energy level from time to time, and sometimes, the energy level may not be enough for the energy cost of the sensing tasks. In the scene, important events may happen at random time and duration, which means that not every frame is worth being captured and the system should save more energy for important frames.

We thus introduce **AdaSens**, an optimization framework for adaptive environment monitoring on intermittently-powered sensors. In AdaSens, we have multiple nodes equipped with a camera, a super-capacitor and an energy harvesting module. A remote conductor node is built upon these sensor nodes,

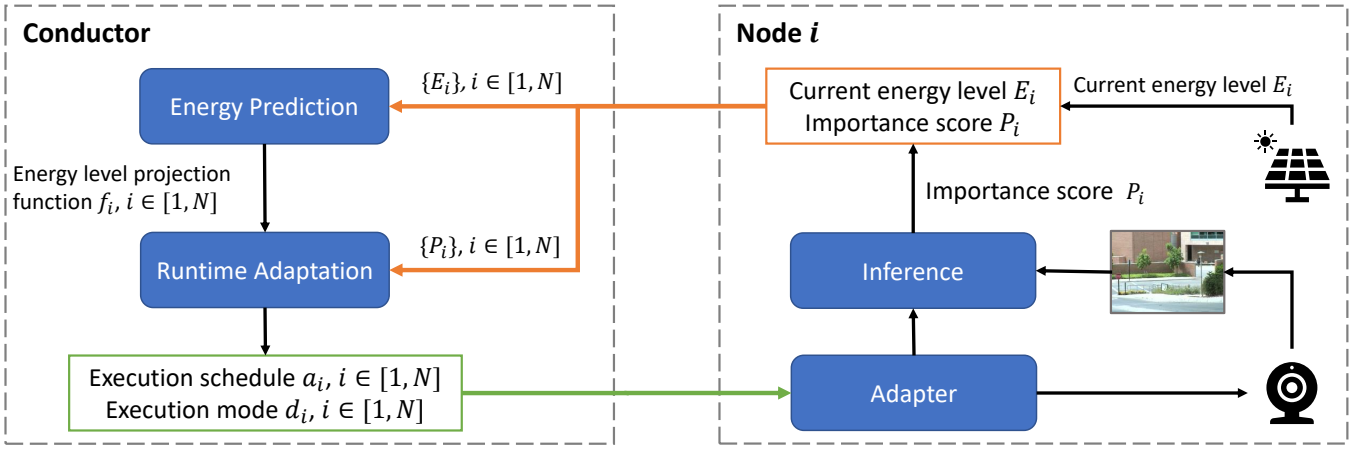


Fig. 1: **The overall framework of AdaSens.** The sensor nodes perceive the environment and perform inference about the scene. The conductor uses such information to predict the energy level of each sensor node in the future. Then it conducts run-time adaptation to decide the executions for each sensor node in the next adaptation period.

which will perform system-level optimization to guide the operation of the sensor nodes. Our objective is to cover as much as possible of the targeted scene given the limited and unstable harvesting power. The sensor nodes are responsible for measuring the environment, acquiring the current energy level of itself and performing inference on captured data. The information will be sent to a remote server, which serves as the *conductor* node. The conductor will take such information to predict the energy level of each sensor node and decide the future execution schedules with a run-time adaptation scheme. In this way, every sensor node is scheduled to collectively optimize the scene coverage given the intermittent energy availability. The main contributions of our work include:

- We develop a framework AdaSens for adaptive environment monitoring by coordinating intermittently-powered sensors, with a conductor node to guide the operations of sensor nodes for optimizing the coverage of scene under limited and unstable harvesting power.
- We formulate the operations of sensor nodes as a mixed-integer linear programming (MILP) problem to optimize an objective that captures spatial, temporal and content coverage of the scene.
- We demonstrate the effectiveness of AdaSens on a real-world surveillance dataset and achieve better performance than other methods.

II. AdaSens Framework

A. Overview

With an intermittently-powered sensor network for environment monitoring, our objective is to develop a system that covers as much as possible of the targeted scene under limited and unstable power supply. In the proposed system, we have N intermittently-powered nodes equipped with one or multiple sensors that has a super-capacitor and energy harvesting module. A conductor node is built upon these sensor nodes, which periodically performs system-level optimization to guide the operations of sensor nodes.

The overall framework is shown in Fig. 1. The system is equipped with a conductor node that coordinates the operations of individual sensor nodes and carries out global optimization for the scene coverage, and multiple intermittently-powered sensor nodes that perceive the scene. The information captured on different sensor nodes may have redundancy. The sensor nodes are responsible for perceiving the environment, acquiring the current energy level of itself and inferring the importance of the scene. The information will be sent to the conductor node. The conductor will take such information to predict the energy level of each sensor node in the future. Then a run-time adaptation module will decide the execution schedules for each sensor node in the next adaptation period. In this way, every sensor node is scheduled as to optimize the scene coverage given the power situation.

Notations. We denote the N sensor nodes as the set of $\mathcal{N} = \{1, 2, ..., N\}$ and divide the targeted area into M partitions as the set of $\mathcal{M} = \{x_1, x_2, ..., x_M\}$, where $x_m, m \in [1, M]$ is the area of the m-th partition. For each sensor node i, $\lambda_{i,m}$ is a binary indicator for sensing coverage. $\lambda_{i,m} = 1$ indicates that the view of node i can cover the m-th partition. The adaptations are made periodically with a period of T. Before each adaptation period, the conductor gets from node i its current energy level E_i and an importance score P_i computed with data from last execution. Based on such information, the conductor decides whether to perform a series of tasks (referred as atomic tasks) at each time step t for each node i, indicated as $a_i[t]$. The conductor considers K periods ahead, i.e., KT steps, when optimizing the spatial-temporal coverage and the content coverage. We refer to K as the adaptation horizon. After optimization, only the schedule of the first T steps will be applied for execution. The optimization considers the following factors: the energy using rate Δeu_i per atomic task of node i, the energy creation rate $\Delta ec_i[t]$ of node i at time t, the energy cost E_c for one-time communication of sensor nodes and the power capacity C of each node. Here we assume all nodes to be identical in hardware settings. Along with the execution schedule, d_i indicates the execution mode

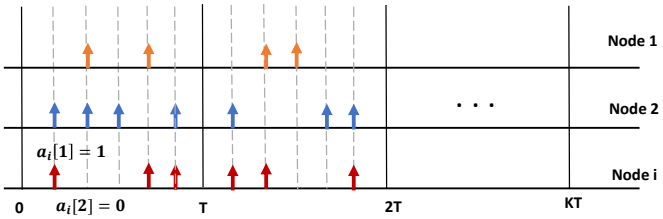


Fig. 2: Illustration of adaptation variables in AdaSens.

of each node i.

Adaptation Variables. In the proposed framework, the conductor aims at optimizing the scene coverage by deciding for every node whether to perform an atomic task at each time step. Fig. 2 illustrates the adaptation variables in the optimization framework. We consider discrete time execution. At the beginning of each adaptation period, the conductor decides whether to perform an atomic task at time t for each node i, indicated as $a_i[t]$. For example, in the figure, $a_i[1] = 1$ means that the node i is instructed to execute an atomic task at the time step 1, while $a_i[2] = 0$ indicates that node i is not required to execute any task at the time step 2. During operation, at time 0, the conductor will try to optimize the scheduling of KT steps ahead, i.e., over $[0, KT)$, according to the predicted energy availability and the current node feedback; but it only sends the optimized schedule for the first T steps, i.e., over $[0, T)$ to the nodes for executions. Then at time T, the conductor will run the optimization again for $[T, T+KT)$ steps and send the optimized schedule of $[T, 2T)$ to each node for executions. Such design is to improve system performance by optimizing over a longer time horizon while running the optimization with a shorter time interval.

B. Conductor

The goal of our conductor is to schedule the operations of individual sensor nodes, with the consideration of the scene coverage. The conductor has two modules: energy prediction and runtime adaptation.

Energy Prediction. We assume that the conductor has environment sensors for acquiring the necessary information for energy prediction. In our framework, we consider solar energy as an example. Given the environment information and the current energy level E_i from sensor node i, the energy prediction module generates the energy level projection function at time t for node i as:

$$f_i(t, E_i^{used}) = E_i + \sum_{j=0}^{t-1} \Delta ec_i[j] - E_i^{used} \tag{1}$$

where E_i^{used} is the energy used from 0 to t-1, which includes the energy cost of the scheduled executions and the communication costs in between. $\Delta ec_i[j]$ is the energy creation rate for sensor node i, which can be computed with current solar radiation and future predicted radiations. Based on [13], the power generated by a solar panel can be calculated as:

$$P_{solar} = AS \tag{2}$$

where A is the area of the solar panel, and S is the solar radiation perpendicular to the panel surface. The solar radiation can be predicted based on the historical data. P_{solar} is then taken as $\Delta ec_i[j]$.

We develop a Long Short-Term Memory (LSTM) model to predict the future solar radiations S based on the environmental history. We use solar profiles from [14] and compute the Pearson's correlation coefficient between environmental sensor measurements and the solar radiations. Based on the analysis, we choose a feature vector with seven sensor measurements, such as air temperature, humidity, pressure, etc. These data are used to train the LSTM model. In this way, we will have a model that takes some historical measured data as input and outputs the predicted solar radiations in the future.

Runtime Adaptation. The runtime adaptation module schedules the future operations of the sensor nodes given the energy level projection functions and the importance scores from last execution. For each discrete time t, we have a binary indicator to indicate whether a sensor node is executing. $a_i[t] = 1$ means that node i is executing at time t, and vice versa. As stated before, we update the execution schedule in a periodic manner with a period of T. For every adaptation, we predict K periods ahead to optimize our objective during the time KT, and then we apply the execution schedule of the first period to each sensor node. The optimization problem is formulated as an MILP problem as shown in Eqn. (3):

$$\begin{aligned}
&\max_{\substack{\forall a_i[t],\\ i\in[1,N],\\ t\in[0,KT)}} \sum_{t=0}^{KT-1} \sum_{m=1}^{M} x_m \cdot \max_{i\in[1,N]} (\lambda_{i,m} \cdot p_i[t] \cdot a_i[t]) \\
&\quad \text{s.t. for } i \in [1, N],\ t \in [0, KT) \\
&\qquad f_i[t] = E_i + \sum_{j=0}^{t-1} (\Delta ec_i[j] - \Delta eu_i \cdot a_i[j]) - \left\lfloor \frac{t}{T} \right\rfloor E_c, \\
&\qquad f_i[t] \le C \\
&\qquad f_i[t] \ge \Delta eu_i \times a_i[t], \quad t \ne kT - 1,\ k \in [1, K] \\
&\qquad f_i[t] \ge \Delta eu_i \times a_i[t] + E_{comm}, \quad t = kT - 1
\end{aligned} \tag{3}$$

The objective function in (3) maximizes the spatial-temporal coverage of the targeted scene. For each scene partition of area x_m, we consider the sensor nodes that are capable to cover this area, indicated by $\lambda_{i,m}$, where $\lambda_{i,m} = 1$ indicates that area x_m is in the view of node i. $a_i[t]$ indicates whether node i is executing at time t. $p_i[t]$ is the decayed importance score of node i at time t, derived from the importance score P_i based on the executions of node i from the last period as:

$$p_i[t] = 0.5 + (P_i - 0.5) \cdot \exp\left(\frac{-t}{\rho}\right) \tag{4}$$

where ρ is the decaying factor. The intuition for the decayed importance score is that the scene is changing as time goes by and the inference of importance will become less confident. By this decaying function, we are trying to make every node become equally important after a certain point of time. Taking $p_i[t]$ as a scaling factor, the objective is to maximize the sum of the areas covered by the system through a period of time. In the defined optimization problem, we have four types of constraints:

- *Energy dynamic function.* This constraint is the energy dynamic function, which computes the projected energy

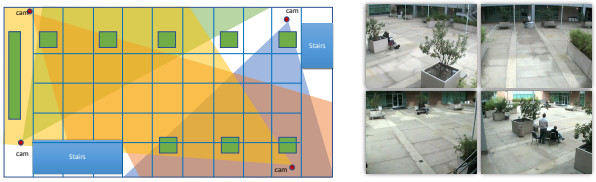
Algorithm 1 Conductor of AdaSens

Input: adaptation period T, horizon K
Input: initial available energy $E_i, \forall i \in [1, N]$
Initialize $P_i = 0.5, \forall i \in [1, N]$.
Initialize set of connected nodes $\Theta = \{i \mid i \in [1, N]\}$.
repeat
  Denote current time as 0.
  Predict solar radiations by LSTM model to estimate energy creation rate $\Delta ec_i[t],\ t \in [0, KT)$ (Eqn. (2)).
  $p_i[t] = decay(P_i, t),\ t \in [0, KT),\ i \in \Theta$ (Eqn. (4)).
  MILP.set$(K, T, p_i[t], E_i, \Delta ec_i[t] \mid t \in [0, KT), i \in \Theta)$
  $\{a_i[t] \mid t \in [0, KT), i \in \Theta\}$ =MILP.solve() (Eqn. (3)).
  if MILP solver has feasible solutions **then**
    Set in-network mode $d_i = 1$ for $i \in \Theta$.
  else
    Set isolation mode $d_i = 0$ for $i \in \Theta$.
  for $i \in [1, N]$ **do**
    Send d_i, $a_i[t], t \in [0, T)$ to node i.
  Wait for T time steps.
  $\Theta = \{i \mid i \in [1, N]\}$.
  for $i \in [1, N]$ **do**
    Receive new E_i, P_i from node i.
    if not hear from node i **then**
      $\Theta = \Theta \setminus \{i\}$.
      Set node i in isolation mode $d_i = 0$.
until Terminated

level of node i at time t by Eqn. (1), denoted as $f_i[t]$. We substitute E_i^{used} in Eqn. (1) with $\sum_{j=0}^{t-1} \Delta eu_i \cdot a_i[j] + \lfloor \frac{t}{T} \rfloor E_c$, the sum of computation and communication costs.

- *Energy capacity constraint.* This constraint is to make sure that the predicted available energy of each node at every time step is within the designed energy capacity.
- *Execution constraint.* This is the execution constraint for computation. For each time step in a period (except the last step), in order to schedule an atomic task, the available energy should be no less than the energy consumption of executing a task.
- *Execution constraint with communication.* This constraint is a special case of the third one. At the last execution time step in a period, the available energy should be no less than the energy cost of the scheduled executions and the one-time communication cost, as the sensor nodes will communicate with the conductor after this time step.

Based on the optimization problem in Eqn. (3), we develop an adaptation algorithm for the conductor (Algo. 1). At each adaptation period, based on the importance score and the available energy of each node, the conductor predicts the energy creation rate and computes the decayed importance scores for the upcoming KT time steps, and then derives the optimal execution strategy for each node by solving the MILP problem and sends the execution strategy of first T time steps to nodes. It then waits for one adaptation period T and receive the new scores and energy availability from nodes. Note that as the energy creation rate is predicted, it is possible that some nodes may not have enough energy to send back the score and energy availability. Thus, for each adaptation period, only the nodes in set Θ, i.e., connected nodes, will be involved in the conductor adaptation. Nodes that do not have enough energy for communication will be put in an isolation mode (more details in subsection II-C next).



(a) Camera locations (b) Camera views

Fig. 3: Camera settings of VideoWeb Day1 dataset.

C. Sensor Node

Each sensor node is equipped with a super-capacitor of capacity E_c and an energy harvesting module. It receives the schedule of executions $a_i[t]$ and and executes accordingly based on the execution mode d_i. It consists of two modules: adapter and inference.

Adapter. At each adaptation period, a node will be in one of the two possible execution modes, indicated by d_i, where $d_i = 1$ denotes the in-network mode and $d_i = 0$ denotes the isolation mode.

(1)*In-network mode*. In this mode, a node has enough energy to communicate with the conductor. The conductor is able to schedule the execution ahead for the node. The node will execute according to the instructions from the conductor. At the beginning of each time step, if the node is instructed to execute an atomic task and there is enough energy to execute it, the node will do so. Otherwise, the node will remain idle. If no task is scheduled by the conductor, the node will remain idle too.

(2) *Isolation mode*. When a node does not have enough energy to communicate with the conductor and the conductor does not know the status of the node, no optimization can be done for the node. The node will execute as many tasks as it can based on its own energy availability. In order to recover from the isolation mode to the in-network mode, the node is required to accumulate the one-time communication energy before it starts executing greedily.

Inference. For each sensor node i, the inference module receives the captured images from the camera and performs an efficient inference on the importance of frames, given as a probability in $[0, 1]$. The importance of the frames is defined based on the application. For example, if we want to monitor deer activity in a habitat, we may defined the importance as whether there exists a deer in the scene.

III. Experimental Results

A. Experiment Setup

Scenarios. We evaluate AdaSens on a multi-view video dataset VideoWeb [15], which contains realistic scenarios in a multi-camera network that involves multiple persons performing different activities. We use the 8-scene Day 1 subset which monitors a courtyard with four views (Fig. 3). We transfer the given labels of important actions to binary indicators of important frames. A global ground truth that combines all important intervals across views is generated for evaluating the performance. We divide the whole area into $M = 12$ partitions with different areas $x_m, m \in [1, M]$ and

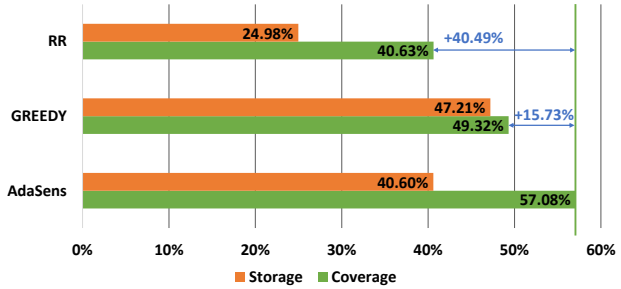


Fig. 4: Comparison on coverage and storage of AdaSens with GREEDY and RR baselines.

obtain the corresponding node covering indicators $\lambda_{i,m}, i \in [1, N], m \in [1, M]$.

Performance Measures. Similar to [16], [17], we consider a coverage metric at frame level, which evaluates how well the frames captured from the sensor nodes cover the important frames. The coverage is computed as the percentage of covered frames by all nodes in the global ground truth. If an important frame is covered by any of the nodes, it will be considered as true positive. Besides computation and communication, the intermittently-powered sensor nodes have limited storage too, so we also evaluate the storage for the processed frames.

Comparison Methods. We are not aware of any previous work on adaptive environment monitoring with intermittently-powered sensors targeting maximizing the coverage of the scene. Therefore, we compare our AdaSens with two baselines we implemented: a greedy baseline (GREEDY) and a round-robin baseline (RR). In GREEDY, every node performs as many tasks as possible given the available energy without coordination among nodes. RR method coordinates nodes to execute in turns.

Experimental Settings. We utilize solar profile from [14] for sensor nodes and use a time step of one minute. Similar to [6], we assume using the low-power OpenChirp network architecture [18] for nodes to send data over long distances. In AdaSens, only P_i and E_i are sent back to the conductor from the nodes, which yields a one-time communication cost of $E_c = 235mJ$. With low-power cameras [19], images can be taken at low energy as $10mJ/frame$. The energy cost of DNN inference on low-power chips can be as low as $26mJ$ [6]. An atomic task is to capture and infer on 10 frames, which yields an energy usage rate of $360mJ$ per task.

B. Comparison with Other Methods

Fig. 4 shows the coverage and the storage needed for captured frames of AdaSens and its comparison with other approaches. From the figure, we can see that:

- On average, AdaSens achieves a coverage of 57.08% of important frames with 40.60% of processed and stored frames given the limited and unstable power supply.
- Compared to GREEDY, AdaSens has 15.73% improvements of the coverage and 14% reduction of the storage.
- Compared to the round-robin method RR, AdaSens achieves 40.49% improvements of the coverage. The storage of the processed frames by RR method is lower than ours as it cannot adapt to the change of energy level and result in fewer important frames captured.

TABLE I: Impact of adaptation period on coverage.

T	10	20	30	40	50	60
Coverage(%)	54.15	55.52	57.08	53.56	54.26	49.14

TABLE II: Results using different day-time solar profiles.

Profile	GREEDY	AdaSens	Coverage gain
SP1	48.36%	58.33%	+20.62%
SP2	49.32	57.08%	+15.73%

C. Impact of Adaptation Period and Horizon

In Tab. I, we show the results on the impact of adaptation period T in AdaSens. We vary T from 10 - 60 minutes with $K = 1$. At first, with increasing adaptation period T, the coverage of important frames increases as it benefits from the following factors: larger T leads to less energy for communication and more energy for computation and longer optimization horizon. The coverage decreases when T gets too large: even with the benefit of longer optimization horizon, the performance degrades because of the less accurate solar energy prediction and the out-of-date important scores of views. We also investigate the impact of adaptation horizon on the coverage, and similar trend is observed. That is, with the increase of K, the coverage increases with a longer optimization horizon, but if K is too large, the performance degrades due to inaccurate energy prediction and out-of-date important scores.

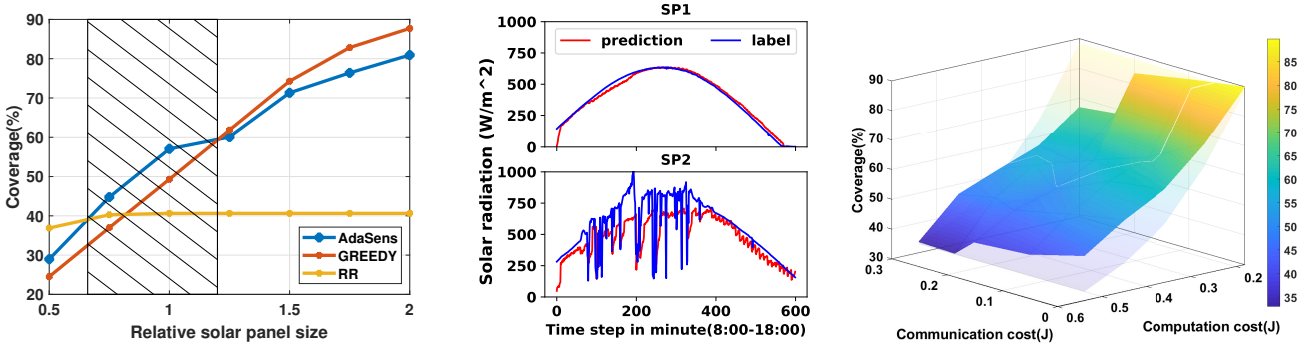
D. Impact of Energy Factors

Energy Availability. The energy availability may be different from what we expect when designing the system, which can be inadequate or overabundent. We study on the impact of energy availability by setting different solar panel sizes (Fig. 5a). We can see that AdaSens outperforms both baselines in the hatched area, which is the most common case in practice. When the energy availability is too inadequate, the RR method dispatches the limited available energy evenly and performs better. When the energy availability is overabundent, the GREEDY method performs better as it can do as much as possible.

Energy Prediction Accuracy. We study different solar profiles to investigate the impact of energy prediction accuracy. In Fig. 5b, we present two different day-time solar profiles: SP1 and SP2, selected from different days in the solar calendar. We can see that SP2 is less predictable as it fluctuates more (SP2 is used for all experiments above). The performance of AdaSens using SP1 and SP2 are shown in Tab. II. In this experiment, we only compare with GREEDY as it performs better than RR. When using the smooth solar profile, the coverage gain of AdaSens is +20.62%. While using the fluctuating solar profile, the coverage gain of AdaSens is +15.73%. It implies that with more predictable energy source, AdaSens performs better.

E. Study on Communication and Computation Costs

We run experiments with different combinations of communication and computation costs. Both costs are within a reasonable range for various computation cost [6], [20] and



(a) Impact of energy availability. (b) Examples of solar profiles. (c) Coverage with different settings.

Fig. 5: Impact of energy availability, examples of solar profiles and coverage with different settings.

communication cost [21] on intermittently-powered sensors. The result is shown in Fig. 5c. The translucent surface in the background is the GREEDY baseline. We observe that when the ratio of communication cost versus computation cost is smaller, our methods provides more benefits on coverage, and vise versa. This is reasonable as when the communication takes too much energy compared to the computation task, it is better not to communicate.

IV. Conclusion

We present a novel adaptive environment monitoring framework with intermittently-powered camera sensors, AdaSens, which optimizes the coverage of important frames by coordinating the sensor operations under intermittent and varying energy supplies. Experiments demonstrate significant improvements of AdaSens over baselines that do not explore system-level coordination and optimization.

Acknowledgements: We gratefully acknowledge the support from National Science Foundation awards CNS-2038853, CNS-1850496, IIS-1724341, CNS-1834701, and Office of Naval Research grant N00014-19-1-2496.

References

[1] J. Hester and J. Sorber, "The future of sensing is batteryless, intermittent, and awesome," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, 2017, pp. 1–6.

[2] Y. Qian, L. Yu, W. Liu, and A. G. Hauptmann, "Electricity: An efficient multi-camera vehicle tracking system for intelligent city," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 588–589.

[3] J. Boice, X. Lu, C. Margi, G. Stanek, G. Zhang, R. Manduchi, and K. Obraczka, "Meerkats: A power-aware, self-managing wireless camera network for wide area monitoring," in *Proc. Workshop on Distributed Smart Cameras*. Citeseer, 2006, pp. 393–422.

[4] X. Liu, P. Ghosh, O. Ulutan, B. Manjunath, K. Chan, and R. Govindan, "Caesar: cross-camera complex activity recognition," in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, 2019, pp. 232–244.

[5] Y. Wu, Z. Wang, Z. Jia, Y. Shi, and J. Hu, "Intermittent inference with nonuniformly compressed multi-exit neural network for energy harvesting powered devices," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[6] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 199–213.

[7] B. Islam and S. Nirjon, "Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 3, Sep. 2020.

[8] J. Hester, K. Storer, and J. Sorber, "Timely execution on intermittently powered batteryless sensors," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3131672.3131673

[9] K. S. Yıldırım, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester, "Ink: Reactive kernel for tiny batteryless sensors," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 41–53.

[10] Y. Wu, Z. Jia, F. Fang, and J. Hu, "Cooperative communication between two transiently powered sensor nodes by reinforcement learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[11] A. Colin, E. Ruppel, and B. Lucia, "A reconfigurable energy storage architecture for energy-harvesting devices," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 767–781.

[12] J. de Winkel, C. Delle Donne, K. S. Yildirim, P. Pawełczak, and J. Hester, "Reliable timekeeping for intermittent computing," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 53–67.

[13] R. Ibrahim, T. D. Chung, S. M. Hassan, K. Bingi, and S. Salahuddin, "Solar energy harvester for industrial wireless sensor nodes," *Procedia Computer Science*, vol. 105, no. C, pp. 111–118, 2017.

[14] A. Maxey, C.; Andreas, "Oak ridge national laboratory (ornl); rotating shadowband radiometer (rsr); oak ridge, tennessee (data); nrel report no. da-5500-56512." http://dx.doi.org/10.5439/1052553.

[15] G. Denina, B. Bhanu, H. T. Nguyen, C. Ding, A. Kamal, C. Ravishankar, A. Roy-Chowdhury, A. Ivers, and B. Varda, "Videoweb dataset for multi-camera activities and non-verbal communication," in *Distributed Video Sensor Networks*. Springer, 2011, pp. 335–347.

[16] S. Lan, R. Panda, Q. Zhu, and A. K. Roy-Chowdhury, "Ffnet: Video fast-forwarding via reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[17] S. Lan, Z. Wang, A. K. Roy-Chowdhury, E. Wei, and Q. Zhu, "Distributed multi-agent video fast-forwarding," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1075–1084.

[18] A. Dongare, C. Hesling, K. Bhatia, A. Balanuta, R. L. Pereira, B. Iannucci, and A. Rowe, "Openchirp: A low-power wide-area networking architecture," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017, pp. 569–574.

[19] S. Naderiparizi, Z. Kapetanovic, and J. R. Smith, "Wispcam: An rf-powered smart camera for machine vision applications," in *Proceedings of the 4th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, 2016, pp. 19–22.

[20] A. Montanari, M. Sharma, D. Jenkus, M. Alloulah, L. Qendro, and F. Kawsar, "eperceptive: energy reactive embedded intelligence for batteryless sensors," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 382–394.

[21] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith, and S. Gollakota, "Lora backscatter: Enabling the vision of ubiquitous connectivity," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, pp. 1–24, 2017.