Real-time In-network Image Compression via Distributed Dictionary Learning

Parul Pandey, *Member, IEEE,* Mehdi Rahmati, *Senior Member, IEEE,*
Waheed U. Bajwa, *Senior Member, IEEE,* and Dario Pompili, *Fellow, IEEE*

Abstract—Multi-camera networks are increasingly becoming pervasive in many monitoring and surveillance applications, and have attracted much attention in distributed systems with collaborative, real-time decision-making capabilities. While in-network data compression brings significant energy savings in camera nodes, signal representation using sparse approximations and overcomplete dictionaries have been shown to outperform traditional compression methods. In this work, an end-to-end and real-time solution is designed and implemented to enable energy-efficient and robust dictionary learning in distributed camera networks by leveraging the spatial correlation of the collected multimedia data. Traditional distributed dictionary learning relies on consensus-building algorithms, which involve communicating with neighboring nodes until convergence is achieved. Existing methods, however, do not exploit spatial correlations in camera networks for improved energy efficiency. In contrast, low-computational-complexity metrics are employed in this work to quantify and exploit the spatial correlation across camera nodes in a wireless network for efficient distributed dictionary learning and in-network image compression. The performance of the proposed approach is validated through extensive simulations on public datasets as well as via real-world experiments on a testbed composed of Raspberry Pi nodes.

Index Terms—Camera networks, dictionary learning, distributed consensus, image compression, mobile computing.

1 Introduction

Motivation: Camera networks are real-time distributed systems that cover large spaces and communicate over (wireless) networks to make decisions collaboratively. To enable surveillance applications, in-network processing of the data captured by any camera node is done using computationally cheap methods and the data is transferred to the server when an event is detected. At the centralized location the data is analyzed using computation-intensive computer vision algorithms [2]. Distributed camera networks have several advantages, including that the subject of interest can be observed from several angles, which helps resolve the problem of occlusion faced by individual cameras and avoid single point of failure [3]. At the same time, for computation-intensive applications, such as object detection on video data captured by camera nodes, distributed camera networks often move video data (generated throughout the time of operation at 30-60 fps) from battery-limited cameras in a network to a central server. This operation is costly (in terms of time and energy), reduces the mission lifetime of nodes, and is inherently unscalable.

These limitations call for efficient image-compression methods for transmission and storing of data in the network. Signal representation using sparse approximations and overcomplete dictionaries [4]–[7] have received considerable attention in recent years in the area of image feature extraction, data compression, and bit-rate reduction [8]–[10] for both storage and transmission. This technique has been shown to achieve dramatic improvement over JPEG/JPEG2000 compression techniques [9], [11], [12]. The goal of dictionary learning is to learn an overcomplete dictionary D such that data samples, represented as a matrix Y, are well approximated by no more than T_0 columns of D. Mathematically, the problem of dictionary learning can be expressed as,

$$(D, X) = \arg\min_{D,X} \|Y - DX\|_F^2 \ \text{ s.t. } \forall s, \|x_s\|_0 \leq T_0, \quad (1)$$

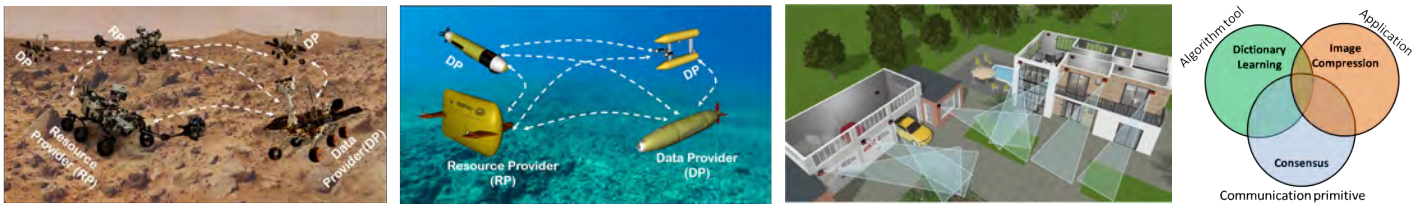
where $D \in \mathbb{R}^{n \times K}$ with $K > n$ is an overcomplete dictionary having unit l_2-norm columns, $Y \in \mathbb{R}^{n \times S}$ is the data available at a centralized location, $X \in \mathbb{R}^{K \times S}$ are the sparse coefficients of the data having no more than $T_0 \ll n$ nonzero coefficients per sample, and x_s represents the s^{th} column in X. Note that, while the dictionary-learning problem as formulated in (1) cannot be efficiently solved due to the presence of the l_0-norm, there exist several computationally efficient and theoretically optimal approximations of this formulation; see, e.g., [8], [13]–[18] and the references therein.

Once a dictionary is learned, each node can obtain a sparse approximation of the signal that can be used to compress the image captured by the nodes and consequently help save significant space when storing the data and energy when offloading data to a centralized location or to neighboring nodes. Consensus forms the communication primitive to be used between neighboring camera nodes for dictionary learning in a setting where the data is distributed such as in distributed camera networks [22]–[24]. In particular, consensus is an iterative process where the camera nodes communicate with their neighbors for a fixed number of iterations or until convergence. This iterative communication introduces an additional overhead of energy consumption on nodes in the network. Based on our experiments (discussed in more detail later in the article), we observed that executing distributed dictionary learning in a network of 10 nodes consumes 20% of battery capacity per node in the network. This figure is based on a lithium battery pack expansion board for Raspberry Pi with maximum battery capacity of 3.8 Ah [25].

Novelty: Our focus in this paper is on proof-of-concept of an

- *The authors are with the Department of Electrical and Computer Engineering, Rutgers University–New Brunswick, NJ, USA. Emails: {parul.pandey, mehdi.rahmati, waheed.bajwa, pompili}@rutgers.edu.*

- *A shorter version of this work appeared in the Proc. of IEEE International Conference on Autonomic Computing (ICAC), Trento, Italy, Sept'18 [1].*



(a) (b) (c) (d)

Fig. 1: Three different scenarios/environments where image compression using dictionary learning brings benefits in saving battery capacity of camera nodes in the network. Each sensor node can serve as a Data Provider (DP) or a Resource Provider (RP). The RP nodes have sophisticated computational capabilities and higher battery capacity than DP nodes. (a) Mars exploration with *Curiosity* rover [19]. (b) A network of Autonomous Underwater Vehicles (AUVs), which serve as DP and continuously capture raw data, while RPs—such as *Mesobot* and *Sentry* [20]—create bathymetric maps and perform sidescan of the seafloor by processing images. (c) Deployment of multiview cameras in a smart home [21]. (d) Image compression is a promising application for sparse representation in dictionary learning; as cameras are distributed, consensus algorithms are employed as the communication primitive to reach the agreement among the cameras.

end-to-end and real-time solution for energy-efficient and robust dictionary learning in distributed camera networks. To this end, our main contribution is development of a real testbed-based evaluation of a solution that exploits spatial correlation of the data collected by camera sensors. Many solutions exist that make use of dictionary learning at camera nodes for surveillance applications [26], [27]. In all these solutions, however, dictionary learning is only executed locally at each node, which is energy draining for the nodes. In our envisioned scenario, we consider a system of battery powered camera nodes communicating with each other over a wireless network to learn the dictionary. The learned dictionary is used to compress images captured by the cameras before sending to the remote server. An object-detection algorithm is run on these images at the server, where the object to be detected depends on the purpose of surveillance by the camera nodes. Our focus was to develop energy-efficient distributed dictionary-learning techniques that can then be used for compressing images captured by camera nodes and to increase the lifetime of the nodes. Figs. 1(a)–(c) show examples of various futuristic applications of the proposed distributed solution in various environments such as on Mars and in underwater environments as well as for consumer applications such as in smart home cameras. Our work lies at the intersection of three domains—dictionary learning, image compression, and consensus—and its goal is to bridge the gap between them to support a real-world application, as depicted in Fig. 1(d).

Approach: Image sensing is done via directional sensors and the visual area captured by these sensors is defined by the *Field of View (FoV)* of the camera nodes. In this article we utilize the spatial correlation in the data captured by camera nodes in the network to reduce the energy consumption of the network during distributed dictionary learning. To enable this approach we have to keep in mind the constraints of distributed networks in a real-world setting, i.e., the global knowledge of topology may not be available to the nodes in the network and the interconnection between nodes in the network cannot be described by a complete graph. We rely on computationally cheap metrics to quantify the correlation of data among camera nodes to enable long-term deployment of battery limited camera nodes. Distributed dictionary learning can be used to enable a variety of applications in a distributed camera network such as background modeling, object classification, and image denoising [23], [27]–[31]; in this article, we focus on in-network image compression. Finally, we will quantify the performance of images compressed using estimated dictionary on object detection algorithms.

Contributions: The following are our main contributions.

- We design and implement an end-to-end system to enable energy-efficient sensing of multimedia data at a local distributed camera network and computation of the data in a remote cloud infrastructure.
- We develop a novel protocol that identifies camera nodes in the network with correlated image data by relying on data-driven metrics and using only local knowledge of the topology available to the camera nodes.
- We quantify the trade-off in energy savings and accuracy loss of our solution on publicly available datasets and via experiments on a testbed of Raspberry Pis.

Article Organization: In Sect. 2, we present our pipeline and workflow for image compression using a dictionary-learning framework. In Sect. 3, we detail the design of our protocol that is proposed to bring energy savings to the distributed dictionary-learning algorithm executed in a camera network. In Sect. 4, we provide quantitative results that demonstrate the merits of our contributions via both simulations and testbed experiments. In Sect. 5, we cover the work done in the area of distributed systems employing consensus and analyze the difference of these approaches from ours. Finally, in Sect. 6, we conclude our article and outline future work.

2 Dictionary Learning and Consensus in a Distributed Camera Network

In this section, we present our envisioned scenario in a distributed surveillance system as shown in Fig. 1(c). We assume that the network consists of a set of resource-constrained camera nodes and limited battery capacity. Our scenario, as shown in Fig. 2, is that a network of local nodes perform dictionary learning collaboratively. The estimated dictionary is then used for compression of incoming images. The compressed images are then sent to the cloud for storage and further analysis on these images are done on the cloud (such as person or object detection) [32], [33]. We now present two application workflows used in this article to realize our above envisioned scenario, namely, distributed dictionary learning and image compression, and the different tasks implemented within these workflows.

Assume N camera nodes used for surveillance form an ad-hoc network and communicate over a wireless network. Any wireless communications technology such as Bluetooth, WiFi Direct, 802.11 ad-hoc mode, or Near-field Communication (NFC) can be used in this scenario without relying on any existing fixed infrastructure. Two camera nodes are considered neighbors in a certain time window if they can communicate with each other. Each node is capable of storing some local data, performing

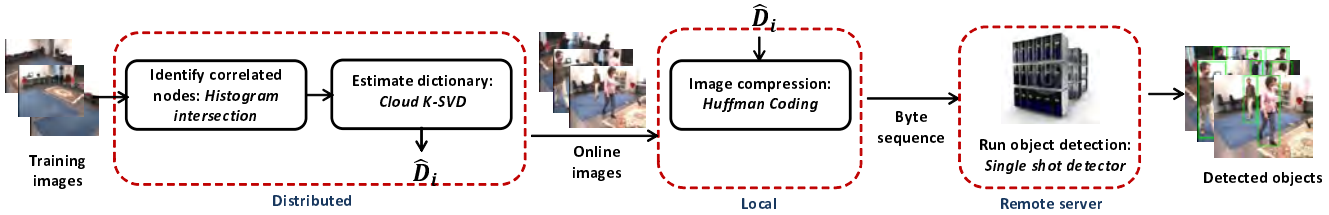


Fig. 2: Our envisioned pipeline for image compression using a dictionary learned collaboratively and the compressed images being sent to the cloud for storage or further processing. The figure shows different tasks of this pipleine and the location where they are implemented.

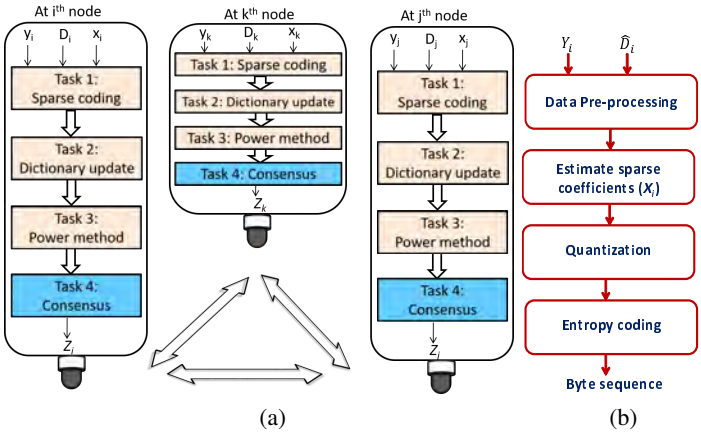


Fig. 3: (a) Workflow for distributed dictionary learning executed at each camera node in the network. Task 1, 2, and 3 are executed locally at each node; whereas Task 4, i.e., consensus, involves communication among nodes that are in communication range; (b) Workflow for image-compression application that uses the estimated dictionary to calculate the sparse coefficients; this workflow is executed locally at each node.

computations on it, and exchanging messages with its neighbors. Next, we assume each node i, where $i \in [1 \cdots N]$, has a collection of local data, expressed as a matrix $Y_i \in \mathbb{R}^{n \times S_i}$, with S_i representing the number of data samples at the i^{th} node. In case of distributed camera networks, each camera node is taking images of a scene from a different angle, which form the data samples at the i^{th} node. We can express all this distributed data into a single matrix $Y = [Y_1 \cdots Y_N] \in \mathbb{R}^{n \times S}$, where $S = \sum_{i=1}^{N} S_i$ denotes the total number of data samples distributed across the N nodes.

We now cast the dictionary-learning problem: assume that data Y is available at a centralized location, the problem of learning an overcomplete dictionary D can be expressed as in (1), where $X \in \mathbb{R}^{K \times S}$ are the sparse coefficients of the data having no more than $T_0 \ll n$ nonzero coefficients per sample, and x_s represents the s^{th} column in X. In a distributed setting, the goal of dictionary learning is to have individual nodes collaboratively learn dictionaries $\{\widehat{D}_i\}_{i \in N}$ from global data Y such that these dictionaries be close to a dictionary D that could be learned from the global data Y in a centralized manner. We now present the tasks of this algorithm, as described in [23]—see also Fig. 3(a).

Task 1–Sparse coding: Each node i uses the local estimate $\widehat{D}_i$ of the centralized dictionary D and calculates the sparse coefficients of its local data by solving the following equation without collaborating with other nodes.

$$\forall s,\ \tilde{x}_{i,s}^{(t)} = \arg\min_{x \in \mathbb{R}^K} \|y_{i,s} - \widehat{D}_i^{(t-1)} x\|_2^2 \ \text{s.t.} \|x\|_0 \leq T_0, \quad (2)$$

where $y_{i,s}$ and $\tilde{x}_{i,s}^{(t)}$ represent the s^{th} sample of the data and its coefficients at node i at the t^{th} dictionary learning iteration. It is worth noting here that the sparse coding task as formulated in (2) is known to be NP-hard. Nonetheless, there exist several variations of this formulation, many of which are based on fast greedy strategies, that approximate (2) in a computationally efficient manner [34].

Task 2–Dictionary update: Dictionary update stage involves computing the dominant left (u_1)- and right-singular (v_1) vectors of the "reduced" error matrix $\widehat{E}_{i,k,R}^{(t)}$. The *Cloud K-SVD* algorithm of [23] defines $\widehat{d}_{i,k}^{(t)} = u_1$ and $\widehat{x}_{i,k,R}^{(t)} = \widehat{d}_{i,k}^{(t)^T} \widehat{E}_{i,k,R}^{(t)}$, where u_1 is the dominant vector of a matrix $\widehat{M}^{(t)}$ that is formally described in [23]. We need to only worry about calculating u_1 collaboratively. To this end, at each node, $\widehat{M}_i^{(t)} = \widehat{E}_{i,k,R}^{(t)} \widehat{E}_{i,k,R}^{(t)}$ is calculated. Our goal now is computing the dominant eigenvector of $\widehat{M}^{(t)}$ in a collaborative manner. In order to estimate the eigenvectors in the distributed network, Cloud K-SVD algorithm uses the distributed power method.

Task 3–Power method: Power method is an iterative procedure used to estimate the eigenvectors of a matrix. Power method is run for a fixed number of iterations or until convergence. Cloud K-SVD algorithm is interested in distributed power method as M_i's are distributed in the network. To this end, each site calculates $z_i = \widehat{M}_i^{(t)} \widehat{q}_i^{(t_p-1)}$ locally, where $\widehat{q}_i^{(t_p-1)}$ denotes an estimate of the dominant eigenvector of $\widehat{M}^{(t)}$ at i^{th} site after $(t_p - 1)$ power method iterations. We initialize each site with the same value (q^{init}). In Cloud K-SVD, one of the ways that this can be achieved is if each node uses the same seed for a random number generator. Next, the sites collaborate to calculate $\widehat{v}_i^{(t_p)} = \sum_i \widehat{M}_i^{(t)} \widehat{q}_i^{(t_p-1)}$ at each site. The normalized $\widehat{v}_i^{(t_p)}$ is an estimate of the dominant eigenvector of $\widehat{M}^{(t)}$.

Task 4–Consensus averaging: To calculate the approximation of $\sum_i \widehat{M}_i^{(t)} \widehat{q}_i^{(t_p-1)}$, nodes in the network make use of the distributed consensus technique. Each node is initialized as $z_i^{(0)} = \widehat{M}_i^{(t)} \widehat{q}_i^{(t_p-1)}$ at each node. We now give an example of how the consensus-averaging problem in a network can be defined. Let $Z^{(0)} = [z_1^{(0)}, \cdots, z_N^{(0)}]^\top$ be the initial value at each node. Each node achieves perfect consensus averaging as $t_c \to \infty$, where t_c is the number of consensus iterations, and obtains $Z_{i,T}^{(\infty)^\top} = \frac{1}{N} \sum_{j=1}^{N} z_j^{(0)} = \frac{1}{N} \sum_{j=1}^{N} \widehat{M}_j^{(t)} \widehat{q}_j^{(t_p-1)}$.

We take advantage of the wireless medium being inherently broadcast and hence use a broadcasting-based consensus method proposed in [35]. We use the asynchronous time model, which is well matched to the distributed nature of sensor networks. In this model, each sensor node is assumed to have a clock that ticks independently according to a Poisson process; consequently, the inter-tick times are exponentially distributed and independent across nodes and over time [35]. The asynchronous broadcast consensus assumes each node i broadcasts its own data to its $\mathcal{N}_i$ neighboring nodes within its communication range. The neighbors, which received the data, update their data according to the

weighted average of their current data as follows,

$$z_k^{(t_c+1)} = \gamma z_k^{(t_c)} + (1-\gamma) z_i^{(t_c)}, \forall k \in \mathcal{N}_i, \quad (3)$$

where $\gamma \in (0,1)$ stands for the mixing parameter, which weights the data values from each node, and $\mathcal{N}_i$ denotes the neighboring nodes of the i^{th} node. The remaining nodes in the network update their values as,

$$z_k^{(t_c+1)} = z_k^{(t_c)}, \forall k \notin \mathcal{N}_i. \quad (4)$$

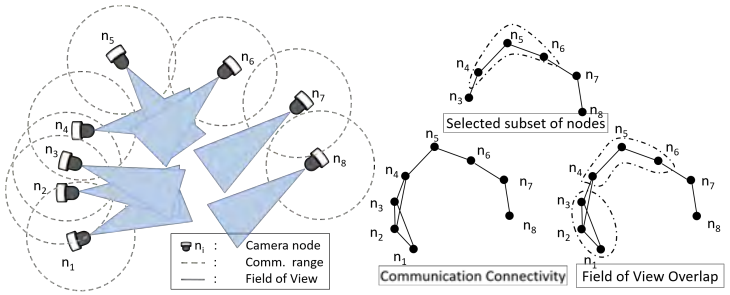
Image compression: In Fig. 3(b), we show the workflow for image compression algorithm. Once a dictionary has been learned by the algorithm explained above, it can be used in an image compression scheme in the following way. At the i^{th} node, for a new incoming image or an image that is to be sent to the remote server, we first form the set of vectors from non-overlapping patches of the image. This is denoted by $\{y_j\}_{j=1}^{L}$ or Y, where L is the number of patches. Next, we estimate the sparse approximation X of Y using the estimated dictionary $\widehat{D}_i$. The non-zero coefficients are quantized, for which a uniform quantization scheme along with thresholding is used [36], [37]. The quantized X matrix is then entropy coded to form a bit sequence. The compression is done by finding the sparse coefficients, which are then vector quantized before transmission. We can change the number of sparse coefficients to impact the amount of data in terms of number of bits transmitted.

3 Energy Efficient Dictionary Learning

In the last section, we discussed the scenario of how local distributed network and the remote cloud network can combine their sensing (local camera nodes) and computational (remote cloud) capabilities to implement a distributed surveillance system. We also introduced a distributed dictionary learning algorithm to accomplish this task. However, distributed dictionary learning is an energy-intensive technique, which reduces the battery capacity of cameras and consequently the mission lifetime of the network nodes. Our goal in this section is to present a solution to reduce the energy consumed by the nodes to perform dictionary learning in a camera network. To this end, we utilize a low-computational metric to identify nodes in the network with correlated data. We design a novel protocol that identifies such nodes. We also explain the challenges associated with identifying these nodes and provide novel solutions to address these challenges.

3.1 Overview of the Proposed Solution

We now explain how our proposed solution is executed in a sensor network to reduce the energy consumption of the network when distributed dictionary learning algorithm is run at the nodes. The aim of the proposed method is to reduce the energy consumption of the network when distributed dictionary learning algorithm is run at the camera nodes. To this end, we first identify only those nodes in the network that have correlated multimedia data. In Sect. 3.3 we present an algorithm that uses features of camera sensors to identify camera nodes with correlated data by utilizing the attributes of the camera nodes. However, choosing only a certain subset of nodes and silencing or shutting down other nodes can cause the network to be disconnected. This problem is further exacerbated because, as explained in Sect. 3.4, global knowledge of topology of the network may not be available to individual nodes. To this end, we present an algorithm in Sect. 3.5 to select a subset of nodes without disconnecting any node from the network.



(a) (b)

Fig. 4: (a) Field of View (FoV) and the communications range of the cameras for scenario that was mentioned in Fig. 1(c)—as an example, the FoVs' of nodes n_1–n_3 are overlapping, but the FoVs' of n_7–n_8 have zero overlap although they are in the communication range of each other; (b) Connectivity of the cameras, based on their communication range and FoVs' overlap. These overlapping FoV nodes are shown with the dashed lines in the figure. The subsets of nodes are selected in such a way as to leverage the overlap of the neighboring nodes.

With these subset of nodes we perform energy efficient distributed dictionary learning.

3.2 System Model

We consider a scenario with a number of heterogeneous camera nodes as in Fig. 1(c). We divide these camera nodes into two categories, namely, *Data Provider (DP)* and *Resource Provider (RP)*. The DP nodes are *dumb nodes* in the sense that they are severely resource constrained and have limited computational capabilities. The RPs, on the other hand, have sophisticated computational capabilities and higher battery capacity than DPs. In a homogeneous environment, where there is no difference between the hardware capabilities of devices, we can assume that the role of RPs can be randomly selected or given to nodes that have residual battery capacity over a certain value.

3.3 Quantifying Spatially-correlated Nodes

To reduce the energy consumption of the network during distributed dictionary learning, we identify nodes in the network that have correlated data. Furthermore, we silence these nodes with correlated data, i.e., these nodes do not participate in the consensus process, which results in saving the battery capacity of the network and will extend the mission lifetime of the camera network. To this end, we utilize the Field of View (FoV) of the camera nodes, which refers to the directional view of a camera sensor and is assumed to be an isosceles triangle (two-dimensional approximation) [38].

To identify camera nodes with spatially correlated data we utilize the following attributes of the camera nodes, (i) camera configuration, i.e., area of overlap between FoV of two cameras and (ii) data collected by the nodes. The former technique is more helpful when the nodes have been deployed with known FoV whereas the latter is helpful when the camera configuration is not known in advance. We also allocate a fixed energy budget for the RPs for this phase, i.e., we assume that each RP can only spend a pre-defined E_{setup} [Wh] to identify spatially correlated nodes in its neighborhood. This is done to keep the overhead of our solution low. We now explain in detail how these two techniques are implemented.

Field of View (FoV): Since the FoV of cameras is limited to the area they observe, the information they get is directly related to the directional sensing and configuration of the cameras. Assume a camera's FoV is described by $(P, R, \vec{V}, \alpha)$ as in [38], in which

P stands for the location of the camera, R represents the sensing radius, $\vec{V}$ indicates the sensing direction (i.e., the center line of sight of the camera's FoV), and α is the offset angle. Focal length of each camera, its location, and its sensing direction (shown by f, P, and $\vec{V}$, respectively) can be estimated as shown in [39]. A model for the spatial correlation can be derived based on the above parameters as can be seen in [38]. However, this information is not always available or may change over time.

Similarity metric to identify correlated nodes: In many cases, specially in unknown environments, the coordinate/specification or configuration of the cameras is not known. This can happen when camera nodes are deployed randomly or if the configuration has changed over time. If the camera configuration is not available we are motivated to use the data captured by the camera to identify the nodes in the network that have correlated data. Each DP identifies the RP node with which it will communicate based on metrics such as max received signal strength or residual battery capacity. Next, the RPs have to identify which of the DPs are spatially correlated and for that the DPs send a histogram of their collected online image data; such payload would include {`nodeID`, H_{nodeID}}. We choose a normalized similarity metric $S(i,j)$ that gives the *intersection between the histograms of two images* [40]. This metric is used to quantify data correlation between different camera nodes as,

$$S(i,j) = \frac{\sum_{k=0}^{b-1} \min(H_i^k, H_j^k)}{\sum_{k=0}^{b-1} H_i^k}, \tag{5}$$

where the histogram H_i^k of node i is a b-dimensional vector, and b is the number of bins in the histogram with k as the index of array H. The data collected by the cameras are in the RGB space. Therefore, we calculate $S(i,j)$ by first computing the histogram with b equal to 10 bins of each color channel separately for images of both nodes i and j. Now, by using (5), we calculate the intersection of histogram of each channel separately for the two images. This algorithm in [40] identifies the number of pixels that are found in both images and then takes an average across the channels.

3.4 Challenges for the Selection of Correlated Nodes

We now present challenges associated with selecting correlated camera nodes in the network. These challenges mainly arise because the global knowledge of the topology of the network may not be available. To this end, we identify two types of nodes, namely, critical nodes and nodes with overlapping field of view.

Critical nodes: A node is defined as a *critical node* if, when removed from the network, it will cause any of its neighboring nodes to be disconnected from the rest of the network. For example, in Fig. 4(a) n_7 is critical to node n_8 because when node n_7 is removed, n_8 cannot communicate with the rest of the network. We identify the nodes that have only one neighbor as the end nodes (n_e). Any of the end nodes cannot be critical because removing them will not make the network disconnected. A node in the distributed camera network can qualify as either a critical or non critical node. We would like to point out that when the spatial correlation between camera nodes is 1, the proposed method can be equivalent to the case where no data correlations exists among the nodes, i.e., dictionary learning happens at each individual node in the network.

Nodes with overlapping field of view: We make a note here that the nodes that have overlapping FoVs have spatially correlated data, but may not necessarily be within the communication range of each other. The nodes that are in the same FoV but not in the communication range of each other cannot know about each others presence and hence both nodes will remain active to participate in the consensus process. For example, in Fig. 4(b), we see that nodes n_1–n_3 have overlapping FoVs; as a result these nodes have correlated data and we can pick one of these nodes to participate in the consensus process. Similarly, while nodes n_4–n_6 have overlapping FoVs, they are not in the communication range of each other and have only n_5 as common node; as a result, they know of each others presence only via node n_5.

To identify the presence of nodes in communication range of each other, each RP broadcasts the data packets that it has received from its neighboring RP. This will be only done if the RP has not exhausted its energy budget (E_{setup}). To reduce the bandwidth and energy required to identify correlated nodes and prevent flooding the network, each node will communicate with only its neighbors. Also, identification of correlated nodes will only be done once after the network is set up.

3.5 Selecting a Subset of Nodes

To save the battery capacity of the network we select a subset of nodes in the network. To this end, we identify nodes in the network that have correlated data. Since only a subset of nodes are run, this leads to saving the energy of the network and increasing the mission lifetime of the network. As discussed in the last section, two challenges in distributed camera networks are that nodes might not be in field of view of each other or they may not be in communication range of each other. These two challenges can exist together or individually at each node. Identifying nodes with correlated data can only be achieved using local knowledge of the topology available to the nodes because construction of global topology of the network is energy and time consuming. We will later also quantify the battery capacity that is saved by using our proposed methodology of selecting of subset of nodes to do distributed dictionary learning in comparison to existing distributed dictionary learning approaches.

Neighborhood discovery: Neighborhood discovery is a two-step process where each camera node identifies its own neighbors as well as the neighbors of its neighbors. To this end, each node broadcasts a packet with the following information: node ID and residual battery capacity {`nodeID`, `ResBatCap`}. Once each node receives the information from its neighbors, it updates the set of neighboring nodes given by $\mathcal{N}_{nodeID}$. It then broadcasts this packet payload given by: {`nodeID`, $\mathcal{N}_{nodeID}$} Upon receiving this packet, the neighboring node updates its data structure, which stores information about the neighbors of its neighbors.

Identifying critical nodes: We present three steps that each node can use based on the available local information to identify if it is a critical node. As mentioned earlier, the global knowledge is not always available and our proposed approach focuses on circumventing this constraint. For illustration, we consider two neighboring nodes, n_i and n_j, and assume the following:

1) All RPs are considered critical nodes;
2) If a node has no neighbors other than critical nodes and is not an n_e node, then it is identified as a critical node;
3) If the neighbors of node n_j are a subset of node n_i's neighbors, then node n_i becomes critical to node n_j.

Various works have been done in this area that use distributed depth first search (DFS) and other spanning tree-based algorithms to find critical nodes in the network [41], [42]. These methods require passing multiple messages and increase network traffic

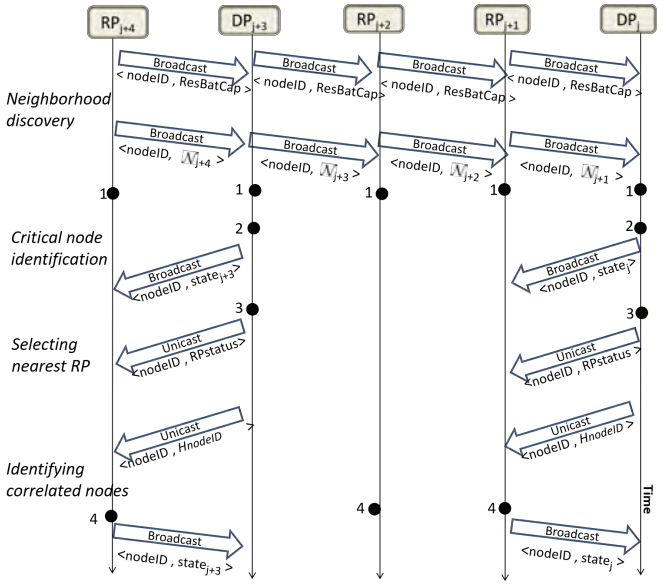


Fig. 5: Illustration of the proposed protocol to reduce energy consumption of the nodes in the network to execute distributed dictionary learning. Tasks performed locally at each node are shown using black dots. Task 1 identifies neighbor of each node in the network, Task 2 involves critical node selection procedure at the DPs, Task 3 involves selecting the nearest RP of each DP, and Task 4 involves identifying the spatially-correlated nodes.

and total energy consumed by the nodes. We, on the other hand, use only local knowledge available to the nodes to find the set of critical nodes. The conditions mentioned above may not identify the critical nodes of the network as the global knowledge of the topology of the graph is not available to the nodes. They, however, help each node in making a local decision if it is a critical node to its neighboring nodes and if removing the node, its neighbors will become disconnected from the network.

Communication protocol: Fig. 5 illustrates the timing diagram for the set of nodes and the corresponding tasks. We consider two DPs that are exchanging data with the chosen RPs. A signal is broadcast from RPs and the DPs acknowledge the signal with the requested information. Our first goal is to identify nodes whose collected data is highly correlated by utilizing the techniques mentioned earlier. When each node has identified the node to which it is spatially correlated, they inform each other. One of the nodes among the pair of spatially-correlated nodes is considered as the member of the subset of nodes selected to execute distributed dictionary learning. The nodes among the group that is spatially correlated are members of the subset. If there are multiple nodes with data correlation higher than a threshold, then only one of those is active and the others do not participate in the process. If one or more nodes among the group of nodes is critical, then the critical nodes are part of the subset of nodes that participate in the process. This is because we have to make sure that the topology in the subset remains connected, otherwise some of the nodes may not be able to take part in the consensus process. If none of them is critical, then the node with the highest number of neighbors within its communication range is in the subset. The packets forwarded by RP to its neighboring RP consists of {`RPnodeID`, `DPnodeIDs`, H_i, H_j}.

4 Performance Evaluation

The goal of this section is to present the performance of distributed dictionary learning algorithm in terms of energy consumed and accuracy achieved using our proposed protocol. We present our results on publicly available multiview camera datasets and dataset collected via experiments. We also discuss experimental results obtained on a testbed composed of Raspberry Pi nodes to study the performance of the proposed solution in a real-world setting.

Metrics: Our goal in this paper is to understand the impact of using dictionary learning algorithm for image compression on a real-world application like object detection. To this end, we have used two metrics, namely, mean squared error (MSE) and precision. MSE is used as a metric to show the performance of our algorithm after compressed image is reconstructed at the server for object detection. Furthermore, we use "precision" to measure the accuracy of running an object detection algorithm on compressed images. We analyze dictionary-learning performance using the following metrics.

- *Mean Squared Error (MSE)*: The pixel-wise mean squared difference between the image reconstructed via estimated dictionary/sparse coefficients and the original one.
- *Area Under Curve (AUC)*: The area under the Receiver Operative Characteristic (ROC) curve, which measures the object-detection accuracy.
- *Energy*: The energy consumed by the Raspberry Pi devices, in Wh.

4.1 Public Dataset

We first present our results on a public dataset. We use the Multi-camera Pedestrian Videos dataset [43] by The Ecole polytechnique federale de Lausanne (EPFL). We will discuss the different steps required to prepare the data for distributed dictionary learning, cost of dictionary learning, and performance of distributed learning on this dataset and using our proposed solution.

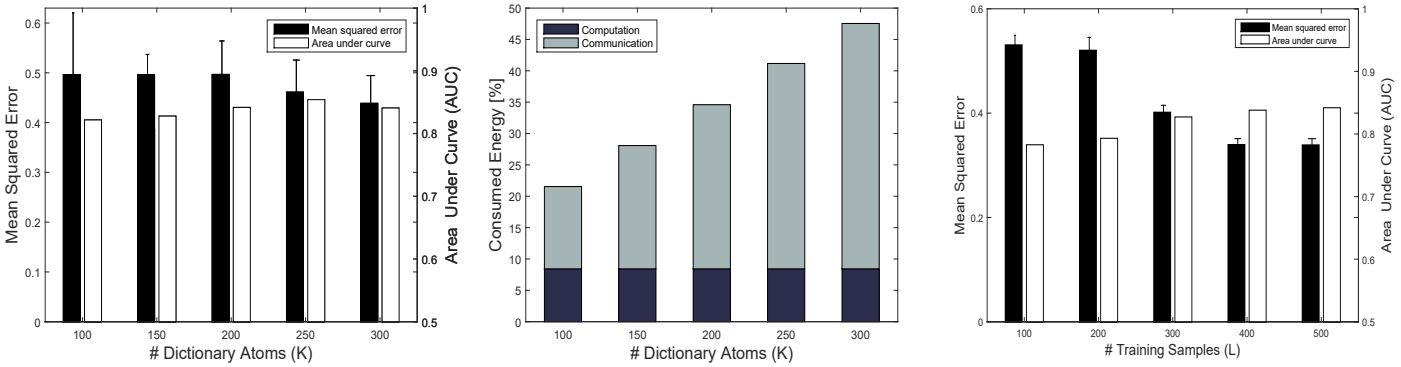
Datasets: The dataset [43] consists of three different scenarios captured by four digital video cameras placed in different locations. The video format is DV PAL, downsampled to 360×288 pixels at 25 fps. The dataset consists of 4 different locations and we show two example sequences from the dataset in Fig. 6. Unfortunately, the dataset does not provide any information about the camera configuration; hence, we identify nodes that are spatially correlated by executing (5) on data captured by these nodes.

Data preprocessing: We take non-overlapping 8×8 patches from each image frame, convert each patch to a one-dimensional vector. This vector becomes a column of data matrix Y. We take multiple such samples from different image frames and place them at random column indices. By taking a fixed number of patches from the original image we create a new image that is lower in size than the original one. Running dictionary learning algorithm on this new image, instead of the original image, significantly reduces the computational cost at a camera node. We also preprocess images by first converting each of them to gray scale with pixel values in the range 0 to 255, subtracting the mean from each image, and normalizing each image using 2-norm.

Energy savings vs accuracy tradeoff: The existing work in the development of dictionary learning algorithms have focused on maximizing the performance of the algorithm in terms of accuracy achieved by the algorithm. Our work, on the other hand, focuses on identifying "good-enough" parameters, i.e., parameters that give acceptable accuracy in each frame of the video with significant savings in time and energy as these algorithms have to be run on resource-constrained devices. To this end, we study the accuracy-energy tradeoff of various parameters in the dictionary learning algorithm and use it to select the parameters that can be used in

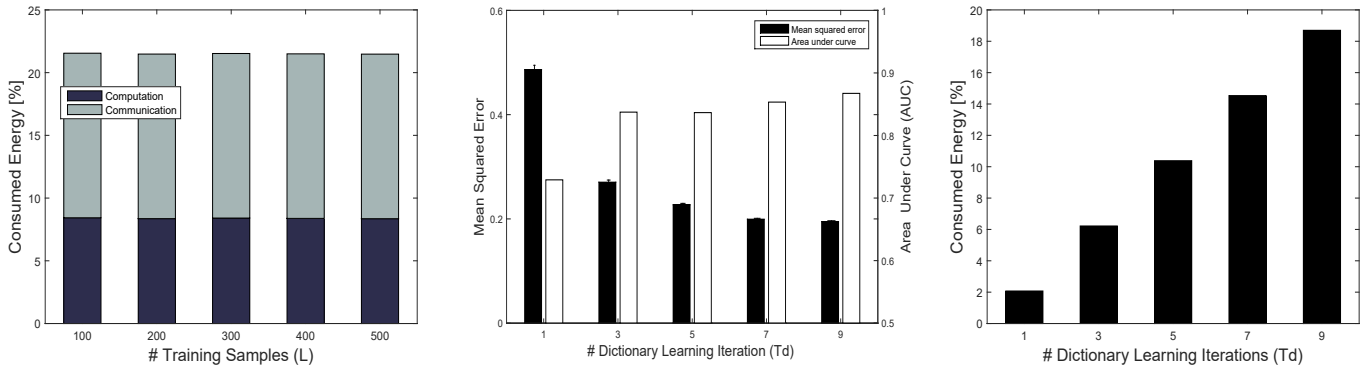
(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 6: Example scenarios—(a-d) room and (e-f) terrace—at a particular time slot taken from cameras with varying viewing angles and field of view. These images are part of a multiview dataset [43] and are used in our experiments to study the energy-efficiency performance of distributed dictionary learning.



(a) (b) (c)

Fig. 7: **Public dataset:** Accuracy-energy tradeoff of different parameters of distributed learning algorithm. (a) Accuracy obtained by varying number of atoms (K) in a dictionary. Left y-axis is mean squared error and Right y-axis is the object detection accuracy in terms of area under curve of ROC curve; (b) Energy consumed by the dictionary learning algorithm by varying number of atoms in a dictionary; (c) Accuracy obtained by varying number of training samples in the data (L). Left y-axis is mean squared error and Right y-axis is the object detection accuracy in terms of area under curve of ROC curve.

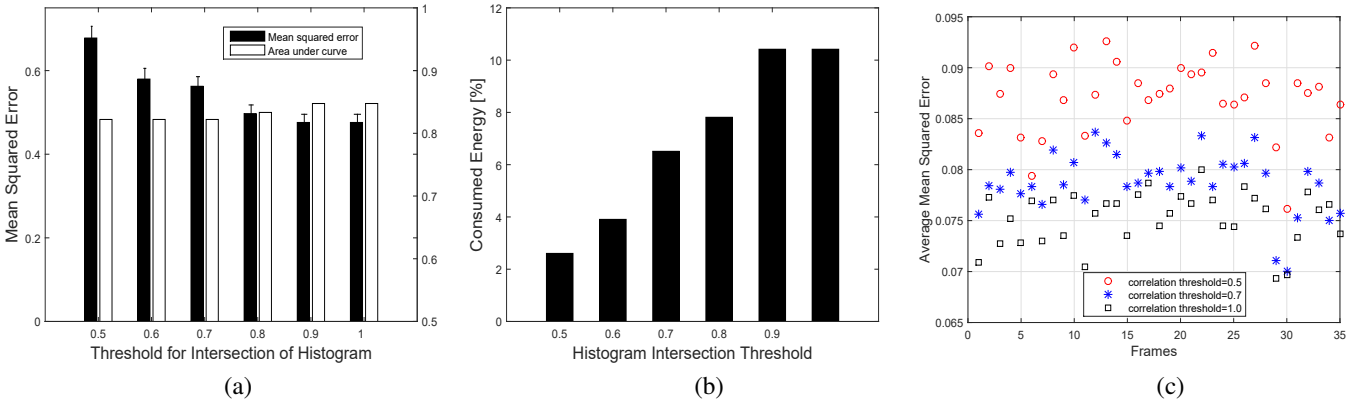


(a) (b) (c)

Fig. 8: **Public dataset:** (a) Energy consumed by the dictionary learning algorithm by varying number of training samples (L) in the data; (b) Accuracy obtained by varying the number of dictionary learning iterations (t_d). Left y-axis is mean squared error and Right y-axis is the object detection accuracy in terms of area under curve of ROC curve; (c) Energy consumed by the dictionary learning algorithm by varying the number of dictionary learning iterations.

the implementation of dictionary learning algorithm on a resource-constrained device for a real-world application. To this end, we study the impact of the following three parameters, namely, dictionary size or number of atoms in the dictionary (K), number of training samples (L), and dictionary learning iterations (t_d), on the performance of these algorithms. We show the performance of dictionary learning algorithm using the MSE metric. Since our goal is to compress image and process them at the server for detecting authorized or unauthorized objects of interests, we study the object detection accuracy achieved at the server. To study the performance of object detection algorithm, we use Single-shot detector [44] convolution neural network and use AUC metric to quantify its performance. To measure the energy consumed by a Raspberry Pi device when running the distributed dictionary learning algorithm, we use a power monitor, PortaPower $3-20$ V Dual USB Power Monitor. This device displays current, voltage, and amount of current received by Raspberry Pi devices while our dictionary learning algorithm runs.

Number of atoms: We observe that as we increase the number of dictionary atoms, the performance of the dictionary learning algorithm increases. Fig. 7(a), the left y-axis shows the MSE of dictionary learning algorithm and right y-axis shows the object detection accuracy. We see that the MSE reduces by 1.33% and the object detection accuracy increases by 2% as the number of dictionary atoms are increased from 100 to 300. At the same time, however, there is significant increase in energy costs as shown in Fig. 7(b). The energy cost increases by 45% as the number of dictionary atoms are increased from 100 to 300. This increase in energy costs is due to the increase in communication cost that comes with the increase in number of dictionary atoms. Recall that nodes in the network communicate with each other to update each atom of the dictionary sequentially. Therefore, as



(a) (b) (c)

Fig. 9: **Public dataset:** (a) Average mean squared error achieved in distributed dictionary learning when nodes with spatially correlated data (above a threshold) are identified and one of them is selected along with nodes which do not have correlated data with any other node; (b) Energy savings obtained in the dictionary learning process by removing spatially correlated nodes which have histogram intersection value above a threshold (given in the x-axis); (c) Average mean squared error (y-axis) achieved over incoming test frames (x-axis) by using dictionaries learned by using training dataset that has removed data that is spatially correlated with other nodes;

the number of dictionary atoms increases, the communications cost in the network also increases. Among the parameters we have discussed above, the size of dictionary, i.e., number of atoms in the dictionary, impacts the cost of image compression at run-time, i.e., when the dictionary has been learned and is being used for image compression. The other parameters explained above, namely, number of training samples and number of dictionary learning iterations impact only when we are estimating the dictionary, which is not at run time.

Number of training samples: We observe that as we increase the number of training samples, the performance of the dictionary learning algorithm increases. In Fig. 7(c), we see that the MSE reduces by 50% and the object detection accuracy increases by 10% as the number of training samples are increased from 100 to 400. The object detection rate rises steeply and then saturates. However, since increasing the number of training samples only impacts the sparse coding task and that is a task executed locally at the sensor nodes (Task 1 in Fig. 3). This only impacts the computation energy and does not impact energy required for communication. As a result it does not significantly increase the energy consumed, as can be seen in Fig. 8(a).

Number of dictionary-learning iterations: In Fig. 8(b), we see that as we increase the number of dictionary learning iteration from 1 to 9, the MSE decreases by 62% and the object detection accuracy increases by 16%. However, this comes at the cost of significant increase in energy as shown in Fig. 8(c). The increase in cost is quantified in Fig. 8(c), which shows the cost increases by 50% as the number of dictionary learning iterations are increased from 1 to 9. This increase in energy costs is due to increase in communication costs with increase in number of iterations.

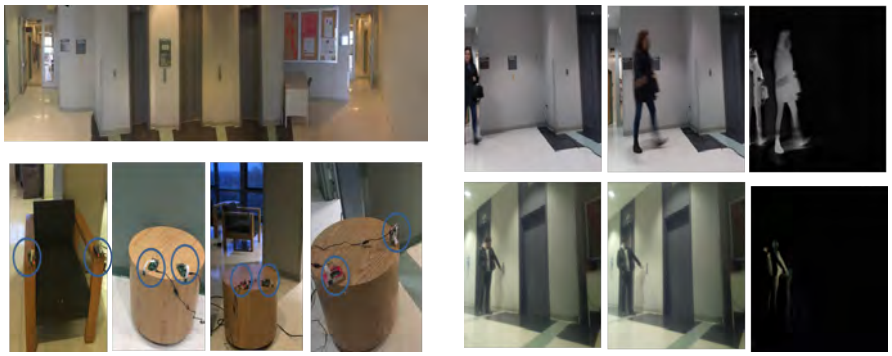
Spatial correlation: We consider the dataset in [43], which consists of four cameras under different settings. In this scenario, since no information is given about the camera configuration and the nodes are located in an indoor environment, we assume the nodes are fully connected. However, we do not make any assumption about the spatial correlation of data captured by these nodes. We vary the threshold of correlation coefficient and identify the nodes that capture data with correlation above the threshold. We use the intersection of histogram metric mentioned in (5) to quantify the spatial correlation between training data at different nodes. Each node creates a set that includes itself and the other nodes with which it has correlated data above the threshold. We randomly select one of the nodes from this set. Nodes that do not have data correlated with any other node are also included in the dictionary-learning process.

In Fig. 9(a), the left y-axis shows the MSE of dictionary learning and the right y-axis shows the object-detection accuracy. As the threshold for spatial correlation changes from 1.0 to 0.7, we get an accuracy loss of 15% and the object-detection accuracy falls by 2.3%. In Fig. 9(b), we show the performance when the dictionary is estimated and the data is selected at each node using different values of threshold for intersection of histogram metric (5). We first identify nodes that have data with correlation above a pre-defined threshold. Only one of the nodes among these correlated nodes participate in the dictionary learning process. We note that the dictionary created using lower threshold coefficient performs worse in terms of MSE; interestingly, however, the performance remains constant over the next 35 incoming frames for which the tests are performed. In Fig. 9(c), we show the energy consumption of the nodes reduces by 75% when the threshold for spatial correlation changes from 1.0 to 0.7.

4.2 Testbed Experiments

We now give details for our experimental setup used to test the performance of distributed dictionary learning algorithm in terms of accuracy and energy consumption of the nodes. We also study the benefits obtained by our proposed solution compared to existing distributed dictionary learning methods. We now present our envisioned scenario in a distributed surveillance systems as shown in Fig. 1(c). We assume that a set of resource-constrained nodes with cameras as sensors and limited battery capacities are used. Our scenario as shown in Fig. 2 is that a network of local nodes perform distributed dictionary learning. The estimated dictionary is then used for compression of incoming images. The compressed images are then send to the cloud for storage or person detection or detection of unauthorized objects (such as detection guns and alarming the building security).

Experimental setup: Our testbed consists of 10 Raspberry Pi nodes, which are small single-board computers with the following characteristics—Model 3B, 1.2 GHz 64-bit quad-core processor, 1 GB RAM and with Wireless LAN and Bluetooth capabilities. For our experiments we use Raspberry Pi nodes that are equipped with a built-in IEEE 802.11n wireless Network Interface Card (NIC). The NIC can be used as a software access-point for



Node	Threshold		
	0.7	0.8	0.9
1	[2,3,4,5,6,7]	[4]	-
2	[1,4]	-	-
3	[1,4]	-	-
4	[1,2,3,7]	-	-
5	[1,6,7]	[1]	-
6	[1,5]	-	-
7	[1,4,5,8]	-	-
8	[1,7,9]	-	-
9	[1,8]	-	-
10	-	-	-

Fig. 10: (a) Testbed set up in the ECE Department at Rutgers University. The goal of the testbed is to simulate a surveillance system in which images that are sent to the server for further processing are compressed using distributed dictionary learning. The testbed of 10 Raspberry Pi nodes and cameras cover a region in a hallway with three elevators and two doors; (b) Preprocessing using frame differencing and thresholding of incoming images at local nodes is done to identify if there is a significant change in the consecutive images. If the number of non-zero pixels in the difference image is above a pre-determined threshold value then the image is compressed and sent to the cloud. Frame differencing is a low-complexity technique to identify that we are not sending frames with redundant information. (Top) High movement; (Below) Low movement; (c) Table showing correlation coefficient of image data captured by of various nodes in our testbed.

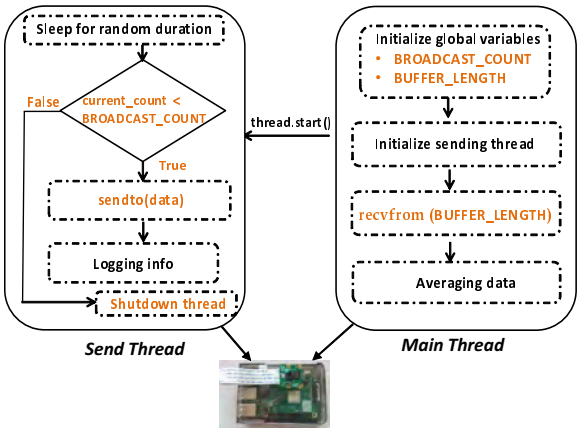


Fig. 11: Communication workflow implemented at the Raspberry Pi nodes our testbed. The communication workflow is used to enable consensus in the network. Two separate thread separate the receiving and transmission of data at each node

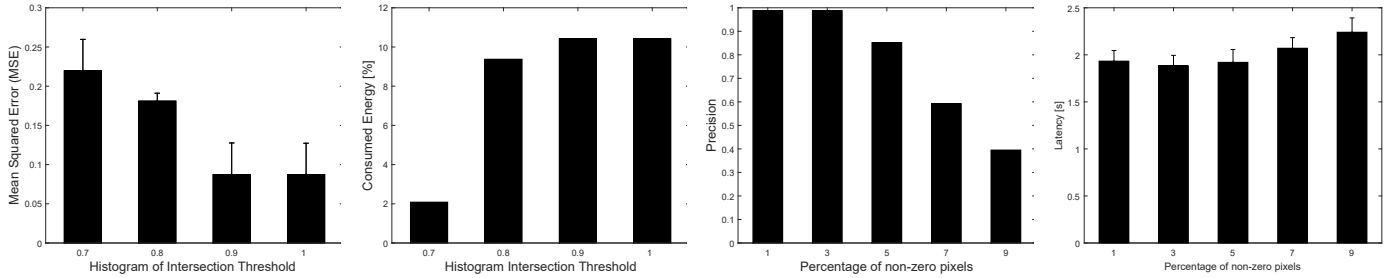
the wireless local-area network. Hence, we follow the standards of IEEE 802.11n [45]–[47] to establish communication between nodes in the testbed. The wireless performance of IEEE 802.11n protocol is well established in the literature and has been shown to achieve a high data-rate of up to 100 Mbits/sec to reliably support wireless videos (e.g., two 20 Mb/s HTDV streams between adjacent rooms). The cameras used in our testbed are Sony IMX219 8-megapixel sensor 1080P, along with Picamera Python library. To measure the power we use a power monitor, PortaPower 3-20V Dual USB Power Monitor that displays current, voltage, and amount of current received by devices. Fig. 10 indicates our setup and the communication workflow for a surveillance application. Before we start estimating the dictionary collaboratively, we use (5) to identify nodes in the network with correlated data. We use the method illustrated in Sect. 3. To this end, each node estimates histogram of the image collected by it and broadcasts it to the nearby nodes.

Online image compression: In the online phase, we use the dictionary we learned using the training images and apply it to perform image compression on the incoming images. We want cameras in the network to send a frame to the server for object detection only when there is a significant change from the last frame that was sent by the camera node. This will help in reducing the energy required to compress and transmit images, which have redundant information. To identify which image to send, we first take the difference of consecutive images at each camera node and use Otsu thresholding [48] to calculate a global threshold. See Fig. 10(b) for visualized results. All pixels in the difference image which are below the threshold are given a zero pixel value. In the gray scale image, if the percentage of pixels that are above the threshold (selected as 0.02 in experiments) are greater than 10%, then we compress those frames using the learned dictionary and send them to the remote server.

At the server we can use image decoding techniques to convert the byte sequence to image data. Since we are using Huffman coding at the local nodes, we used Huffman decoding at the server to reconstruct images. Next, we execute object detection on these reconstructed images. Dictionaries need to be transmitted to server only once by the nodes in the network and can be reused for stream of incoming images. Our solution assumes that, once the dictionary has been learned by the camera nodes, the nodes continue to use the learned dictionary for image compression unless the approximation error between the sparsely coded image and the original image is higher than an acceptable level (based on a predefined threshold). In case of high approximation errors, which can happen when the environment in which camera nodes are situated changes significantly, the dictionary-learning algorithm can be re-triggered. In such situations, our solution will be run again and a new dictionary will be learned.

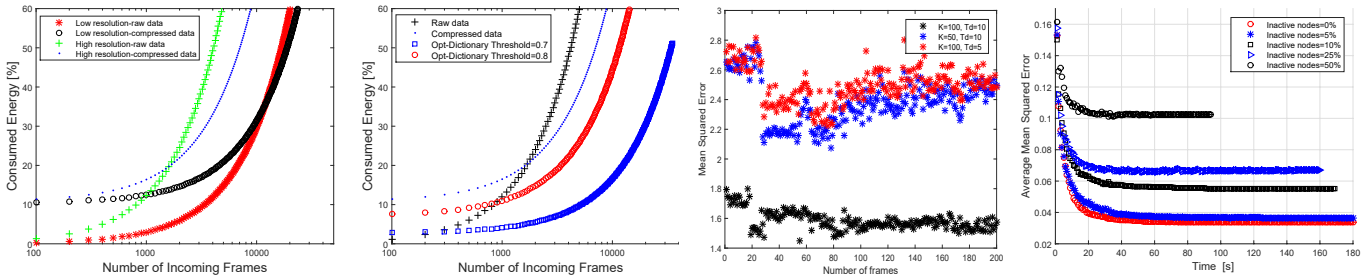
As the number of objects (since we are focusing on person detection in the dataset, the objects here will be number of people in the images) in the image is increased, the preprocessing algorithm in our pipeline which is based on differencing of consecutive images is triggered. And if the number of non-zero pixels in the difference image is above a pre-determined threshold value then the image is compressed and sent to the cloud. Hence, irrespective of the number of people in the image this algorithm will be triggered. Object detection will then be applied at the cloud.

Communication with remote server: We initialize an Amazon EC2 compute instance ("t2.micro" instance) where the object detection algorithm is run on the compressed image. We use the public DNS of the created instance to securely transmit data from each Raspberry Pi to the server. Once the images have been received at the server, the object detection algorithm is executed. To this end, we use single-shot detector [44] convolution neural



(a) (b) (c) (d)

Fig. 12: (a) Mean Squared Error (MSE) performance of our testbed; (b) Energy consumed by the nodes when we vary the threshold for histogram intersection coefficient. Only nodes that have value above this participate in the dictionary learning algorithm; (c) Latency (time taken from the instant image is captured by the camera till the image reaches the server) in our experimental testbed when estimated dictionary is used to compress images; (d) Precision of the object detection algorithm for various percentage of non-zero pixels in the difference image of consecutive images.



(a) (b) (c) (d)

Fig. 13: Comparison of energy costs consumed by the network of Raspberry Pi nodes in the network (a) Raw images versus compressed images are sent to the cloud after compressing using the estimated dictionary. We consider here two image resolutions: high-resolution (1920, 1080) and low-resolution (366, 288); (b) Images are compressed using dictionary learned for a camera network in which nodes that have correlated data with other nodes are silenced; (c) MSE of consensus as the number of nodes participating in the consensus process is varied; (d) MSE as the complexity of the image on which object detection is executed at the server is varied. These images in the network are compressed using learned dictionary before transmitting to the cloud.

network for object detection. Two separate thread separate the receiving and transmission of data at each node as shown in Fig. 11.

Performance in the testbed: We study the performance of our testbed in terms of mean squared performance, energy costs, and latency incurred by processing images in our testbed. We vary the threshold for histogram intersection from 1.0 to 0.7 and see that in Fig. 12(a-b) the cost of dictionary learning falls by 75% and the MSE increases by 35%. We also study the latency as the percentage of non-zero pixels vary in Fig. 12(c-d). The non-zero pixels are achieved after taking a difference of pixel values of consecutive images and in the new image obtained after subtraction. We replace all the pixels that have value below a predefined threshold with zero. Latency is the time taken to compress the image by using the algorithm explained in Fig. 3(b) and to send it to the server. We observe that, as we increase the percentage of non-zero pixels after image differencing (which is done by decreasing the threshold), the latency increases and the precision at the server falls. This is because at high latency fewer images reach the server. For the images that do not reach the server, the object-detection algorithm does not detect the object in those frames, as a result of which the precision decreases.

Energy costs: We now discuss the long term energy costs of running image compression using estimated dictionary. In Fig. 13(a), we see the impact of image resolution on energy cost of compressing image using dictionary estimated via our proposed methodology and sending to the cloud. We study two resolutions: high-resolution Raspberry Pi with 1920×1080 and EPFL images with 366×288 resolution. In Fig. 13(b), we compress images using the dictionary learned via our proposed algorithm at different thresholds of histogram intersection. We see that sending raw data leads to more energy consumption for the nodes in comparison with when compression using dictionary learning is performed. This is because the energy cost of estimating the dictionary via our proposed method is much lower; hence we see savings in the energy consumed at the devices. We also observe that, for high resolution images, the energy consumed at the camera nodes when sending raw data images exceeds the energy consumed by images compressed by dictionary after 200 images have been sent.

Distributed dictionary learning under non-ideal conditions: We consider a scenario where a certain percentage of randomly selected nodes from the network are not allowed to participate in the consensus process. For these simulations we considered there are 20 nodes in the network and we fix the number of times a node can broadcast its data to be 20. We again plot average mean square error which is the difference between average of the data values under perfect consensus ($t_c \rightarrow \infty$) and the data values at the nodes after a fixed number of consensus iterations, averaged over number of nodes. We observe in Figs. 13(c-d), as the number of nodes that participate in the consensus process reduces the mean squared error of the network increases. We also see that the consensus process ends earlier as fewer nodes are broadcasting in comparison to when all the nodes are participating in consensus (0% inactive nodes).

Re-triggering dictionary learning algorithm: Our solution assumes that, once the dictionary has been learned by the camera nodes, the nodes continue to use the learned dictionary for image compression unless the approximation error between the sparsely

coded image and the original image is higher than an acceptable level (based on a pre-defined threshold). In case of high approximation errors—that can happen when the environment in which camera nodes are situated changes significantly—the dictionary-learning algorithm can be re-triggered. In such situations, our solution will be run again and a new dictionary will be learned.

5 Related Work

We compare here the work done in the area of distributed camera networks, dictionary learning, and image compression. We cover different communities in which distributed consensus has been studied and consider works that tackled the problem of energy-efficient consensus.

Mobile cloud computing: Various works exist in the domain of mobile computing that solve the issue of limited computational capability of devices. We now briefly reiterate some of these works here. These works can be divided into two categories. First, where the resource-constrained device offloads its tasks to a remote cloud. This research involves code partitioning to determine which tasks in the application will be executed locally and which in the cloud [49]. COSMOS [50] suggests a system to fill the gap between the demands for computing resources by individual mobile devices and the ones cloud providers offer. Second, when the resource-constrained device offloads tasks to mobile devices in the vicinity, which execute the tasks in parallel and return results back to the device. Authors have implemented a preliminary ad-hoc distributed prototype to offload portions of a service from a resource-constrained device to a nearby servers [51] or nearby devices [52]. A participatory computing system is proposed in [53] to derive the minimal workload for individual participating devices to achieve the overall system performance requirement. Serendipity [54] also exploits the computational resources available in other mobile systems in a collaborative manner.

Similar solutions can be seen in the area of distributed camera networks. These works have focused on conserving energy for applications via, first, offloading or handover [55] of computationally intensive tasks to other nodes and, second, using dynamic power management schemes [56]. Our solution looks at the problem of energy-efficient computation in distributed networks from a different angle. Dictionary learning is an energy-intensive algorithm because of the high communication between the nodes. The computation cost of executing distributed dictionary learning on nodes is much lower than the communication cost. In this paper, we give more detail on the energy cost of both computation and communication tasks in dictionary learning algorithm. To reduce the energy cost in distributed dictionary learning we focus on identifying a subset of nodes on which dictionary learning can be executed. To identify these nodes we rely on the data captured by the nodes as a result of which the lifetime of nodes is extended.

Distributed consensus: Many existing works have considered consensus for ad-hoc networks via gossip algorithms where each node uniformly at random contacts a neighbor within its transmission range and exchanges data [57]. There have been a few recent works that have focused on gossiping via broadcasting [35] and geographic gossiping [58]. However, these works make some unrealistic assumptions that are not consistent with real-world deployment scenarios such as knowledge of topology, unit disk graph models, uniform distribution of nodes, and homogeneous battery capacity of nodes. In our work we do not make such assumptions and rely on the local knowledge available to the nodes to make decisions. Consensus algorithm has been applied to two cooperative control problems including *rendezvous* and *axial alignment* [59]. However, increasing the number of nodes may not be feasible due to the limitations of the defined scenario. Our focus in this paper is on proof-of-concept of an end-to-end and real-time solution for energy-efficient and robust dictionary learning in distributed camera networks. To this end, our main contribution is development and a real testbed-based evaluation of a solution that exploits spatial correlation of the collected multimedia data for energy efficiency. Within the proposed solution, consensus is just a small routine that can be replaced with any other distributed message passing protocol.

Dictionary learning: Dictionary learning can be performed in either centralized [4] or distributed environments [22]. Centralized dictionary learning is not efficient as (i) the nodes have to communicate with a central node (usually at a far distance), which imposes high communications costs, especially in extreme environments; (ii) move the nodes in order to get closer to the central node imposes extra delays, which challenges the real-time processing; and (iii) the structure is likely to break if the centralized node fails to provide the required support. This failure can happen due to energy, communication, or computational restrictions in a network. On the other hand, when there are multiple sensors in the network, e.g., camera network, where images are collected and stored in multiple geographic locations, a distributed solution is energy efficient, avoids a single point of failure, and provides more flexibility and robustness against the topology variations of the network. Our focus in this paper is to present a solution to make distributed dictionary learning energy efficient so that it can be beneficial to run on resource-constrained nodes.

Image compression: Image compression is a popular tool for image data size reduction, so that data transmission becomes feasible in bandwidth-limited and error-prone channels. Compared to the traditional lossy compression methods—such as in Joint Photographic Experts Group (JPEG) and Discrete Wavelet Transform (DWT)—dictionary learning can be used to enhance image-compression algorithms. A sparse model seems to fit natural images as well as human visual perception of images [60]. An improved sparse representation algorithm was proposed in [61], in which dictionaries with overlapping atoms are generated in the wavelet coefficient domain and in the pixel domain so as to remove blocking artifacts. Dictionary learning has been shown to achieve dramatic improvement over JPEG/JPEG2000 compression techniques [9], [11], [12].

To the best of our knowledge, no works exist in the literature that are similar to the framework of this work on distributed image compression. We focused on dictionary learning in this work for image compression because of the ability of a learned dictionary to outperform JPEG in different scenarios and the ability of dictionary learning to obtain an appropriate basis when JPEG basis is not the best one for compression purposes. However, there is value to also developing a JPEG-based distributed image compression system and this would be investigated in the future.

We conclude by noting that the existing works on distributed image compression [62], [63] in a wireless sensor network rely on require partitioning of images to implement JPEG in a distributed network. These work make strong assumptions that each nodes or multiple cluster heads in the network have complete knowledge of the topology of the wireless network and these works are applicable to scenarios where low volume of data has to be compressed. This is different from our work in which each nodes assumes only knowledge of the neighboring nodes and also works

with video-based data (as shown in our experimental section for 25 fps videos). Also, these work assume the wireless network to be a multi-hop wireless network where nodes, where the image data taken at the source node is compressed as it travels from the source to base station. Our work on the other hand focuses on a camera sensor network for surveillance where each camera node captures an image rather than one image captured by the source node in existing works.

6 Conclusion

We presented an energy-efficient technique to reduce the cost of dictionary learning in a distributed camera network to support real-time in-network image compression. We designed a protocol that identifies spatially-correlated nodes using local knowledge of the network available to the camera nodes. To this end, we utilized low-computational-complexity metrics such as intersection of histograms to quantify the correlation of data among camera nodes instead of using pixel-by-pixel cross-correlation function (2D cross-correlation). Based on exhaustive experiments and simulations, we have shown to be able to reduce the cost of dictionary learning by 75%; because of this reduction in energy we are able to send more image data to the cloud server and consequently to extend the mission time of camera nodes. We also presented via simulations performance results in terms of accuracy and energy consumed by the nodes to run consensus and dictionary-learning algorithms. The proposed approach was validated through extensive simulations on public datasets as well as via real-world experiments on a testbed of Raspberry Pi nodes.

Acknowledgment: We thank ECE grad student Brian Qiu for his help with the experiments. This work was partially supported by the NSF NRI Award No. IIS-1734362, the NSF CAREER Award No. CCF-1453073, and the DARPA Lagrange Program under ONR/SPAWAR contract N660011824020.

References

[1] P. Pandey, M. Rahmati, D. Pompili, and W. U. Bajwa, "Robust distributed dictionary learning for in-network image compression," in *Proceedings of International Conference on Autonomic Computing (ICAC)*. IEEE, 2018, pp. 61–70.

[2] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl, "Autonomous multicamera tracking on embedded smart cameras," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 1, p. 092827, 2007.

[3] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, 2008.

[4] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[5] J. Fu, H. Yuan, R. Zhao, and L. Ren, "Clustering k-svd for sparse representation of images," *EURASIP Journal on Advances in Signal Processing*, vol. 2019, no. 1, p. 47, 2019.

[6] X. Zhan, R. Zhang, D. Yin, and C. Huo, "Sar image compression using multiscale dictionary learning and sparse representation," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 5, pp. 1090–1094, 2013.

[7] M. Nejati, S. Samavi, N. Karimi, S. M. R. Soroushmehr, and K. Najarian, "Boosted dictionary learning for image compression," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4900–4915, 2016.

[8] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2691–2698.

[9] I. Horev, O. Bryt, and R. Rubinstein, "Adaptive image compression using sparse dictionaries," in *Proceedings of the 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2012, pp. 592–595.

[10] J.-W. Kang, C.-C. J. Kuo, R. Cohen, and A. Vetro, "Efficient dictionary based video coding with reduced side information," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2011, pp. 109–112.

[11] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–282, 2008.

[12] A. Qayyum, A. S. Malik, M. Naufal, M. Saad, M. Mazher, F. Abdullah, and T. A. R. B. T. Abdullah, "Designing of overcomplete dictionaries based on dct and dwt," in *2015 IEEE Student Symposium in Biomedical Engineering & Sciences (ISSBES)*. IEEE, 2015, pp. 134–139.

[13] I. Tosic and P. Frossard, "Dictionary learning," *IEEE Signal Processing Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[14] Z. Shakeri, A. D. Sarwate, and W. U. Bajwa, "Sample complexity bounds for dictionary learning from vector- and tensor-valued data," in *Information Theoretic Methods in Data Science*, M. R. D. Rodrigues and Y. C. Eldar, Eds. Cambridge, UK: Cambridge University Press, 2020, ch. 5.

[15] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit," Computer Science Department, Technion, Tech. Rep., 2008.

[16] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1001–1016, 2014.

[17] K. Huang, Y. Wu, H. Wen, Y. Liu, C. Yang, and W. Gui, "Distributed dictionary learning for high-dimensional process monitoring," *Control Engineering Practice*, vol. 98, p. 104386, 2020.

[18] H.-W. Chen, L.-W. Kang, and C.-S. Lu, "Dictionary learning-based distributed compressive video sensing," in *28th Picture Coding Symposium*. IEEE, 2010, pp. 210–213.

[19] "NASA's mars science laboratory," 2020, Accessed: 2020-03-16. [Online]. Available: https://mars.nasa.gov/msl

[20] "Woods Hole Oceanographic Institution (WHOI) Autonomous Underwater Vehicles (AUVs)," 2020, Accessed: 2020-04-14. [Online]. Available: https://www.whoi.edu/what-we-do/explore/underwater-vehicles/

[21] "Home Design 3D," 2018, Accessed: 2018-02-16. [Online]. Available: https://en.homedesign3d.net

[22] J. Liang, M. Zhang, X. Zeng, and G. Yu, "Distributed dictionary learning for sparse representation in sensor networks," *IEEE Transactions on Image Processing*, vol. 23, no. 6, pp. 2528–2541, 2014.

[23] H. Raja and W. U. Bajwa, "Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data," *IEEE Transactions on Signal Processing*, vol. 64, no. 1, pp. 173–188, 2016.

[24] P. Chainais and C. Richard, "Learning a common dictionary over a sensor network," in *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2013, pp. 133–136.

[25] "Raspberry Pi Power Battery Power Pack," Accessed: 2018-02-16. [Online]. Available: http://wiki.sunfounder.cc/index.php?title=Raspberry_Pi_Lithium_Battery_Power_Pack

[26] B.-B. Benuwa, B. Ghansah, and E. K. Ansah, "Deep locality-sensitive discriminative dictionary learning for semantic video analysis," *Software: Practice and Experience*, pp. 1–19, 2020.

[27] J. Huo, Y. Gao, W. Yang, and H. Yin, "Multi-instance dictionary learning for detecting abnormal events in surveillance videos," *International journal of neural systems*, vol. 24, no. 03, p. 1430010, 2014.

[28] Z. Shakeri, H. Raja, and W. U. Bajwa, "Dictionary learning based nonlinear classifier training from distributed data," in *Proceedings of 2nd Global Conf. Signal and Information Processing (GlobalSIP'14), Symposium on Network Theory*. IEEE, Dec. 2014, pp. 759–763.

[29] J. Liu, X.-C. Tai, H. Huang, and Z. Huan, "A weighted dictionary learning model for denoising images corrupted by mixed noise," *IEEE transactions on image processing*, vol. 22, no. 3, pp. 1108–1120, 2012.

[30] Z. Xing, M. Zhou, A. Castrodad, G. Sapiro, and L. Carin, "Dictionary learning for noisy and incomplete hyperspectral images," *SIAM Journal on Imaging Sciences*, vol. 5, no. 1, pp. 33–56, 2012.

[31] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin, "Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 130–144, 2011.

[32] Y.-L. Chen, T.-S. Chen, T.-W. Huang, L.-C. Yin, S.-Y. Wang, and T.-c. Chiueh, "Intelligent urban video surveillance system for automatic vehicle detection and tracking in clouds," in *Proceedings of International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2013, pp. 814–821.

[33] B. Song, M. M. Hassan, A. Alamri, A. Alelaiwi, Y. Tian, M. Pathan, and A. Almogren, "A two-stage approach for task and resource management in multimedia cloud environment," *Computing*, vol. 98, no. 1-2, pp. 119–145, 2016.

[34] W. U. Bajwa and A. Pezeshki, "Finite frames for sparse signal processing," in *Finite Frames*, P. Casazza and G. Kutyniok, Eds. Cambridge, MA: Birkhäuser Boston, 2012, ch. 10, pp. 303–335.

[35] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, 2009.
[36] K. Skretting, "Image compression tools for matlab," Accessed: 2018-02-14. [Online]. Available: http://www.ux.uis.no/~karlsk/ICTools/ictools.html
[37] A. Moffat, "Huffman coding," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–35, 2019.
[38] R. Dai and I. F. Akyildiz, "A spatial correlation model for visual information in wireless multimedia sensor networks," *IEEE Transactions on Multimedia*, vol. 11, no. 6, pp. 1148–1159, 2009.
[39] D. Devarajan, Z. Cheng, and R. J. Radke, "Calibrating distributed camera networks," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1625–1639, 2008.
[40] S.-H. Cha and S. N. Srihari, "On measuring the distance between histograms," *Pattern Recognition*, vol. 35, no. 6, pp. 1355–1370, 2002.
[41] S. Xiong and J. Li, "An efficient algorithm for cut vertex detection in wireless sensor networks," in *Proceedings of International Conference on Distributed Computing Systems*. IEEE, 2010, pp. 368–377.
[42] O. Dagdeviren and V. K. Akram, "An energy-efficient distributed cut vertex detection algorithm for wireless sensor networks," *The Computer Journal*, vol. 57, no. 12, pp. 1852–1869, 2014.
[43] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple Object Tracking using K-Shortest Paths Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.
[44] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proceedings of European Conference on Computer Vision*. Springer, 2016, pp. 21–37.
[45] "IEEE 802.11n," 2009. [Online]. Available: https://standards.ieee.org/standard/802_11n-2009.html
[46] E. Perahia, "Ieee 802.11 n development: History, process, and technology," *IEEE Communications Magazine*, vol. 46, no. 7, pp. 48–55, 2008.
[47] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Impact of ieee 802.11 n/ac phy/mac high throughput enhancements on transport and application protocols—a survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2050–2091, 2017.
[48] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
[49] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.
[50] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proceedings of the 15th ACM International Symposium on Mobile Ad-hoc Networking and Computing*. ACM, 2014, pp. 287–296.
[51] A. Messer, I. Greenberg, P. Bernadat, D. Milojicic, D. Chen, T. J. Giuli, and X. Gu, "Towards a distributed platform for resource-constrained devices," in *Proceedings of the International Conference on Distributed Computing Systems*. IEEE, 2002, pp. 43–51.
[52] P. Pandey, H. Viswanathan, and D. Pompili, "Robust orchestration of concurrent application workflows in mobile device clouds," *Journal of Parallel and Distributed Computing*, vol. 120, pp. 101–114, 2018.
[53] Z. Dong, L. Kong, P. Cheng, L. He, Y. Gu, L. Fang, T. Zhu, and C. Liu, "REPC: Reliable and efficient participatory computing for mobile devices," in *Proceedings of the 11th International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2014, pp. 257–265.
[54] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proceedings of the 13th ACM international symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2012, pp. 145–154.
[55] L. Esterle, P. R. Lewis, X. Yao, and B. Rinner, "Socio-economic vision graph generation and handover in distributed smart camera networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 2, p. 20, 2014.
[56] U. A. Khan and B. Rinner, "Online learning of timeout policies for dynamic power management," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 4, p. 96, 2014.
[57] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.
[58] A. D. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient averaging for sensor networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1205–1216, 2008.
[59] W. Ren, H. Chao, W. Bourgeous, N. Sorensen, and Y. Chen, "Experimental validation of consensus algorithms for multivehicle cooperative control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 745–752, 2008.
[60] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1," *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.
[61] K. Skretting and K. Engan, "Image compression using learned dictionaries by RLS-DLA and compared with K-SVD," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 1517–1520.
[62] H. Wu and A. A. Abouzeid, "Energy efficient distributed image compression in resource-constrained multihop wireless networks," *Computer Communications*, vol. 28, no. 14, pp. 1658–1668, 2005.
[63] S. Heng, C. So-In, and T. G. Nguyen, "Distributed image compression architecture over wireless multimedia sensor networks," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

Parul Pandey is currently a Data Scientist at Intel Corporation, OR. She completed her PhD in Electrical and Computer Engineering (ECE) in 2019 from Rutgers University, NJ, USA, under the guidance of Dr. Pompili at the Cyber-Physical Systems Laboratory (CPS Lab). Her research interests span applied machine learning, mobile computing, and distributed computing. Her PhD research focused on accelerating computation-intensive applications running on resource-constrained systems such as mobile phones, drones, and robots. She received the BS degree in electronics and communication engineering from Indira Gandhi Institute of Technology, Delhi, India, and the MS degree in ECE from the University of Utah, Salt Lake City, UT, USA, in 2008 and 2011, respectively. She received best paper awards at the 2017 IEEE Wireless On-demand Network Systems & Services (WONS) and at the 2016 IEEE Transactions on Automation Science and Engineering (T-ASE); Teaching and Graduate Asst. Professional Development Fund Award, Rutgers University (2016, 2017); Travel awards for doctoral consortium at ACM Richard Tapia Celebration of Diversity in Computing; Grace Hopper Celebration Scholar (2014); Rutgers ECE Research Excellence Award (2013).

Mehdi Rahmati is currently a faculty member at Cleveland State University, Ohio. He completed his PhD in Electrical and Computer Engineering (ECE) in 2020 from Rutgers University, NJ, USA, under the guidance of Dr. Pompili at the Cyber-Physical Systems Laboratory (CPS Lab). Previously, he worked as a full-time faculty member at Bahar Institute of Higher Education and as a lecturer at Azad University of Tehran and Mashhad, both in Iran, after receiving his BS and MS in electrical engineering from Ferdowsi University and Iran University of Science and Technology in 2001 and 2009, respectively. Currently, he is working on underwater acoustic communications and human-robot dynamic interaction in uncertain environments. He has received the best demo award at the 2019 IEEE International Conference on Sensing, Communication and Networking (SECON), the best paper award at the 2017 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), and the best paper runner-up award at the 2015 ACM International Conference on Underwater Networks & Systems (WUWNet). In 2019 he was the recipient of the Rutgers Raritan River Consortium (R3C) mini-grant award for research on the Raritan river, basin and bay. He has served as a TPC member in a dozen international conferences, and is currently serving as demo and poster chair for WUWNet'19.

Waheed U. Bajwa has been with Rutgers since 2011, where he is currently an associate professor in the Dept. of ECE and an associate member of the graduate faculty of the Dept. of Statistics and Biostatistics. His research interests include statistical signal processing, high-dimensional statistics, machine learning, harmonic analysis, inverse problems, and networked systems. He has received a number of awards including the Army Research Office Young Investigator Award (2014), the National Science Foundation CAREER Award (2015), Rutgers Presidential Merit Award (2016), Rutgers Presidential Fellowship for Teaching Excellence (2017), and Rutgers Engineering Governing Council ECE Professor of the Year Award (2016, 2017, 2019). He is a co-investigator on a work that received the Cancer Institute of New Jersey's Gallo Award for Scientific Excellence in 2017, a co-author on papers that received Best Student Paper Awards at IEEE IVMSP 2016 and IEEE CAMSAP 2017 workshops, and a Member of the Class of 2015 National Academy of Engineering Frontiers of Engineering Education Symposium. Dr. Bajwa is currently serving as the Lead Guest Editor for a special issue of IEEE Signal Processing Magazine on "Distributed, Streaming Machine Learning," a Guest Editor for a special issue of Proceedings of the IEEE on "Optimization for Data-driven Learning and Control," a Senior Area Editor for IEEE Signal Processing Letters, an Associate Editor for IEEE Transactions on Signal and Information Processing over Networks, and an elected member of the Big Data Special Interest Group and Sensor Array and Multichannel (SAM) and Signal Processing for Communications and Networking (SPCOM) Technical Committees of the IEEE Signal Processing Society.

Dario Pompili is an associate professor with the Dept. of ECE at Rutgers University. Since joining Rutgers in 2007, he has been the director of the CPS Lab, which focuses on mobile edge computing, wireless communications and networking, acoustic communications, and sensor networks. He received his PhD in ECE from the Georgia Institute of Technology in 2007. He had previously received his 'Laurea' (combined BS and MS) and Doctorate degrees in Telecommunications and System Engineering from the U. of Rome "La Sapienza," Italy, in 2001 and 2004, respectively. He has received a number of awards in his career including the NSF CAREER'11, ONR Young Investigator Program'12, and DARPA Young Faculty'12 awards. In 2015, he was nominated Rutgers-New Brunswick Chancellor's Scholar. He served on many international conference committees taking on various leading roles. He published about 200 refereed scholar publications, some of which selected to receive best paper awards: with more than 12K citations, Dr. Pompili has an h-index of 43 and an i10-index of 107 (Google Scholar, Feb'21). He is a Fellow of the IEEE Communications Society (2021) and a Distinguished Member of the ACM (2019). He is currently serving as Associate Editor for IEEE Transactions on Mobile Computing (TMC).