Distributed Multi-agent Video Fast-forwarding

Shuyue Lan
Northwestern University
shuyuelan2018@u.northwestern.edu

Zhilu Wang
Northwestern University
zhiluwang2018@u.northwestern.edu

Amit K. Roy-Chowdhury
University of California, Riverside
amitrc@ece.ucr.edu

Ermin Wei
Northwestern University
ermin.wei@northwestern.edu

Qi Zhu
Northwestern University
qzhu@northwestern.edu

ABSTRACT

In many intelligent systems, a network of agents collaboratively perceives the environment for better and more efficient situation awareness. As these agents often have limited resources, it could be greatly beneficial to identify the content overlapping among camera views from different agents and leverage it for reducing the processing, transmission and storage of redundant/unimportant video frames. This paper presents a consensus-based distributed multi-agent video fast-forwarding framework, named **DMVF**, that fast-forwards multi-view video streams collaboratively and adaptively. In our framework, each camera view is addressed by a reinforcement learning based fast-forwarding agent, which periodically chooses from multiple strategies to selectively process video frames and transmits the selected frames at adjustable paces. During every adaptation period, each agent communicates with a number of neighboring agents, evaluates the importance of the selected frames from itself and those from its neighbors, refines such evaluation together with other agents via a system-wide consensus algorithm, and uses such evaluation to decide their strategy for the next period. Compared with approaches in the literature on a real-world surveillance video dataset VideoWeb, our method significantly improves the coverage of important frames and also reduces the number of frames processed in the system.

CCS CONCEPTS

• **Computing methodologies** → *Computer vision*; • **Computer systems organization** → *Embedded and cyber-physical systems.*

KEYWORDS

Video fast-forwarding, multi-agent, distributed optimization

ACM Reference Format:
Shuyue Lan, Zhilu Wang, Amit K. Roy-Chowdhury, Ermin Wei, and Qi Zhu. 2020. Distributed Multi-agent Video Fast-forwarding. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394171.3413767

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413767

1 INTRODUCTION

In many intelligent multi-agent systems, a network of agents with cameras can jointly perform tasks such as search and rescue, surveillance, and environment monitoring. These cameras may be fixed surveillance cameras, or built-in cameras on robots/drones. They generate a large amount of videos from different viewpoints, and sometimes transmit these videos to a cloud server for further analysis, decision making, and storage. For many applications, the processing and transmission of the video frames from the agents to the server needs to be performed in an online manner at real time or near real time. However, the agents often have limited computation, communication, storage, and energy resources [1, 27, 49], which make it challenging or even intractable to process and transmit all the video data at real time. Thus, methods that can select an *informative subset* of the video frames for processing, transmission and storage are greatly needed to reduce the resource requirements.

In the literature, video summarization techniques, which generate a compact summary of the original video, have been widely studied [6, 14, 35, 56–58]. Multi-view video summarization that handles video streams from multiple cameras has also been addressed in several works [5, 10, 34, 36, 38]. However, these methods need to process an entire video (i.e., every frame in the video) and often take a long time for generating a summary, which are not suitable for real-time and online applications. Some methods [3, 15, 20, 41, 42, 44, 48] have been developed for video fast-forwarding by adjusting the playback speed of a video, but they still require processing of the entire video and will not reduce the amount of data to be transmitted when used in a multi-agent scenario.

A recent work [24] performs fast-forwarding for a single agent in an online manner, and processes a fraction of the video frames by automatically skipping unimportant frames via reinforcement learning. However, the method considers only a single agent and cannot be easily extended to the multi-agent domain. Moreover, as we observe that there are often significant overlaps among the videos captured by the different agents, we start by asking the following question: *Is it possible to develop a method for multiple agents to collaboratively perform fast-forwarding that is efficient, causal, online and results in an informative summary for the scene?*

In this paper, we introduce the **D**istributed **M**ulti-agent **V**ideo **F**ast-Forwarding framework (**DMVF**), a consensus-based framework that collaboratively fast-forwards videos at different views for efficient processing, transmission and storage of video data (see Fig. 1). In our target scenario, cameras at multiple locations observe the same environment from different views that may be overlapping. Each camera embeds a reinforcement learning based

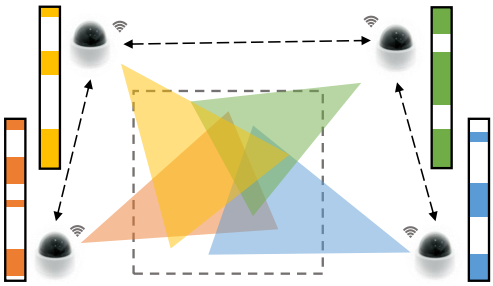


Figure 1: Illustration of distributed multi-agent video fast-forwarding setup. Multiple cameras at different locations observe the same environment from different points of view. Each view has a fast-forwarding agent that periodically adapts its strategy for selectively processing input video frames. At every adaptation period, agents communicate with each other to collaboratively decide each agent's fast-forwarding strategy for the next period.

fast-forwarding agent with multiple strategies to choose from, i.e., it can skip the frames of its video input at different paces (e.g., slow, normal or fast). Agents are connected by a predetermined undirected communication network, where each agent can communicate with a set of neighboring agents. We also assume the communication graph is connected.

During operation, each agent adapts its fast-forwarding strategy periodically based on how important its selected frames are when compared with other agents' frames. At every adaptation period, each agent evaluates the importance of the selected frames from itself and those from its neighbors by comparing their similarities. Intuitively, if an agent's frames can better represent/cover the views of other agents, they are regarded as more important. The agents then refine their evaluation by running a system-wide consensus algorithm [52] and reach an agreement on the importance score for every agent's selected frames. Based on the score ranking and the system requirement, each agent selects a fast-forward strategy for its next adaptation period. Intuitively, agents with lower scores on their selected frames could be given a faster pace for the next period to reduce their processing and transmission load, while the ones with higher scores should be given the same (or slower) pace.

It is worth noting that in our approach, each agent only processes a very small portion of the frames, which greatly helps reduce the computation load on resource-limited embedded platforms. Agents also do not transmit their entire video streams but only a fraction of them. From the system perspective, both intra-view redundancy at each agent and the inter-view redundancy across different agents are reduced. Furthermore, the online and causal nature of our proposed DMVF framework enables the users to begin fast-forwarding at any point when executing certain multi-agent perception tasks. Our approach is particularly useful for resource-constrained and time-critical systems such as multi-robot systems.

To summarize, the following are the main contributions.

- We formulate the multi-agent video fast-forwarding problem as a multi-agent reinforcement learning problem. Each agent can fast-forward its video input without processing the entire video.
- We design a distributed and consensus-based framework that enables adaptive fast-forwarding strategies/paces for all agents and optimizes the computation and communication load globally.
- We demonstrate the effectiveness of DMVF on a challenging multi-view dataset, VideoWeb [4], achieving real-time speed on a practical embedded platform. Compared with other methods in the literature, our approach achieves significantly better coverage of the important frames/events and reduces the computation and storage load, as well as the communication to the cloud.

2 RELATED WORKS

2.1 Video Fast-forwarding

Video fast-forwarding is applied when users are not interested in parts of the video. Some commercial video players offer manual control on the playback speed, e.g., Apple QuickTime Player with 2, 5 and 10 multi-speed fast-forward. Researchers adapt the playback speed based on the motion activity patterns present in a video [3, 39, 40] and the similarity of each candidate clip to the query clip [41]. Other works focus on developing fast-forwarding policies using mutual information between frames [17, 18] and shortest path distance over the semantic graph built from frames [44, 48]. Another family of work (hyperlapse) [15, 20, 42] fast-forwards videos with the objective of speed-up and smoothing. More recently, Lan et al. [24] propose an online deep reinforcement learning agent for skipping frames and fast-forwarding. Different from these methods, our work focuses on multi-agent video fast-forwarding that collaboratively and distributively fast-forwards videos in each view based on the information from its own perception and neighbors'.

2.2 Video Summarization

The objective of video summarization is to generate a compact subset of videos that can describe the main content of the original video. Many offline methods, which require the entire video being available before processing, are developed with unsupervised learning [7, 8, 12, 13, 28] or supervised learning based on video-summary labels [11, 14, 35, 43, 55–57]. There also work on summarization for crawled web images/videos [22, 23, 37, 50] and photo albums [47]. Learning video summarization from unpaired data is studied in [45]. The other branch of work is online video summarization, which summarizes videos by automatically scanning in an online fashion. Various methods are proposed with submodular optimization [6], Gaussian mixture model [33], and online dictionary learning [58].

What is more related to our work is video summarization from multi-view videos. In [10], the authors introduce the problem of multi-view video summarization and solve it with random walk over spatio-temporal graphs. Joint embedding and sparse optimization are proposed in [36, 38]. More recently, Elfeki et. al [5] adapt DPP (Determinantal Point Processes) to multi-view for offline multi-view video summarization. All these methods require the availability of all frames from each view. In [34], the authors propose a two-stage system, i.e., online single-view summarization and distributed view selection for the multi-view case. Different from these previous methods, our approach does not process all the frames, which significantly reduces computation and communication load, and it collaboratively fast-forwards the videos of multiple agents to further improve the efficiency and coverage.

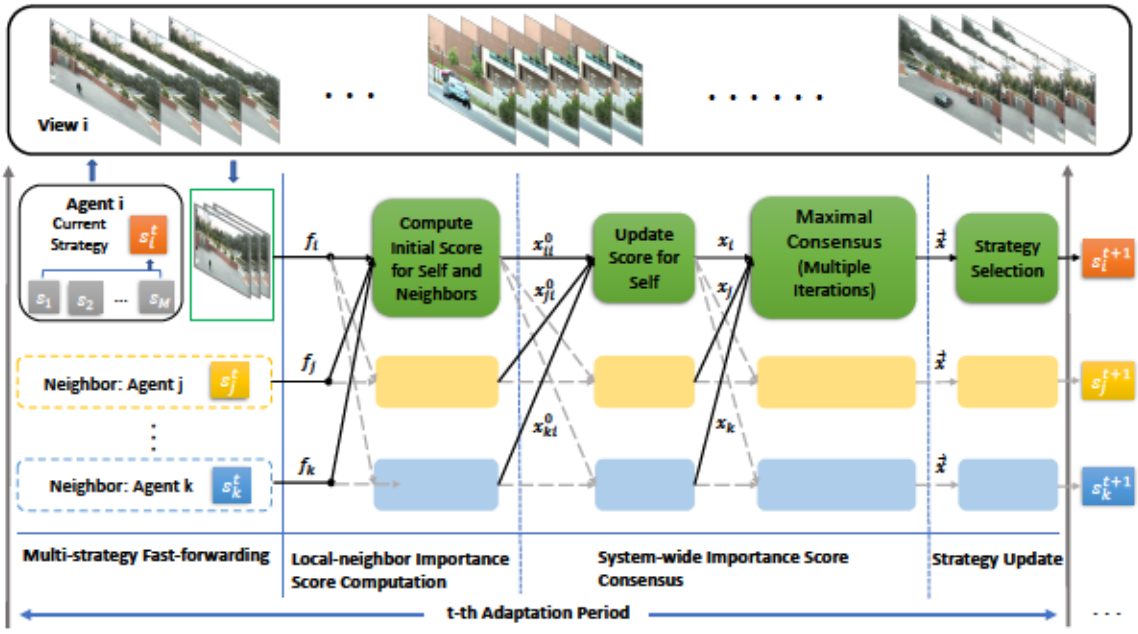


Figure 2: Our DMVF framework. At every adaptation period t, each agent i first fast-forwards its video input with current strategy s_i^t and selects a set of frames f_i (see footnote 1). It then receives neighbor agents' selected frames (e.g., f_j and f_k) and computes an initial importance score for itself and its neighbors. Afterwards, agent i refines and finalizes the importance score with other agents via a system-wide consensus algorithm (maximal consensus is shown in the figure). Based on this importance score vector $\vec{x}$, agent i chooses its strategy for the next period s_i^{t+1} (so does every other agent).

2.3 Distributed Consensus

A fundamental problem in distributed multi-agent systems is the minimization of a sum of local objective functions while maintaining agreement over the decision variable, often referred to as consensus optimization. Seminal work in [52] proposes a distributed consensus protocol for achieving agreement in a multi-agent setting by iteratively taking weighted average with local neighbors. The work in [31] presents a distributed gradient descent (DGD) method, where each agent iteratively updates its local estimate of the decision variable by executing a local gradient descent step and a consensus step. Follow-up works [16, 19, 26, 29, 30, 32] extend this method to other settings, include stochastic network, constrained problems, and noisy environment. More recently, EXTRA [46], which takes a careful combination of gradient and consensus steps, is proposed to improve convergence speed and is shown to achieve linear convergence with constant stepsize. In computer vision, consensus based methods are used applications such as human post estimation [25], background subtraction [54], bundle adjustment [9] and multi-target tracking [21], etc. To the best of our knowledge, this is the first distributed consensus based work to address multi-agent video fast-forwarding.

3 METHODOLOGY

3.1 Problem and Solution Overview

Our objective is to collaboratively fast-forward multi-view videos from different agents by adapting the skipping strategy of each agent in an efficient, online and distributed manner. Fig. 2 shows the workflow design of our framework (taking one agent i for illustration). Given the incoming multi-view video streams $V = \{v_1, \cdots, v_N\}$ captured at different agents, our goal is to generate a final summary $F = \{f_1, \cdots, f_N\}$ for the scene while reducing the computation, communication, and storage load.

In our framework, the fast-forwarding agent of each view is modeled as a reinforcement learning agent with multiple available strategies $S = \{s_m, m = 1, \cdots, M\}$. During operation, at every adaptation period t (with the period length as T), each agent i fast-forwards its own video stream with a current strategy $s_i^t \in S$ and selects a subset of frames f_i [1]. Note that the frames being skipped are not processed nor transmitted. The details of this step are introduced in Section 3.2. Agent i then communicates with its neighbors and receives their selected frames, e.g., f_j and f_k as shown in the figure. Based on such information, agent i computes an initial importance score for itself and its neighbors (Section 3.3). Afterward, agent i refines its initial score together with other agents in the system via a system-wide consensus algorithm, including first an update of its own score and then multiple iterations to reach system-wide consensus (Section 3.4). Note that during the consensus process, only scores are transmitted among events (not selected frames). After running the consensus algorithm, each agent will have the same copy of the final importance scores for their selected

[1] Strictly speaking, f_i should have a superscript t to represent current period (i.e., f_i^t). We omit it for better readability wherever it does not cause confusion. The same goes for many other symbols in this section, such as x_{ii}^0, x_i, $\vec{x}$.

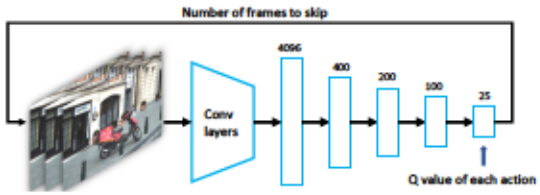


Figure 3: The model structure of normal-pace strategy. It takes a frame in a video stream as an input for the deep neural network and outputs the number of frames to skip.

frames in the current period, defined as $\vec{x} = [x_1, x_2, ..., x_N]$. Agent i then chooses its fast-forwarding strategy for the next period s_i^{t+1} based on the rank of its importance score x_i (Section 3.5). More details are presented below and the notations are in Table 1.

M	number of available fast-forwarding strategies
N	number of camera views / agents
V	the set of N views $\{v_i\}$, $i \in [1, N]$
S	the set of available strategies $\{s^m\}$, $m \in [1, M]$
s_i^t	strategy being used in agent i at adaptation step t
s_i^{t+1}	strategy for agent i in the next adaptation step $t+1$
F	summary of the scene: $\{f_1, \cdots, f_N\}$
$\vec{x}$	importance score vector after consensus
T	period of strategy update

Table 1: Notation used in our proposed framework.

3.2 Multi-strategy Fast-forwarding

On each camera that captures a view of the scene, we have a multi-strategy fast-forwarding agent that can adaptively fast-forward the incoming videos with different paces. In this work, we leverage a state-of-the-art fast-forwarding algorithm, FFNet [24], and derive three different strategies/paces for fast-forwarding: normal-pace, slow-pace, and fast-pace. Note that our approach can be easily extended to consider other numbers of strategies/paces.

Normal-pace Strategy: The normal-pace strategy utilizes the same structure as FFNet [24] (Fig. 3). The fast-forwarding problem is modeled as a Markov decision process (MDP) and solved with a deep Q-learning (DQL) model. It learns a policy for skipping unimportant frames entirely and presenting the important ones for further processing. The state is defined as the feature of the current frame. The action set includes the possible numbers of frames to skip. An immediate reward at time step k is defined as

$$r_k(normal) = -SP_k + HR_k, \tag{1}$$

where SP_k is the "skip" penalty and HR_k is the "hit" reward for current action a_k. The skip penalty gives high scores for skipping unimportant frames and low scores for skipping important ones. The hit reward encourages the agent jumping to an important frame or a position near an important frame. With the definition of states, actions and rewards, a skipping policy is learned for selecting the action that maximizes the expected accumulated reward. As our normal-pace strategy, we use an action space of size 25, i.e., skipping from 1 to 25 frames.

Slow-pace Strategy: The slow-pace strategy aims at skipping fewer frames and thus retaining more frames in the selected buffer, possibly including more numbers of important frames. To meet this goal, we modify the immediate reward in FFNet at time step k as

$$r_k(slow) = (-SP_k + HR_k) \times (1 - \frac{sigmoid(a_k)}{2}). \tag{2}$$

Intuitively, if the agent skips with a larger step, it will receive smaller immediate reward. We also change the action space to 15 to prevent the agent from skipping too much.

Fast-pace Strategy: The goal of the fast-pace strategy is to skip more unimportant frames for more efficient processing and transmission. We modify the immediate reward at time step k as

$$r_k(fast) = (-SP_k + HR_k) \times (1 + \frac{sigmoid(a_k)}{2}). \tag{3}$$

This reward definition ensures that the agent will get larger immediate reward if it skips with a larger step. The action space is set to 35 to allow the agents to skip larger steps.

In our framework, each agent updates its fast-forwarding strategy periodically based on the evaluation of the relative importance of its selected frames in the current period (when compared with other agents). Lower importance may lead to a faster strategy for the next period for reducing data processing and transmission, while higher importance may lead to a slower strategy for reducing the likelihood of missing important frames.

3.3 Local-neighbor Importance Score

In this step, for every agent i, we compute an initial importance score for itself and its neighbors by comparing the similarities between their selected frames. First, we evaluate the similarity between two frames x and y by computing the exponential of the scaled negative L2-norm of the feature representations of the two frames, as defined in the following equation.

$$sim(x, y) = e^{-\alpha||x-y||_2}, \tag{4}$$

where α is used to scale the L2-norm to restrict the similarity value to a satisfactory range. In the experiment, we set $\alpha = 0.05$.

The similarity of agent j to i is then defined as

$$sim\_agent(v_i, v_j) = \frac{1}{|v_j|} \sum_{s=1}^{|v_j|} \max_{1 \le a \le |v_i|} sim(p_s(v_j), p_a(v_i)), \tag{5}$$

where $|v_j|$ denotes the number of selected frames from agent j and $p_s(v_j)$ denotes a selected frame s from agent j. The similarity for frame $p_s(v_j)$ to agent i is the maximum among the similarities between $p_s(v_j)$ and frames of agent v_i. Then the agent-to-agent similarity of agent j to agent i is the average frame similarity.

We define the communication connections among agents as an undirected graph $G = (V, E)$. With this definition, we compute the importance score of agent j estimated by agent i as

$$x_{ij}^0 = \begin{cases} \frac{1}{|V_i|-1} \sum_{v_k \in V_i, k \neq j} sim\_agent(v_j, v_k) & \text{if } i = j \text{ or } (i,j) \in E \\ 0 & \text{o.w.} \end{cases} \tag{6}$$

where $V_i = \{v_k | (i,k) \in E\} \bigcup \{v_i\}$, is the set of the neighbors of agent i and itself. $|V_i|$ represents the number of agents in V_i. This initial important score will then be refined via a consensus process.

Algorithm 1 Algorithm for Computing Importance Score

Input: Selected frames from agents at current period: $f_1, f_2, ..., f_N$
Output: List of important scores $\vec{x}_i$ for each agent/view $v_i \in V$

for each agent $v_i \in V$ **do**
  Send selected frames f_i to neighbors.
  $V_i = \{v_i$'s neighboring agents and itself$\}$.
  for all $v_j \in V_i$ **do**
    $f_j \leftarrow Receive\_Selected\_Frames\_From(v_j)$.
  for all $v_j \in V_i$ **do**
    $x^0_{ij} = \frac{1}{|V_i|-1} \sum_{v_k \in V_i, k \neq j} sim\_agent(v_j, v_k)$.
    Send x^0_{ij} to v_j.
  for all $v_j \in V_i$ **do**
    $x^0_{ji} \leftarrow Receive\_Initial\_Score\_From(v_j)$.
  $x_i = \frac{\sum_{j \in V_i} \frac{1}{n_j} x^0_{ji}}{\sum_{j \in V_i} \frac{1}{n_j}}$.
  $\vec{x}_i = [0, 0, \ldots, 0], \vec{x}_i[i] = x_i$.
  for $t = 1 : graphDiameter$ **do**
    Send $\vec{x}_i$ to neighbors.
    for all $v_j \in V_i$ **do**
      $\vec{x}_j \leftarrow Receive\_Score\_Vector\_From(v_j)$.
      for $k = 1 : N$ **do**
        $\vec{x}_i[k] = \max(\vec{x}_i[k], \vec{x}_j[k])$
  return $\vec{x}_i$

3.4 System-wide Importance Score Consensus

To refine the initial importance score and reach an agreement across all agents on the relative importance of their frames, we mainly use a maximal consensus algorithm in our framework. We have also explored multiple variants of our framework with different consensus methods.

Maximal Consensus Algorithm in DMVF: There are three steps in this algorithm. First, each agent communicates with its neighbors and sends its initial importance scores for each of them. At the end of this step, agent i will have the initial scores of itself from its own computation and from the evaluation by its neighbors (i.e. $\{x^0_{ji}\}$, $j \in V_i$). Then, in the second step, agent i updates its score as

$$x_i = \frac{\sum_{j \in V_i} \frac{1}{n_j} x^0_{ji}}{\sum_{j \in V_i} \frac{1}{n_j}}, \tag{7}$$

which means the importance score of agent i is updated as the weighted average of the initial importance scores evaluated by itself and its neighbors. Then an importance score vector $\vec{x}_i$ for all agents is constructed by agent i, with only the i-th element set to x_i and all others set to zero. In the third step, all agents will run a maximal consensus algorithm over the importance score vector. This algorithm only requires the number of consensus steps to be the diameter of the graph G to reach an agreement (the convergence is guaranteed). In the end, every agent will have the same copy of the importance score vector for all agents, i.e., $\vec{x}_i = \vec{x} = [x_1, x_2, ..., x_N]$. Details of the importance score consensus is shown in Algorithm 1.

Other Consensus Methods: Aside from the main DMVF with maximal consensus algorithm, we also developed multiple variants of our framework with different consensus methods, including DMVF-DGD, DMVF-EXTRA, DMVF-AVE, and DMVF-ONE.

- **DMVF-DGD.** In this design, we leverage the distributed gradient descent (DGD) [31] method for reaching consensus on the importance score of every agent. Each consensus step can be represented by multiplication by an $N \times N$ row stochastic consensus matrix P, with $P_{ij} \neq 0$ if and only if (i, j) in E. In the experiment, the consensus matrix P is defined as

$$\begin{aligned} P_{ij} &= \frac{1}{d_{max}+1} && i \neq j, (i,j) \in E, \\ P_{ij} &= 0 && i \neq j, (i,j) \notin E, \\ P_{ii} &= \frac{d_{max}+1-d_i}{d_{max}+1} && i = j, \end{aligned} \tag{8}$$

  where $d_i = n_i$ is the degree of agent i (i.e., number of neighbors of agent i), $d_{max} = \max_i\{d_i\}$ is the maximum degree in the system. The objective of this distributed optimization is

$$\sum_i f_i(\vec{x}) = \sum_i \left(\frac{1}{n_i} \sum_{(i,j) \in E} (x_j - x^0_{ij})^2 \right) \tag{9}$$

  where $\vec{x}$ is the importance scores for all agents and x_j is the j-th element corresponding to the importance score of agent j. x^0_{ij} is agent j's score evaluated by agent i (computed by Equation (6)). n_i is the total number of neighbors of agent i. The DGD iteration step is as follows:

$$\vec{x}^{t+1}_i = \sum_j P_{ij} \vec{x}^t_j - \gamma^t \nabla f_i(\vec{x}^t_i) \tag{10}$$

  where P_{ij} is the (i, j) element of the consensus matrix P and the stepsize γ^t is gradually reduced based on $\gamma^t \sim \frac{1}{t}$.
- **DMVF-EXTRA:** In this variant, we utilize the decentralized exact first-order algorithm (EXTRA) [46] as the consensus method in the framework. EXTRA has the same consensus matrix and objective as DGD in Equation (8) and Equation (9), respectively. The EXTRA iteration step is as follows:

$$\vec{x}^{t+2}_i = \sum_j M_{ij} \vec{x}^{t+1}_j - \sum_j \frac{M_{ij}}{2} \vec{x}^t_j - \alpha[\nabla f_i(\vec{x}^{t+1}_i) - \nabla f_i(\vec{x}^t_i)] \tag{11}$$

  where $M = I + P$ and M_{ij} is the (i, j) element of M. α is a constant for the iteration stepsize.
- **DMVF-AVE:** First, each agent sends its initial importance score evaluations of its neighbors to them. Then, every agent i updates its score by taking the average of the initial score evaluations of itself from its own computation and from its neighbors:

$$x_i = \frac{\sum_{j \in V_i} x^0_{ji}}{n_i + 1}. \tag{12}$$

  Then, similar to the consensus algorithm in DMVF, all agents will run a maximal consensus algorithm on the importance scores.
- **DMVF-ONE:** In this design, each agent takes x^0_{ii} as its updated score, i.e., $x_i = x^0_{ii}$. Then, all agents will run a maximal consensus algorithm on the importance score.

3.5 Strategy Selection

Based on the final importance scores in $\vec{x}$, the agents with higher scores could be assigned with a slower strategy for the next period, while the agents with lower scores could be faster. In our framework, given the system requirement, the portion of different strategies are pre-defined, which means there should be a fixed number of agents under each strategy after every update. In our experiment, there are three strategies for 6 agents. The system requirement is denoted as $X/Y/Z, X+Y+Z=6$, which means the system requires X agents to use the fast strategy, Y agents to use the normal strategy and Z agents to use the slow strategy. The strategy for each agent will then be assigned based on the ranking of its importance score in $\vec{x}$.

4 EXPERIMENTAL RESULTS

In this section, we present the experimental results of our DMVF framework and its comparison with several methods in the literature. We also demonstrate the trade-off between coverage and efficiency in DMVF, evaluate how various degrees of communication network connectivity affect performance of DMVF, present analysis on different consensus methods, and report timing efficiency. The source code can be found at https://github.com/shuyueL/DMVF.

4.1 Experimental Setup

Dataset: We evaluate the performance of our framework on a publicly available multi-view video dataset VideoWeb [4]. It consists of realistic scenarios in a multi-camera network environment that involves multiple persons performing dozens of different repetitive and non-repetitive activities. We use the Day 4 subset of the dataset, as it involves multiple vehicles and persons. It has 6 scenes and each scene has 6 views of videos. All videos are captured at 640×480 resolution and approximately 30 frames/second. 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 also looked into other multi-view datasets, such as the well-known Office, Campus, Lobby, Road, Badminton datasets from [10], and BL-7F from [34]. The first class of datasets are all of small sizes, with only 3-4 videos (1 video per view) available for each scene. The same problem occurs in BL-7F dataset as it does not have enough overlapping videos for training of the fast-forwarding agent.

Evaluation Metrics: Similar to [24], we consider a *coverage* metric at frame level, which evaluates how well the results from the fast-forwarding methods cover the important frames labeled in the ground truth. The coverage is computed as the percentage of covered frames by all agents in the global ground truth. If an important frame is covered by any of the agents, it will be considered as true positive. We also evaluate the efficiency of various methods by considering their *processing rate*, i.e., the average percentage of frames processed at the agents. The processing rate measures the computation load of the agents. In this work, the communication load can also be indicated by the processing rate, since the number of transmitted frames is proportional to the number of processed frames (with the addition of processed frames from the neighbors).

Implementation Details: The multi-strategy fast-forward agents are implemented using the TensorFlow library and modeled as 4-layer neural networks. ϵ-greedy strategy is used to better explore the state space during the training process. The strategy update period T is set to 100 frames of the raw video inputs. The three strategies used in our framework are defined in Section 3.2. The operating points of agents with the slow, normal and fast strategies are shown in Table 2.

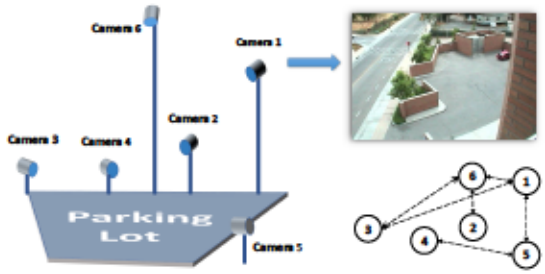


Figure 4: The communication graph of agents. The left part of the figure shows the physical locations of the cameras and the environment. The right-bottom graph is the communication graph of the agents for the main set of experiments (built according to the similarity of views).

Strategy	Slow	Normal	Fast
Processing rate(%)	8.69	6.02	3.73
Coverage(%)	73.45	61.91	55.89

Table 2: Operating points of 3 fast-forwarding strategies.

In our main set of experiments (except for the study in Section 4.4), we evaluate the proposed framework on the aforementioned dataset with a communication graph of agents based on their view similarity, as shown in Fig. 4. We deploy the proposed DMVF on an actual embedded platform. Five agents are implemented on 2 workstations and 3 laptops, and the other one is run on an Nvidia Jetson TX2. The communication between agents is implemented with WiFi network using TCP protocol.

Comparison Methods: We compare our approach with several methods for video fast-forwarding and video summarization, which include both online and offline methods. (1) **FFNet** [24]. FFNet is an online single-agent fast-forwarding approach that utilizes Q-learning. Applying FFNet to multi-agent scenarios assumes that all agents perform video fast-forwarding independently without any communication between them. (2) **Random**. The random method skips the incoming frames randomly. (3) **Uniform**. The uniform method skips the incoming frames periodically. (4) **OK** (Online Kmeans) [2]. Online Kmeans is a classical clustering based method working in an online update fashion. The frames that are the closest to the centroid in each cluster are selected as the summary. (5) **SC** (Spectral Clustering) [53]. Spectral Clustering is a different clustering based method to group all the frames in a video to several clusters. The summary is composed by the frames that are closest to each centroid. (6) **SMRS** (Sparse Modeling Representative Selection) [8]. This is an offline method that requires all videos available before the processing and outputs the summary of each video. It takes the entire video as the dictionary and finds the representative frames based on the zero patterns of the sparse coding vector.

Methods	Random	Uniform	OK	SC	SMRS	FFNet	DMVF
Coverage(%)	50.78	25.80	50.21	44.74	42.36	61.91	65.87
Processing rate(%)	4.20	3.70	100	100	100	6.02	5.06

Table 3: Comparison on coverage and processing rate between DMVF and other approaches. Compared with FFNet, DMVF achieves better coverage (6.40% improvement) while reducing the processing rate by 15.95%. For other methods, DMVF achieves either much better coverage or much lower processing rate or both.

Figure 5: Representative frames generated by DMVF from the VideoWeb dataset.

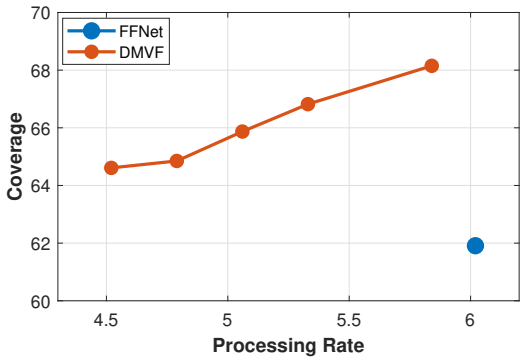


Figure 6: Trade-off between coverage and processing rate in DMVF, under different system requirements.

Experimental Settings: We use the penultimate layer (pool 5) of the GoogLeNet model [51] (1024-dimensions) to represent each video frame. For each method, we tune its parameters for best performance, and control them to keep the fast-forwarded videos to the same percentage of raw frames for a fair comparison. During evaluation, we extend the frames selected by each method before and after by a small window, whose size is the same for all methods. For FFNet, we keep all the settings as described in their paper and train it on the VideoWeb dataset. For OK and SC, we set the number of clusters to 20. We randomly use 80% of the videos for training and the remaining 20% for testing. 5 rounds of experiments are run and the reported result is the average performance.

4.2 Comparison with Other Approaches

Table 3 shows the coverage metric and the processing rate of DMVF on the VideoWeb dataset and its comparison with other approaches. In this experiment, the system requirement is 3/2/1, i.e., 3 fast strategies, 2 normal strategies and 1 slow strategy for the agents (further study on different system requirements is shown later). From the table, we can clearly see the improvement from DMVF:

- Compared with the state-of-the art method for fast-forwarding, FFNet, our approach DMVF achieves better coverage (6.40% improvement) while reduces the processing rate by 15.95%.
- For those methods that require processing the entire video (processing rate of 100%), i.e., OK, SC and SMRS, our framework DMVF achieves higher coverage (more than 25% increase) and much lower processing rate.
- Compared with Random and Uniform methods, DMVF offers significant improvement in coverage with modest increase of processing rate.

Fig. 5 shows a qualitative example for fast-forwarding videos with DMVF. As we can see, it selects more important frames from multiple views, and creates a compact subset of frames.

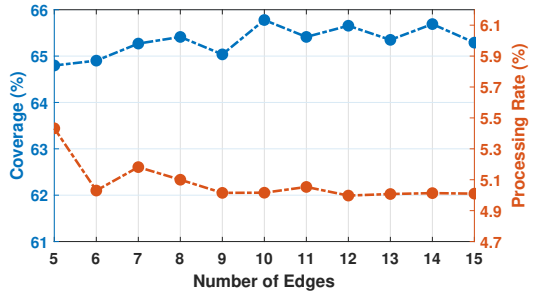
4.3 Coverage-Efficiency Trade-off

Table 3 shows that DMVF can significantly outperform other approaches in the literature. Furthermore, as shown in this section, DMVF enables flexible coverage-efficiency trade-off with different system requirements. Note that when deploying a video fast-forwarding strategy, the goal of achieving high efficiency (i.e., low processing rate) contradicts the goal of maintaining high coverage, and the ability to trade off between the two is desirable.

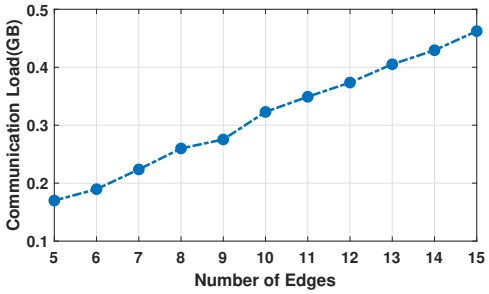
Fig. 6 shows that different trade-offs between coverage and efficiency can be easily achieved in DMVF by changing the X/Y/Z parameters in system requirement. The points on the red curve of DMVF are achieved by the following X/Y/Z: 2/2/2, 2/3/1, 3/2/1, 4/1/1, and 5/0/1. All points outperform the FFNet result. Note that changing these parameters is much more flexible and systematic than deploying FFNet on each agent and manually adjusting their skipping speeds, showing DMVF's capability for reconfiguration and adaptation to accommodate varying system operation needs.

4.4 Impact of Connectivity

For a distributed multi-agent application, such as DMVF, connection pattern among agents and its resulting communication load is an important factor. In real applications, the connections among different agents may vary because of the connection capacity of agents, physical distance between agents, and the network bandwidth, etc. Here, we evaluate the performance and communication load of the DMVF framework under wireless communication with different connection patterns. We use Erdos-Renyi method to generate random graphs as the connection patterns among agents, as shown in Fig. 7. Each random graph is generated with a parameter P, i.e., a fixed probability of each edge being present or absent, independent of the other edges. Higher probability yields higher connectivity of the multi-agent systems. We generate over 40 graphs with different



(a)



(b)

Figure 7: Coverage, processing rate and communication load under varying degree of connectivity in DMVF.

values of P and evaluate their performance with respect to different number of edges in the graph (results are averaged for each number of edges). In the experiment, the total raw input data is 12.36 GB. We observe the following from the results.

- From Fig. 7 (a), we can observe the robustness of DMVF under varying degrees of connectivity. DMVF maintains a higher coverage and lower processing rate when compared to FFNet, regardless of the number of edges in the connection graph.
- From Fig. 7 (b), DMVF leads to only a small amount of communication overhead, from 0.17 GB to 0.46 GB under full connectivity (1.37% to 3.72% of raw data).

4.5 Analysis on Consensus Algorithms

Here, we evaluate DMVF with the various consensus algorithms introduced in Section 3.4 (communication graph is as in Fig. 4). Fig. 8 shows the comparison between FFNet and DMVF variants with respect to average coverage and processing rate. We also evaluate the number of iterations each method needed to reach consensus, and list them together with the data from Fig. 8 in Table 4. From Fig. 8 and Table 4, we have the following observations:

- Compared to FFNet, any variant of DMVF outperforms both in coverage and in processing rate.
- Compared to the gradient descent based variants, i.e., DMVF-DGD and DMVF-EXTRA, the proposed DMVF needs much fewer iterations to reach consensus.
- Among the three maximal consensus based variants, i.e. DMVF, DMVF-AVE, and DMVF-ONE, DMVF has the highest average coverage while having the same magnitude of iteration count.

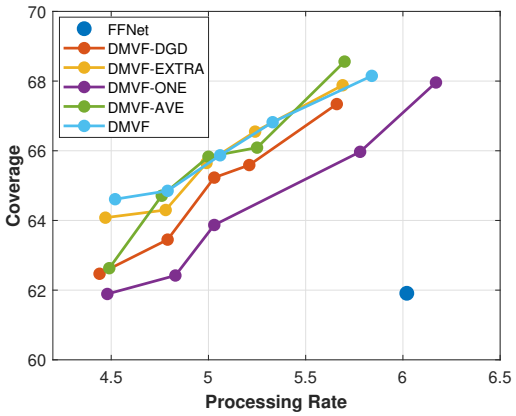


Figure 8: Comparison between DMVF with different consensus algorithms and FFNet.

Methods	Ave. Coverage	Ave. Proc. Rate	Iterations
FFNet	61.91%	6.02%	#
DMVF-AVE	65.56%	5.04%	5
DMVF-ONE	64.42%	5.26%	**4**
DMVF-EXTRA	65.69%	**5.03%**	93.0
DMVF-DGD	64.82%	**5.03%**	234.48
DMVF	**66.06%**	5.11%	5

Table 4: Comparison of coverage, processing rate and iterations of DMVF with different consensus methods.

4.6 Timing Efficiency

We evaluate the timing efficiency of DMVF on the aforementioned distributed embedded platform. Our framework achieves an average frame rate of 313 FPS and a worst-case rate of 280 FPS. Such high frame rate may not be achieved on less-capable embedded platforms, but the low processing rate from DMVF should still reduce computation cost and achieve near real-time speed. For example, even when we intentionally slow down the computation by only using the ARM cores on TX2 (i.e., not using the GPU), DMVF achieves 94 FPS, which is sufficient for most real-time monitoring cases.

5 CONCLUSION

In this paper, we propose a distributed multi-agent video fast-forwarding framework, aka DMVF, that optimizes the coverage and processing rate by enabling agents associated with different cameras to communicate with their neighbors and adaptively update their reinforcement learning based fast-forwarding strategies. Our experiment shows the effectiveness of DMVF when compared with other fast-forwarding methods in the literature.

ACKNOWLEDGMENTS

We gratefully acknowledge the support from NSF grants 1834701, 1834324, 1839511, 1724341, and ONR grant N00014-19-1-2496. Roy-Chowdhury also acknowledges support from CISCO.

REFERENCES

[1] Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdhury. 2007. A survey on wireless multimedia sensor networks. *Computer networks* 51, 4 (2007), 921–960.

[2] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*.

[3] Kai-Yin Cheng, Sheng-Jie Luo, Bing-Yu Chen, and Hao-Hua Chu. 2009. SmartPlayer: user-centric video fast-forwarding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 789–798.

[4] Giovanni Denina, Bir Bhanu, Hoang Thanh Nguyen, Chong Ding, Ahmed Kamal, Chinya Ravishankar, Amit Roy-Chowdhury, Allen Ivers, and Brenda Varda. 2011. Videoweb dataset for multi-camera activities and non-verbal communication. In *Distributed Video Sensor Networks*. Springer, 335–347.

[5] Mohamed Elfeki, Aidean Sharghi, Srikrishna Karanam, Ziyan Wu, and Ali Borji. 2018. Multi-View Egocentric Video Summarization. *arXiv preprint arXiv:1812.00108* (2018).

[6] Ehsan Elhamifar and M Clara De Paolis Kaluza. 2017. Online Summarization via Submodular and Convex Optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[7] Ehsan Elhamifar and Zwe Naing. 2019. Unsupervised Procedure Learning via Joint Dynamic Summarization. In *IEEE International Conference on Computer Vision (ICCV)*.

[8] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. 2012. See all by looking at a few: Sparse modeling for finding representative objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Anders Eriksson, John Bastian, Tat-Jun Chin, and Mats Isaksson. 2016. A Consensus-Based Framework for Distributed Bundle Adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Yanwei Fu, Yanwen Guo, Yanshu Zhu, Feng Liu, Chuanming Song, and Zhi-Hua Zhou. 2010. Multi-view video summarization. *IEEE Transactions on Multimedia* 12, 7 (2010), 717–729.

[11] Boqing Gong, Wei Chao, Kristen Grauman, and Fei Sha. 2014. Diverse Sequential Subset Selection for Supervised Video Summarization. In *Advances in Neural Information Processing Systems (NIPS)*.

[12] Genliang Guan, Zhiyong Wang, Shaohui Mei, Max Ott, Mingyi He, and David Dagan Feng. 2014. A Top-Down Approach for Video Summarization. *ACM Transactions on Multimedia Computing, Communications, and Applications* 11, 1 (2014), 4.

[13] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. 2014. Creating summaries from user videos. In *European Conference on Computer Vision (ECCV)*.

[14] Michael Gygli, Helmut Grabner, and Luc Van Gool. 2015. Video summarization by learning submodular mixtures of objectives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[15] Tavi Halperin, Yair Poleg, Chetan Arora, and Shmuel Peleg. 2017. EgoSampling: Wide View Hyperlapse from Egocentric Videos. *IEEE Transactions on Circuits and Systems for Video Technology* (2017).

[16] A. Jadbabaie, J. Lin, and S. Morse. 2003. Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules. *IEEE Trans. Automat. Control* 48, 6 (2003), 988–1001.

[17] Junfeng Jiang and Xiao-Ping Zhang. 2010. A new player-enabled rapid video navigation method using temporal quantization and repeated weighted boosting search. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference on*.

[18] Junfeng Jiang and Xiao-Ping Zhang. 2011. A smart video player with content-based fast-forward playback. In *Proceedings of the 19th ACM international conference on Multimedia*.

[19] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson. 2008. Subgradient Methods and Consensus Algorithms for Solving Convex Optimization Problems. *Proceedings of IEEE Conference on Decision and Control(CDC)* (2008), 4185–4190.

[20] Neel Joshi, Wolf Kienzle, Mike Toelle, Matt Uyttendaele, and Michael F Cohen. 2015. Real-time hyperlapse creation via optimal frame selection. *ACM Transactions on Graphics* 34, 4 (2015), 63.

[21] Ahmed T Kamal, Jawadul H Bappy, Jay A Farrell, and Amit K Roy-Chowdhury. 2015. Distributed multi-target tracking and data association in vision networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 7 (2015), 1397–1410.

[22] Aditya Khosla, Raffay Hamid, Chih-Jen Lin, and Neel Sundaresan. 2013. Large-scale video summarization using web-image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[23] Gunhee Kim, Leonid Sigal, and Eric P. Xing. 2014. Joint Summarization of Large-scale Collections of Web Images and Videos for Storyline Reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Shuyue Lan, Rameswar Panda, Qi Zhu, and Amit K Roy-Chowdhury. 2018. Ffnet: Video fast-forwarding via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[25] Ita Lifshitz, Ethan Fetaya, and Shimon Ullman. 2016. Human pose estimation using deep consensus voting. In *European Conference on Computer Vision (ECCV)*.

[26] I. Lobel and A. Ozdaglar. 2008. Convergence Analysis of Distributed Subgradient Methods over Random Networks. *Proceedings of Annual Allerton Conference on Communication, Control, and Computing* (2008).

[27] Tao Ma, Michael Hempel, Dongming Peng, and Hamid Sharif. 2013. A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems. *IEEE Communications Surveys & Tutorials* 15, 3 (2013), 963–972.

[28] Y. F. Ma, X. S. Hua, and H. J. Zhang. 2005. A Generic Framework of User Attention Model and Its Application in Video Summarization. *IEEE Transactions on Multimedia* (2005).

[29] I. Matei and J. S. Baras. 2011. Performance Evaluation of the Consensus-Based Distributed Subgradient Method Under Random Communication Topologies. *IEEE Journal of Selected Topics in Signal Processing* 5, 4 (2011), 754–771.

[30] Angelia Nedić. 2011. Asynchronous broadcast-based convex optimization over a network. *IEEE Trans. Automat. Control* 56, 6 (2011), 1337–1351.

[31] Angelia Nedic and Asuman Ozdaglar. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Automat. Control* 54, 1 (2009), 48–61.

[32] A. Nedić, A. Ozdaglar, and P. A. Parrilo. 2010. Constrained Consensus and Optimization in Multi-agent Networks. *IEEE Trans. Automat. Control* 55(4) (2010), 922–938.

[33] Shun-Hsing Ou, Chia-Han Lee, V Srinivasa Somayazulu, Yen-Kuang Chen, and Shao-Yi Chien. 2014. Low complexity on-line video summarization with gaussian mixture model based clustering. In *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*.

[34] Shun-Hsing Ou, Chia-Han Lee, V Srinivasa Somayazulu, Yen-Kuang Chen, and Shao-Yi Chien. 2015. On-line multi-view video summarization for wireless video sensor network. *IEEE Journal of Selected Topics in Signal Processing* 9, 1 (2015), 165–179.

[35] Rameswar Panda, Abir Das, Ziyan Wu, Jan Ernst, and Amit K Roy-Chowdhury. 2017. Weakly Supervised Summarization of Web Videos. In *IEEE International Conference on Computer Vision (ICCV)*.

[36] Rameswar Panda, Abir Dasy, and Amit K Roy-Chowdhury. 2016. Video summarization in a multi-view camera network. In *23rd International Conference on Pattern Recognition (ICPR)*.

[37] Rameswar Panda and Amit K Roy-Chowdhury. 2017. Collaborative Summarization of Topic-Related Videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[38] Rameswar Panda and Amit K Roy-Chowdhury. 2017. Multi-view surveillance video summarization via joint embedding and sparse optimization. *IEEE Transactions on Multimedia* 19, 9 (2017), 2010–2021.

[39] Kadir A Peker, Ajay Divakaran, et al. 2003. An extended framework for adaptive playback-based video summarization. In *Internet Multimedia Management Systems IV*.

[40] Kadir A Peker, Ajay Divakaran, and Huifang Sun. 2001. Constant pace skimming and temporal sub-sampling of video using motion activity. In *IEEE International Conference on Image Processing (ICIP)*.

[41] Nemanja Petrovic, Nebojsa Jojic, and Thomas S Huang. 2005. Adaptive video fast forward. *Multimedia Tools and Applications* 26, 3 (2005), 327–344.

[42] Yair Poleg, Tavi Halperin, Chetan Arora, and Shmuel Peleg. 2015. Egosampling: Fast-forward and stereo for egocentric videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[43] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. 2014. Category-specific video summarization. In *European Conference on Computer Vision (ECCV)*.

[44] Washington LS Ramos, Michel M Silva, Mario FM Campos, and Erickson R Nascimento. 2016. Fast-forward video based on semantic extraction. In *IEEE International Conference on Image Processing (ICIP)*.

[45] Mrigank Rochan and Yang Wang. 2019. Video Summarization by Learning from Unpaired Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[46] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. 2015. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization* 25, 2 (2015), 944–966.

[47] Gunnar A Sigurdsson, Xinlei Chen, and Abhinav Gupta. 2016. Learning visual storylines with skipping recurrent neural networks. In *European Conference on Computer Vision (ECCV)*.

[48] Michel Melo Silva, Washington Luis Souza Ramos, Joao Pedro Klock Ferreira, Mario Fernando Montenegro Campos, and Erickson Rangel Nascimento. 2016. Towards Semantic Fast-Forward and Stabilized Egocentric Videos. In *European Conference on Computer Vision (ECCV)*.

[49] Dhananjay Singh, Gaurav Tripathi, and Antonio J Jara. 2014. A survey of Internet-of-Things: Future vision, architecture, challenges and services. In *Internet of things (WF-IoT), 2014 IEEE world forum on*.

[50] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. 2015. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[52] John Nikolas Tsitsiklis. 1984. *Problems in decentralized decision making and computation*. Technical Report. Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems.

[53] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[54] Hanzi Wang and David Suter. 2006. Background subtraction based on a robust consensus method. In *18th International Conference on Pattern recognition (ICPR)*.

[55] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. 2019. AdaFrame: Adaptive Frame Selection for Fast Video Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[56] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. 2016. Summary Transfer: Exemplar-based Subset Selection for Video Summarizatio. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[57] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. 2016. Video summarization with long short-term memory. In *European Conference on Computer Vision (ECCV)*.

[58] Bin Zhao and Eric P Xing. 2014. Quasi real-time summarization for consumer videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.