Collaborative Multi-Agent Video Fast-Forwarding

Shuyue Lan, Zhilu Wang, Ermin Wei, *Member, IEEE*, Amit K. Roy-Chowdhury, *Fellow, IEEE*, and Qi Zhu, *Member, IEEE*

***Abstract*—Multi-agent applications have recently gained significant popularity. In many computer vision tasks, a network of agents, such as a team of robots with cameras, could work collaboratively to perceive the environment for efficient and accurate situation awareness. However, these agents often have limited computation, communication, and storage resources. Thus, reducing resource consumption while still providing an accurate perception of the environment becomes an important goal when deploying multi-agent systems. To achieve this goal, we identify and leverage the overlap among different camera views in multi-agent systems for reducing the processing, transmission and storage of redundant/unimportant video frames. Specifically, we have developed two collaborative multi-agent video fast-forwarding frameworks in distributed and centralized settings, respectively. In these frameworks, each individual agent can selectively process or skip video frames at adjustable paces based on multiple strategies via reinforcement learning. Multiple agents then collaboratively sense the environment via either 1) a consensus-based distributed framework called DMVF that periodically updates the fast-forwarding strategies of agents by establishing communication and consensus among connected neighbors, or 2) a centralized framework called MFFNet that utilizes a central controller to decide the fast-forwarding strategies for agents based on collected data. We demonstrate the efficacy and efficiency of our proposed frameworks on a real-world surveillance video dataset VideoWeb and a new simulated driving dataset CarlaSim, through extensive simulations and deployment on an embedded platform with TCP communication. We show that compared with other approaches in the literature, our frameworks achieve better coverage of important frames, while significantly reducing the number of frames processed at each agent.**

***Index Terms*—Video fast-forwarding, multi-agent systems, reinforcement learning.**

I. Introduction

With the rapid advancement of camera sensors, a network of agents with cameras are increasingly being explored for tasks such as search and rescue, wide-area surveillance, and environmental monitoring, where the cameras may be built-in cameras in robots, cameras on drones, or fixed surveillance cameras. In these systems, multiple cameras can observe the same environment and generate videos from different angles, often with overlapping views, so that the fusion of all their perceptions may lead to better scene understanding. For many application tasks, this information fusion of large amount of data needs to be performed in real time or near real time. However, the agents often have limited computation, communication, storage, and energy resources, which makes processing and transmitting all the video data quite challenging. This thus motivates the development of methods that can select an informative subset of the video frames to focus on.

In the relevant literature, *video summarization* and *video fast-forwarding* both aim at generating a compact summary of the original video. In particular, video summarization methods often summarize videos in an offline manner, which needs an entire video available at hand before processing it [1], [2], [3], [4], [5]. Multi-view summarization methods that summarize videos from multiple cameras have also been proposed [6], [7], [8], [9], [10]. However, as these methods process the entire videos and are often time-consuming, they are unsuitable for online and real-time applications. On the other hand, video fast-forwarding methods generate the video summary on the fly. Most of such methods adjust the playback speed of a video [11], [12], [13], [14], [15], [16], [17] while processing the entirety of it. One exception is our previous work FFNet [18], which performs video fast-forwarding for a single camera in an online manner and only processes a fraction of the video frames by automatically skipping unimportant frames via reinforcement learning. This shows promising results in reducing system computation and storage load. In this work, we build upon this approach and develop our solution for multi-agent video fast-forwarding systems.

A. Solution Overview

Motivated by the observation that there is often significant overlap among videos captured by cameras from different angles in multi-agent systems, we pose the following question: *Is it possible to leverage the overlapping among different views in multi-agent perception to collaboratively perform fast-forwarding that is efficient, causal, online, and results in an informative summary for the scene in real time?*

In this paper, we introduce two methods for multi-agent video fast-forwarding in distributed and centralized settings, respectively. We target the scenarios where cameras at multiple agents observe the same environment from different angles. Each camera embeds a fast-forwarding agent with multiple strategies,

Manuscript received 9 April 2022; revised 18 October 2022 and 20 December 2022; accepted 26 April 2023. Date of publication 12 May 2023; date of current version 18 January 2024. This work was supported by NSF under Grants 1834701, 1724341, 2038853, and 2024774 and in part by ONR under Grant N00014-19-1-2496. The Associate Editor coordinating the review of this manuscript and approving it for publication was Prof. Ling-Yu Duan. *(Corresponding author: Qi Zhu.)*

Shuyue Lan, Zhilu Wang, Ermin Wei, and Qi Zhu are with the Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208-0001 USA (e-mail: shuyuelan2018@u.northwestern.edu; zhilu.wang@u.northwestern.edu; ermin.wei@northwestern.edu; qzhu@northwestern.edu).

Amit K. Roy-Chowdhury is with the Electrical and Computer Engineering, University of California, Riverside, CA 92521 USA (e-mail: amitrc@ece.ucr.edu).

Digital Object Identifier 10.1109/TMM.2023.3275853

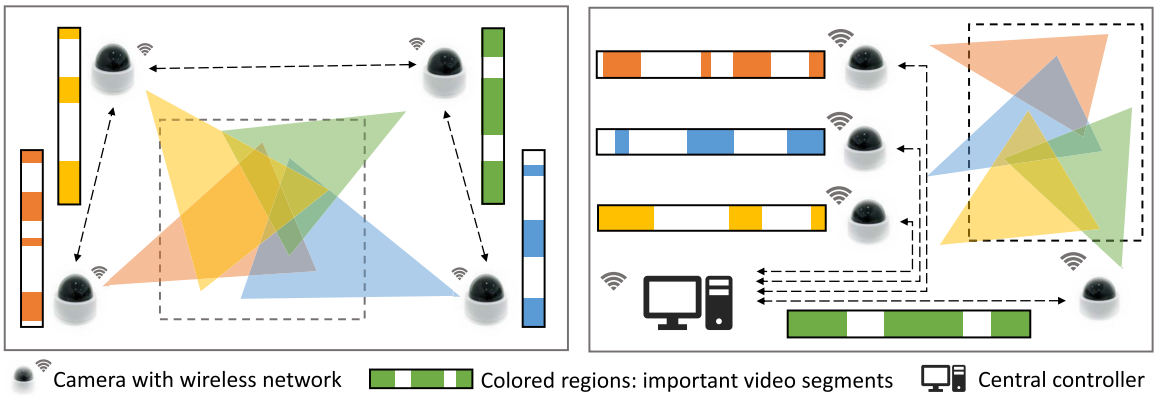


Fig. 1. Illustration of collaborative multi-agent video fast-forwarding. Multiple cameras at different agents are observing the same environment from different overlapping views. Each camera performs video fast-forwarding according to its current fast-forwarding strategy, which is decided either via communication and consensus among neighboring agents in a distributed manner (left) or by a central controller that analyzes the data from each agent (right). The colored regions within the bars represent the important video segments that each agent sees in its view.

i.e., it can skip the frames of its video input at different paces (e.g., slow, normal, or fast). During operation, each camera fast-forwards its own video stream based on a chosen pace and periodically updates its fast-forwarding strategies.

For the distributed setting, part of our work has appeared in [19], named **DMVF**, which chooses and updates fast-forwarding strategies by establishing communication and consensus among connected agents, as shown in the left figure of Fig. 1. Agents are connected by a predetermined undirected communication network,[1] where each agent can communicate with a set of neighboring agents. At every adaptation period, each agent evaluates the importance of the selected frames from itself and those from its neighbors by comparing their similarities. Then a system-wide consensus algorithm is run among agents to reach an agreement on the importance score for every agent's view. Finally, based on the score ranking and the system requirement, each agent selects a fast-forward strategy for its next adaptation period.

For the centralized multi-agent video fast-forwarding setting, we have developed a new framework in this work, named **MFFNet**, which contains a central controller to decide the fast-forwarding strategies for each agent (the right part of Fig. 1). During operation, each camera fast-forwards its own video stream based on a chosen pace given by the central controller, and periodically sends selected frames (i.e., fast-forwarded clips) to the central controller. The central controller receives the selected frames from every agent and composes a more compact summary video for the scene. Moreover, based on the data at hand, the central controller infers the strategy/pace that should be adopted by each agent for the next period and sends such instruction back to the agents. Intuitively, an agent whose view currently contains more important frames than others should be slowed down for the next period to collect more frames; while agents whose views have significant overlaps with the slowed-down agents can be given a faster pace to reduce their processing and transmission load.

In both distributed and centralized settings, each agent only processes a very small portion of frames with fast-forwarding, which significantly reduces the computation load. The agents also do not require transmitting or storing their entire video streams (often only a fraction of them). From the system perspective, both the intra-view at each agent and the inter-view redundancy across different agents are reduced. Furthermore, the online and causal nature of our proposed approaches 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.

The main contributions of this paper include the following.

- We formulate the multi-agent video fast-forwarding problem as a collaborative multi-agent reinforcement learning problem. Each agent can fast-forward its video input without processing the entire video and be easily adapted to different fast-forwarding strategies/paces.
- Building upon our work in single-agent fast-forwarding (FFNet) [18] and distributed multi-agent fast-forwarding (DMVF) [19], we develop a new centralized framework MFFNet for multi-agent fast-forwarding, which uses a central controller to orchestrate the fast-forwarding strategies of agents for achieving better scene coverage with reduced computation and communication load.
- We demonstrate the effectiveness of MFFNet on a challenging multi-view dataset, VideoWeb [20], achieving real-time speed on an embedded platform with TCP communication. We compare MFFNet with DMVF, FFNet, and a few other methods in the literature.
- Moreover, for a more comprehensive comparison, we also include a newly generated multi-camera dataset for multi-agent video fast-forwarding, named CarlaSim, to further evaluate the various methods on moving platforms.

In particular, beyond our recent work [19], this paper introduces the new development of 1) the MFFNet method, 2) the

[1]Note that some agents may not be able to communicate with each other due to practical factors such as the connection capacity of camera nodes, the physical distances between the nodes, the network bandwidth, etc.

new CarlaSim dataset, and 3) the experimental results and analysis of MFFNet, as well as its comparison with DMVF, FFNet and other methods on VideoWeb and CarlaSim.

B. Paper Organization

This paper highlights our new contributions in MFFNet and also introduces our prior work in FFNet and DMVF, providing a holistic view of our solution in video fast-forwarding. More specifically, FFNet is a single-agent video fast-forwarding method that we developed based on reinforcement learning, and we build a multi-strategy video fast-forwarding agent upon FFNet. Both DMVF and MFFNet use this multi-strategy fast-forwarding agent on their camera nodes – DMVF uses a distributed framework to decide the strategies each agent should use, while MFFNet uses a centralized framework to do so. Both methods are efficient and effective on collaborative video fast-forwarding for a network of resource-limited agents.

In the rest of the paper, we first present a review of relevant literature in Section II. This is followed by a review of our work in developing FFNet for single-agent video fast-forwarding in Section III, along with the development of a multi-strategy video fast-forwarding agent. In Sections IV and V, we present our solutions to the multi-agent video fast-forwarding problem for distributed and centralized settings, i.e., DMVF and MFFNet, respectively. Experimental results in real-life data are presented in Section VI.

II. Related Work

A. Video Summarization and Video Fast-Forwarding

The objective of video summarization is to take an entire video as input and output a compact subset of frames that can describe the important content of the original video. Many single-view video summarization methods are developed with unsupervised learning [21], [22], [23], [24] and supervised learning techniques based on video-summary labels [2], [3], [4], [25], [26], [27]. There are methods proposed specifically for summarizing crawled web images/videos [28], [29], [30], [31] and photo albums [32], and online methods developed using submodular optimization [1], Gaussian mixture model [33], and online dictionary learning [5]. Beyond single-view, the multi-view video summarization problem has been addressed by random walk over spatio-temporal graphs [6], joint embedding and sparse optimization [7], [8], DPP (Determinantal Point Processes) [9], and a two-stage system with online single-view summarization and distributed view selection [10]. Different from these methods, our approaches do not process all the frames, which significantly reduces computation and communication load, and they collaboratively fast-forward multi-view videos, further improving the efficiency and coverage.

Video fast-forwarding methods are used for skipping uninteresting/unimportant parts of the video. Commercial video players often offer the users with manual control on the playback speed, such as Apple QuickTime player with 2x, 5x, and 10x speed fast-forward. In the literature, the playback speed can be automatically adjusted based on the similarity of each candidate clip to a query clip [14] and the motion activity patterns in videos [11], [34], [35]. Besides playback speed adjustment, some works develop the fast-forwarding policy based on mutual information between frames [36], [37], shortest path distance over the semantic graph built from frames [16], [17], and visual and textual features [38]. Hyperlapse is also widely studied for fast-forwarding videos aiming at speed-up and smoothing [12], [13], [15]. Different from these approaches that are for single videos, our work focuses on multi-agent video fast-forwarding methods that collaboratively fast-forward videos from different views.

B. Reinforcement Learning

Deep reinforcement learning has been widely used in many computer vision tasks and achieved promising performance, such as in action detection [39], object detection [40], image captioning [41], pose estimation [42], visual tracking [43] and query-conditioned video summarization [44]. There are also approaches applying reinforcement learning to the multi-agent domain, i.e., multi-agent reinforcement learning (MARL) (see a detailed review in [45]). Some recent works have used MARL to address computer vision tasks, such as joint object search [46], multi-object tracking [47], and frame sampling for video recognition [48]. There are also works on building learnable communication protocols for collaborative multi-agent deep reinforcement learning [49], [50]. Our earlier work FFNet conducts single video fast-forwarding via reinforcement learning [18], based on which we further develop two approaches for multi-agent video fast-forwarding in centralized and distributed settings.

C. Multi-Agent System Optimization

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 [51] proposes a distributed consensus protocol for achieving agreement in a multi-agent setting by iteratively taking a weighted average with local neighbors. The work in [52] 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 [53], [54], [55] extend this method to other settings, including stochastic networks, constrained problems, and noisy environments. More recently, EXTRA [56], 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 step size. In computer vision, consensus-based methods are used applications such as human post estimation [57], background subtraction [58], and multi-target tracking [59], etc. To the best of our knowledge, the DMVF framework (more details on [19]) we developed is the first distributed consensus-based framework to address multi-agent video fast-forwarding. In this paper, we further develop a centralized framework MFFNet that facilitates a central controller to adjust the fast-forwarding strategy for multi-agent video fast-forwarding.

III. Single-Agent Video Fast-Forwarding

A. Review of FFNet

FFNet [18] uses a Markov decision process (MDP) to formulate the video fast-forwarding problem and solves it using reinforcement learning, i.e., with a Q-learning agent that learns a policy to skip unimportant frames and present the important ones for further processing. Given the current frame, FFNet decides the number of frames to skip next. The MDP formulation of FFNet is defined as follows:

- *State:* A state s_k describes the environment at time step k. It is defined as the feature vector of the current frame.
- *Action:* An action a_k is performed by the system at step k and devotes to an update of the state. The action set includes the possible numbers of frames to skip.
- *Reward:* An immediate reward $r_k = r(s_k, a_k, s_{k+1})$ is received by the system at time step k as

$$r_k = -SP_k + HR_k. \tag{1}$$

It consists of the "skip" penalty (SP) and the "hit" reward (HR). SP_k defines the penalty for skipping action in the interval t_k at step k:

$$SP_k = \frac{\sum_{i \in t_k} \mathbf{1}(l(i) = 1)}{T} - \beta \frac{\sum_{i \in t_k} \mathbf{1}(l(i) = 0)}{T}, \tag{2}$$

where $\mathbf{1}(\cdot)$ is an indicator function that equals to 1 if the condition holds. T is the largest number of frames we may skip. $\beta \in [0, 1]$ is a trade-off factor between the penalty for skipping important frames and the reward for skipping unimportant frames. HR_k defines the reward for jumping to an important frame or a position near an important frame and is computed as

$$HR_k = \sum_{i=z-w}^{z+w} \mathbf{1}(l(i) = 1) \cdot f_i(z), \tag{3}$$

where $f_i(z)$ extends the one-frame label at frame i to a Gaussian distribution in a neighboring time window w, i.e., $z \in [i - w, i + w]$.

- *Policy:* With the definition of states, actions, and rewards, a skipping policy π is learned for selecting the action that maximizes the expected accumulated reward R:

$$\pi(s_k) = arg \max_a E[R|s_k, a, \pi], \tag{4}$$

where the accumulated reward R is computed as

$$R = \sum_k \gamma^{k-1} r_k = \sum_k \gamma^{k-1} r(s_k, a_k, s_{k+1}), \tag{5}$$

where $\gamma \in [0, 1]$ denotes the discount factor for the rewards in the future.

With Q-learning, the value of $E[R|s, a, \pi]$ is evaluated as $Q(s, a)$. The optimal value $Q^*(s_k, a_k)$ can be calculated by the Bellman equation in a recursive fashion:

$$Q^*(s_k, a_k) = r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1}). \tag{6}$$

The model of FFNet is shown in Fig. 2. When training this model, the mean squared error between the target Q-value and the output of MLP is used as the loss function. ϵ-greedy strategy is utilized to better explore the state space, which picks a random action with probability ϵ and the action that has $Q^*(s, a)$ with probability 1-ϵ.

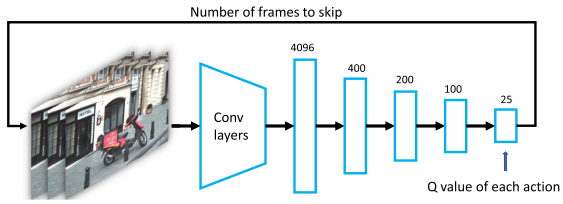


Fig. 2. Model structure of FFNet. It takes a frame in an incoming video stream as an input for the deep neural network and outputs the number of frames to skip.

B. Multi-Strategy Fast-Forwarding Agent

To fit into the multi-agent video fast-forwarding scenario, on each camera that captures a view of the scene, we leverage a multi-strategy fast-forwarding agent that can adaptively fast-forward the incoming videos with different paces. Similar to [19], the FFNet is derived into 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 adopts the same immediate reward design as FFNet:

$$r_k(normal) = -SP_k + HR_k. \tag{7}$$

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

$$r_k(slow) = (-SP_k + HR_k) \times \left(1 - \frac{sigmoid(a_k)}{2}\right). \tag{8}$$

Intuitively, if the agent skips a larger step, it will receive a 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. Thus, we modify the immediate reward at time step k as

$$r_k(fast) = (-SP_k + HR_k) \times \left(1 + \frac{sigmoid(a_k)}{2}\right). \tag{9}$$

This reward definition ensures that the agent will get a larger immediate reward if it skips a larger step. The action space is set to 35 to allow the agents to skip larger steps. For each agent, it can flexibly switch among these strategies to adaptively fast-forward its own videos.

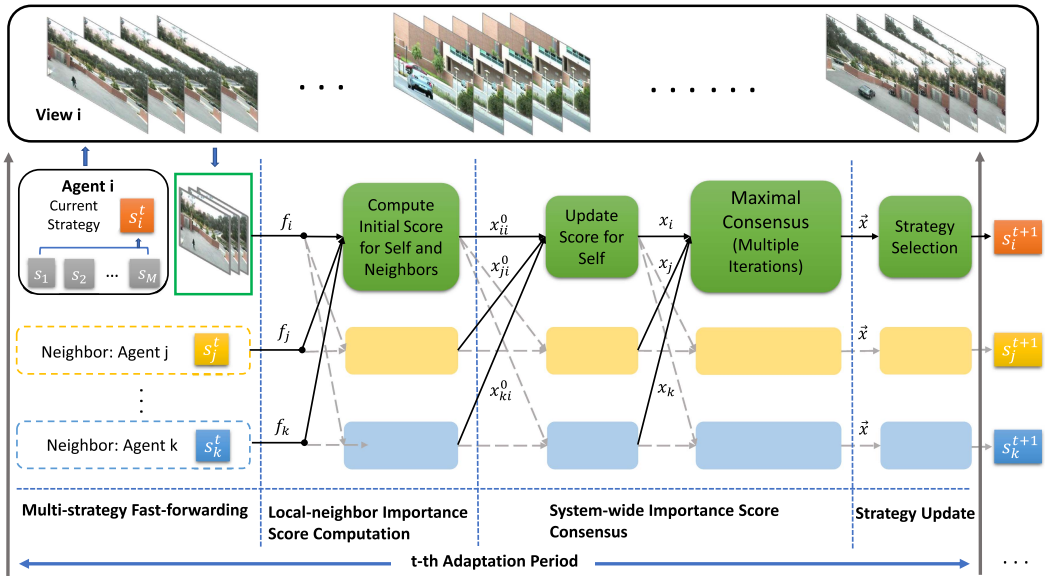


Fig. 3. Workflow of DMVF. At every adaptation period t, each agent i first fast-forwards its video input with current strategy s_i^t and selects a set of frames f_i. It then receives neighbor agents' selected frames (e.g., f_j and f_k) and computes an initial importance score for itself and its neighbors. Afterwards, agent i refines and finalizes the importance score with other agents via a system-wide maximal consensus algorithm. 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).

IV. DMVF: Distributed Multi-Agent Video Fast-Forwading

A. Overview

In this section, we review our approach for addressing the multi-agent video fast-forwarding problem by adapting the skipping strategy of each agent in an efficient, online, and distributed manner, named DMVF (more details in [19]). Fig. 3 shows the workflow design of our framework (take one agent i for illustration). Given the incoming multi-view video streams $V = \{v_1, \ldots, v_N\}$ captured at different agents, our goal is to generate a final summary $F = \{f_1, \ldots, f_N\}$ 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 $S = \{s_m, m = 1, \ldots, M\}$. 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. Note that the frames being skipped are not processed, transmitted, or saved. 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. 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. Note that during the consensus process, only scores are transmitted (not selected frames). After running the consensus algorithm, each agent will have the same copy of the final importance scores for their selected frames in the current period, defined as $\vec{x} = [x_1, x_2, \ldots, x_N]$. 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. The notations are highlighted in Table I.

TABLE I
Notations Used in DMVF

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

B. Local-Neighbor Importance Score Computation

In this step, for every agent i, we compute an initial importance score for itself and its neighbor 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:

$$sim(x, y) = e^{-\alpha||x-y||_2}, \tag{10}$$

where α is used to scale the L2-norm to restrict the similarity value to a satisfactory range ($\alpha = 0.05$ in our experiment).

The similarity of agent j to i is then defined as

$$sim\_agent(v_i, v_j) = \frac{1}{|v_j|} \sum_{s=1}^{|v_j|} \max_{1 \le a \le |v_i|} sim(p_s(v_j), p_a(v_i)), \tag{11}$$

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

$$x_{ij}^0 = \begin{cases} \frac{1}{|V_i|-1} \sum_{v_k \in V_i, k \neq j} sim\_agent(v_j, v_k) \\ \qquad \text{if } i = j \text{ or } (i,j) \in E \\ 0 \qquad\qquad\qquad\qquad \text{o.w.} \end{cases} \tag{12}$$

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 as described in the following section.

C. 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 in [19].

There are three steps in our maximal consensus 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. $\{x_{ji}^0\}$, $j \in V_i$). Then, in the second step, agent i updates its score as

$$x_i = \frac{\sum_{j \in V_i} \frac{1}{n_j} x_{ji}^0}{\sum_{j \in V_i} \frac{1}{n_j}}, \tag{13}$$

which means that 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 $i\text{-}th$ 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., $\vec{x}_i = \vec{x} = [x_1, x_2, \ldots, x_N]$.

D. 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. Given the system requirement, the portions of different strategies are pre-defined, which means there should be a fixed number of agents under each strategy after every update.

TABLE II
NOTATIONS USED IN MFFNET

M	number of available fast-forwarding strategies
N	number of camera views / agents
v_n	the video of view n, $n \in [1, N]$.
V	the set of N views
V'	the subset of V containing selected main views
s_m	available strategy m, $m \in [1, M]$
$\bar{s_n}$	strategy being used in agent n
$\hat{s_n}$	strategy for agent n in the next period
$\{A_n\}$	set of fast-forwarding agents, $\{A_1, \cdots, A_N\}$
$\{B_n\}$	set of buffers, $\{B_1, \cdots, B_N\}$
$\{b_n\}$	set of data received by controller, $\{b_1, \cdots, b_N\}$
F	summary of the scene: $\{f_1, \cdots, f_N\}$
T	period of strategy update
ρ	the threshold for matching frames

V. MFFNET: CENTRALIZED MULTI-AGENT VIDEO FAST-FORWADING

A. Overview

In this section, we present a new method to address the multi-agent video fast-forwarding problem by utilizing a central controller to analyze the data from each agent and adapt the fast-forwarding strategies of agents in an efficient online manner, named MFFNet. Fig. 4 shows the workflow design of our framework. Given the incoming multi-view video streams $V = \{v_1, \ldots, v_N\}$ captured at different agents, the goal of MFFNet is to generate a final summary $F = \{f_1, \ldots, f_N\}$ for the scene while reducing the computation, communication, and storage load.

The fast-forwarding agent of each camera view is modeled as a reinforcement learning agent with multiple available strategies $\{s_m, m = 1, \ldots, M\}$. During operation, each agent n fast-forwards its own video stream with a current strategy $\bar{s}_n$ and keeps the selected frames in its buffer B_n. The frames being skipped are not processed, transmitted, or saved. After a period of time T, each agent sends the selected frames in its buffer to the central controller. The central controller receives selected frames of the last period from all agents and computes their similarity. Based on the similarity computation, the controller chooses the strategy $\hat{s}_n$ for each agent n in the next period and notifies them immediately. Such computation and decision are very fast and only performed once every period. The central controller also generates a more compact summary of the selected frames and stores them. The notations are highlighted in Table II.

B. Central Controller

The responsibility of the central controller is to decide the pace for each agent and generate a more compact summary of the scene. At every period T, it receives the selected frames $\{b_1, \ldots, b_N\}$ from all agents. With those data, it first computes similarity among frames from different agents. Based on the similarity, the central controller decides the new strategies $\{\hat{s}_1, \ldots, \hat{s}_n\}$ for all agents and sends them back. Meanwhile, the controller further reduces redundancy by generating a compact summary $F = \{f_1, \ldots, f_n\}$. The central controller consists

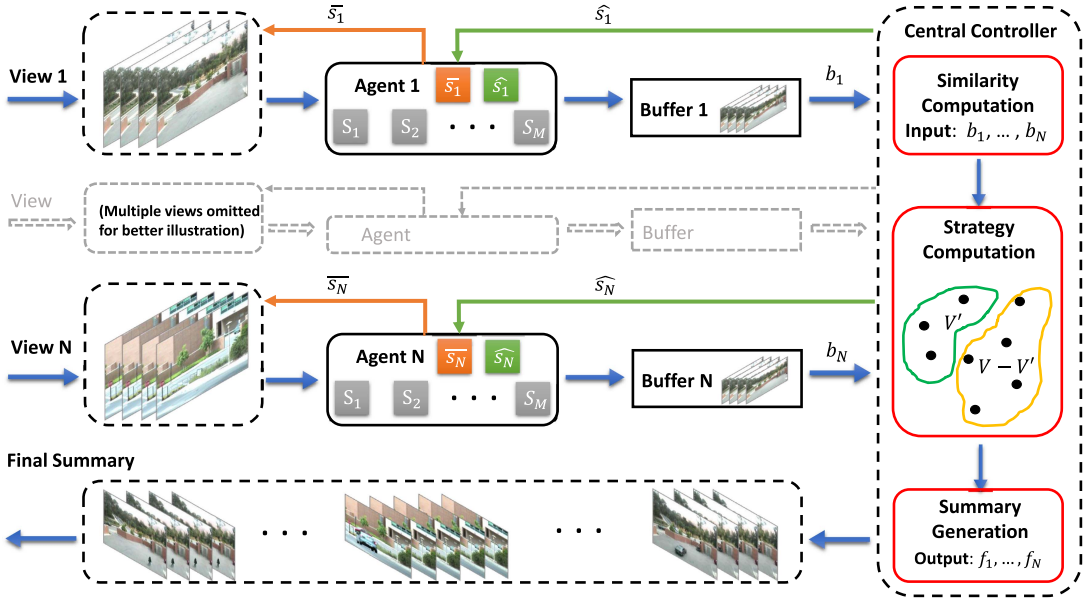


Fig. 4. Workflow overview of MFFNet. Each camera view is associated with an adaptive fast-forwarding agent that supports multiple fast-forwarding strategies/paces. During every period of operation, each agent n uses current strategy $\bar{s_n}$ to fast-forward its video input and saves selected frames in its buffer. At the end of the period, every agent sends the selected frames in its buffer to the central controller. The central controller computes the similarity among the frames from different agents, and based on it, chooses the strategy $\hat{s_n}$ for each agent n in the next period and generates a more compact summary from their selected frames.

of three modules: similarity computation, strategy computation, and summary generation.

Similarity Computation: From each agent n, the central controller receives a set of frames b_n per period. In this module, the similarity between two frames is defined in (10) in Section IV. A threshold ρ is used to match frames. If the similarity of two frames is greater than ρ, we consider them as a match. In order to further compute the strategies for each agent, we define a function named match count $M(\cdot,\cdot)$, which matches frames from two sources and returns the number of matching frames, as shown below:

$$M(u,v) = \sum_{x \in u} I(\max_{y \in v}(sim(x,y)) > \rho), \tag{14}$$

where $I(\cdot)$ is an indicator function that equals 1 if the condition holds.

Strategy Computation: The goal of the strategy computation module is to infer the strategies for all agents in the next period, i.e., $\{\hat{s_1}, \dots, \hat{s_n}\}$. Intuitively, if a view contains a larger number of important frames, it should receive more attention and should not be skipped too much. Following this idea, we formulate the strategy computation problem as an optimization problem for selecting a subset of views V' as the main views from V to better represent the whole scene. The set of main views is selected by

$$V' = arg\max_{\bar{V}} \frac{\sum_{i \in V-\bar{V}} M(b_i, \bigcup_{j \in \bar{V}} b_j)}{\sum_{j \in \bar{V}} len(b_j)}, \tag{15}$$

where b_i and b_j are the frames sent back by the fast-forward agents i and j. $len(\cdot)$ represents the number of frames in a fast-forwarded segment. The set of main views is selected as the subset of views that can cover the most of other views. To avoid the effect of the main view size, we divide the sum of match counts of other views by the total number of frames in the main view set. The detailed algorithm for main view set selection is shown in Algorithm 1. Here, sz denotes the total number of frames in the main views. $score$ is the main view score of the subset $\bar{V}$, as in (15).

For the views in the main view set V', they can cover more content than other views and have more important information. Thus, we use the slow fast-forwarding strategy for each of them. For any other views in $V - V'$, they can be covered significantly by the main views. Therefore, we expect them to fast-forward at a faster speed. More specifically, we decide their strategies by their matching percentage to the main view set. The matching percentage of view n is computed as $mp(n) = M(b_n, \bigcup_{j \in \bar{V}} b_j)/len(b_n)$. If the matching percentage of a view is smaller than a threshold τ, it will be instructed to maintain the normal pace; otherwise, it will be instructed to use the fast strategy, as below:

$$\hat{s}_n = \begin{cases} slow, & if \quad n \in V', \\ normal, & if \quad n \in V - V', mp(n) < \tau, \\ fast, & if \quad n \in V - V', mp(n) > \tau. \end{cases} \tag{16}$$

Data Buffer and Strategy Update: For each agent n, there is a data buffer B_n for storing the selected frames. At every time period T, the agent will send those frames in the buffer to the central controller. The agent will also receive a new strategy/pace instruction from the central controller and adapt it accordingly in the next period.

Summary Generation: After matching the frames among views and choosing the main view, the next step is to generate a more compact summary for the scene. We use the following policy for further reducing redundancy: 1) for the set of main views V', we keep all the frames from its buffer in the summary, and 2)

Algorithm 1: Main View Set Selection Algorithm

Input: a set of data received by the controller, $\{b_1, \ldots, b_N\}$, the similarity threshold ρ
Output: A set of selected main views V'
Initialize the similarity array $Similarity$
for $i = 1$ to N **do**
  for $j = 1$ to $N, j \neq i$ **do**
    for $k = 1$ to $Size(b_i)$ **do**
      for $l = 1$ to $Size(b_j)$ **do**
        $Similarity[i, j, k, l] = sim(b_i[k], b_j[l])$
$MaxScore = 0$
for $\delta = 1$ to $(2^N - 2)$ **do**
  $\bar{V} = \{\}, sz = 0, score = 0$
  for $i = 1$ to N **do**
    if the i-th bit of δ is 1
      $\bar{V} \leftarrow \bar{V} \cup \{i\}$
      $sz \leftarrow sz + Size(b_i)$
  for $i = 1$ to $N, i \notin \bar{V}$ **do**
    for $k = 1$ to $Size(b_i)$ **do**
      $match = 0$
      for $j \in \bar{V}$ **do**
        for $l = 1$ to $Size(b_j)$ **do**
          if $sim(b_i[k], b_j[l]) > \rho$ **then**
            $match \leftarrow 1$
      $score \leftarrow score + match$
  $score \leftarrow score/sz$
  if $score > MaxScore$ **then**
    $V' \leftarrow \bar{V}, MaxScore \leftarrow score$

for the other views, we remove the frames that are matched with the main views (i.e., similar to some frames in the main views) and only keep the remaining ones in the summary. Please note that when generating the summary, we restrict the reduction of frames within a certain time window. That is, if two frames are similar with respect to the similarity threshold ρ and are close to each other in time, we consider them as a match and drop it. Finally, similarly to [18], we also include some neighboring frames of the selected frames in the summary (with selected ones as the window centroids). All these summary frames are denoted as $\{f_1, \ldots, f_n\}$.

C. Central Controller Using RL

In addition, we design another central controller using deep reinforcement learning with our framework. The central controller acts as a feedback-loop controller system, which can be formulated as an MDP with the following definitions of key elements.

State: In our scenario, the fast-forwarded videos of period k from multiple agents are integrated into a single description, which is taken as the state s_k. To be more specific, we consider the concatenation of the average feature vector as a state, which is based on the fast-forwarded frames of different agents.

Action: At each control period k, we consider the action as the combination of different fast-forwarding strategies used in each agent. As we have M available fast-forwarding strategies for all N agents, i.e. $S = \{s^m, m \in [1, M]\}$, the entire action space $AS = \{a^1, a^2, \ldots, a^P\}$, where $P = M^N$.

Reward: After taking one action, i.e., selecting the proper fast-forwarding for each agent, the system transits from state s_k to another state s_{k+1} and an immediate reward $r_k = r(s_k, a_k, s_{k+1})$ is received by the system. The accumulated reward is further defined as

$$R = \sum_k \gamma^{k-1} r(s_k, a_k, s_{k+1}), \tag{17}$$

where $\gamma \in [0, 1]$ is the discount factor for the rewards in the future. The goal of the central controller is to control the fast-forwarding paces of agents to maximize the coverage of the important scenes across multiple views and reduce the redundancy in the final summarized videos, by taking a sequence of actions. For a video available in the training set, we assume that the label of it is a binary vector, in which 1 indicates an important frame and 0 means an unimportant one.

After receiving the strategy instruction from the central controller, each agent will fast-forward its own video stream with the corresponding model and transmit the fast-forwarded video segment during the current control period. During a period of time T, an agent n sends its fast-forwarded frames b_n and the corresponding binary vector of selected frames $\hat{y}_n$ back to the central controller. With this information from all agents, the central controller receives the immediate reward at step k computed by the following equation:

$$r_k = \sum_{n=1}^{N} g(\hat{\mathbf{y}}_{\mathbf{n},\mathbf{k}})^T g(\mathbf{y}_{\mathbf{n},\mathbf{k}}) + \alpha \frac{\|g(\bar{\mathbf{y}}_{\mathbf{k}})\|_1}{\sum_{n=1}^{N} \|\hat{\mathbf{y}}_{\mathbf{n},\mathbf{k}}\|_1}, \tag{18}$$

where $\hat{\mathbf{y}}_{\mathbf{n},\mathbf{k}}$ is the binary vector indicating selected frames from agent n at time step k, and $\mathbf{y}_{\mathbf{n},\mathbf{k}}$ is the ground truth binary vector of the view of agent n during the current period. $\bar{\mathbf{y}}_{\mathbf{k}}$ is the global ground truth binary vector of the scene at time step k, which is generated by

$$\bar{\mathbf{y}}_{\mathbf{k}} = min\left(\sum_{n=1}^{N} \mathbf{y}_{\mathbf{n},\mathbf{k}}, \mathbf{1}\right), \tag{19}$$

where the minimum is taken element-wise.

The first term in (18) gives higher rewards for the fast-forwarding action that selects the frames that match the ground truth better. As neighboring frames are often similar and share the same content, we hope to match the fast-forwarded result to the ground truth in a smoother fashion. That is, if the agent selects a frame that is close to the important frame, we will give some rewards, rather than no reward. To achieve this, we transfer the binary vector of selected frames for both the ground truth $\mathbf{y}_{\mathbf{n},\mathbf{k}}$ and the transmitted results $\hat{\mathbf{y}}_{\mathbf{n},\mathbf{k}}$ to a Gaussian distribution in a time window, denoted by the function $g(\cdot)$. The second term in (18) is used to reduce the redundancy in the fast-forwarded result. If the agents select more frames, the central controller will get a smaller reward for the current strategy selection.

Policy: The policy π decides the action to be executed at each time step by the system, i.e., it chooses the action for the system that maximizes the expected accumulated reward for the current step and the future as shown in (20). In other words, the policy

finds the fast-forwarding strategy of each agent that gives a larger expected accumulated reward.

$$\pi(s_k) = arg\max_a E[R|s_k, a, \pi] \tag{20}$$

Similar to the training of FFNet, We utilize Q-learning to achieve this policy by evaluating the value of $E[R|s, a, \pi]$ as $Q(s, a)$ and use a feed-forward neural network to approximate the Q-value.

VI. Experiments

In this section, we first present the experimental results of our MFFNet framework and its overall comparison with several single-agent fast-forwarding methods in the literature and FFNet, followed by its further comparison with FFNet in coverage-efficiency tradeoff and high-redundancy cases. We then compare MFFNet with our previous distributed multi-agent fast-forwarding framework DMVF in detail. Finally, we also evaluate how communication issues may affect MFFNet, an important practical consideration.

A. Datasets

We evaluate the performance of various methods on a multi-view video dataset VideoWeb [20] with fixed cameras and on a self-built simulated multi-view dataset on moving platforms using the CARLA simulator [60], referred to as CarlaSim.

VideoWeb: This dataset is captured in a realistic multi-camera network environment that involves multiple persons performing many different repetitive and non-repetitive activities. Same as in [19], we use the Day 4 subset of the VideoWeb dataset, which contains 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. The dataset includes the labels for important activities, based on which, we can generate a binary indicator for each frame to label its importance. That is, if a frame contains the labeled important activities, it will be labeled as an important frame with the binary indicator as 1 (otherwise, as 0). With such a frame importance indicator, we can generate a global ground truth across views for evaluation purpose.

CarlaSim: CARLA is a simulator for urban autonomous driving. It provides open digital assets (urban layouts, buildings, and vehicles) and supports flexible specifications of sensor suites, environmental conditions, full control of all static and dynamic actors, map generation, and more. For generating the CarlaSim dataset, we utilize the Town3 environment in the CARLA simulator, which has a 5-lane junction, a roundabout, unevenness, a tunnel, and so on. The multi-view videos are captured by putting multiple cameras on an autonomous car, which runs with a built-in autonomous driving controller. Detailed specifications for generating the video data are shown in Table III. We generate a binary indicator for each video frame according to the existence of vehicles in the view. If a frame captures a nearby vehicle (i.e., with size $>$ 150 pixels), it will be labeled as an important frame. The global ground truth for evaluation is generated by the same method as for the VideoWeb dataset. Fig. 5 shows some example frames in the CarlaSim dataset. From left to right, the columns stand for frames from front-left, front, and front-right views. As the data is collected on a moving platform, it captures more dynamic scenarios than the existing datasets that use stationary cameras (such as VideoWeb) and can help validate the efficacy of our methods in those dynamic scenarios.

TABLE III
Specifications in CARLA for Generating CarlaSim Dataset

Specification	Value
City	Town-3
Number of videos	18
Video length	10000
Resolution	720 x 480
Camera	front, front-left, front-right
Terrain	5-lane junction, roundabout, unevenness, tunnel
Weather	dynamic cloudiness, precipitation, sun angle

Fig. 5. Some illustrative example frames from the CarlaSim dataset. From left to right, the columns stand for frames from front-left, front, and front-right views. The CarlaSim dataset has multiple weather conditions, such as cloudy, rainy, and sunny (rows 1–3). Different terrains exist in the map, such as the tunnel in row 4.

B. Experimental Setup

Implementation Details: Our MFFNet is implemented using the TensorFlow library. The fast-forwarding agents are all modeled as 4-layer neural networks. ϵ-greedy strategy is used to better explore the state space during the training process. In the following experiments, we explore the scenarios of both 3 views ($N = 3$) and 6 views ($N = 6$), and set the similarity threshold to $\rho = 0.525$ and $\rho = 0.575$, respectively. The strategy computation threshold τ is set to 0.4. The strategy update period T is set to 100 frames of the raw video inputs. The 3 strategies used in our framework are FFNet and its variants as defined in

TABLE IV
OPERATING POINTS OF STRATEGIES ON VIDEOWEB

Strategy	Slow	Normal	Fast
Processing rate(%)	8.69	6.02	3.73
3-view Coverage(%)	66.22	52.88	48.38
6-view Coverage(%)	73.45	61.91	55.89

TABLE V
OPERATING POINTS OF STRATEGIES ON CARLASIM

Method	Slow	Normal	Fast
coverage(%)	80.78	67.83	60.78
Processing Rate(%)	18.06	14.76	7.09

Section III-B. The operating points of agents with the slow, normal and fast strategies are shown in Table IV for VideoWeb and Table V for CarlaSim.

Each video frame is represented by the penultimate layer (pool 5) of the GoogLeNet model [61] (1024-dimensions). Each baseline algorithm is evaluated with the same neighboring window extension as ours. We randomly use 80% of the videos for training and the remaining 20% for testing. We report the average performance on 5 rounds of experiments.

Evaluation Metrics: We evaluate the performance of methods with a *coverage* metric and a *processing rate* metric. The coverage metric evaluates how well the resulting fast-forwarding videos across multiple agents cover the important frames in the ground truth. It is computed as the percentage of the important frames that are included in the fast-forwarding videos across agents. In other words, if an important frame is included in any one of the agents' fast-forwarding videos, it will be considered as covered. The processing rate metric measures the percentage of the frames being processed by the system.

Comparison Methods: We compare our MFFNet with the following methods for video fast-forwarding and video summarization: (1) **Random**, which skips the incoming frames randomly. (2) **Uniform**, which fast-forwards the video uniformly. (3) **Online Kmeans (OK)** [62], a clustering-based method working in an online update fashion. The summary result consists of the frames that are the closest to the centroid in each cluster. (4) **Spectral Clustering (SC)** [63], a clustering-based method that provides several clusters from all the frames in a video. The summary is composed by the frames that are closest to each centroid. (5) **Sparse Modeling Representative Selection (SMRS)** [21], which takes the entire video as the dictionary and finds the representative frames based on the zero patterns of the sparse coding vector. (6) **FFNet** [18], the method we developed for single-agent video fast-forwarding. (7) **DMVF** [19], the distributed multi-agent fast-forwarding method we developed.

C. Comparison of MFFNet With Single-Agent Fast-Forwarding Approaches

Table VI shows the coverage metric and the processing rate of the single-agent fast-forwarding approaches in the literature, FFNet, and MFFNet, on the VideoWeb dataset for the 3-view and 6-view scenarios and the CarlaSim dataset. Note that in the cases of single-agent approaches (including FFNet), every view/camera uses the same approach and configuration. In contrast, a multi-agent approach like MFFNet coordinates the operations of multiple views. From the table, we can clearly see its advantage. More specifically:

- For those methods that require processing the entire video (processing rate of 100%), i.e., OK, SC and SMRS, our approach MFFNet achieves higher coverage (more than 25% increase) and much lower processing rate.
- When compared with Random and Uniform methods, MFFNet offers significant improvement in coverage with a modest increase in processing rate.
- When compared with FFNet, our state-of-the-art single-agent approach, MFFNet achieves a slightly better coverage while reducing the processing rate by 9.3% in VideoWeb 3-view, 7.3% in VideoWeb 6-view and 12.20% in CarlaSim. This shows that MFFNet is able to further reduce the computation load in the fast-forwarding process while offering the same (or higher) level of coverage of important frames.

D. Further Comparison of MFFNet With FFNet

Enabling Flexible Coverage-Efficiency Tradeoff: When deploying a video fast-forwarding strategy, the goal of achieving high efficiency (i.e., low processing rate) contradicts with the goal of maintaining high coverage, and the designers may want to trade off between the two metrics. To enable such tradeoff, MFFNet incorporates a tunable parameter, i.e., the similarity threshold ρ. Fig. 6 shows that by changing ρ, different levels of tradeoff between coverage and efficiency can be easily achieved on VideoWeb dataset for 3-view and 6-view scenarios and on CarlaSim. This is much more flexible and systematic than simply deploying FFNet on each agent and manually trying their skipping speeds.

Addressing High-redundancy Cases: The different views in the VideoWeb dataset have a modest level of redundancy across them. When the redundancy level is higher, the improvement of our MFFNet over FFNet will be even more significant. Here we consider the extreme case where each view has the same video data, i.e., the highest level of redundancy. The fast-forwarding performance of MFFNet and FFNet in both VideoWeb and CarlaSim is shown in Table VII. As CarlaSim only has 3 different camera views, thus no results are available for MFFNet-6v. Note that FFNet does not have any strategy changes in different settings, so its results for 3-view and 6-view are the same for the extreme case. From the result, we can see that MFFNet can achieve much higher coverage and lower processing rate than FFNet.

E. Comparison of MFFNet With Distributed Multi-Agent Framework DMVF

In this section, we compare MFFNet with our distributed multi-agent video fast-forwarding framework DMVF [19], on both VideoWeb 6-view and CarlaSim datasets. The results are shown in Table VIII. We have the following findings:

- MFFNet and DMVF are comparable in coverage and processing rates on VideoWeb and CarlaSim. On VideoWeb,

TABLE VI
COMPARISON OF MFFNET WITH SINGLE-AGENT FAST-FORWARDING APPROACHES FOR BOTH VIDEOWEB AND CARLASIM DATASETS

Methods	Random	Uniform	OK	SC	SMRS	FFNet	MFFNet
VideoWeb 3-view Coverage (%)	41.33	27.79	39.92	42.10	31.10	52.88	**53.66**
VideoWeb 3-view Processing rate (%)	4.40	4.00	100	100	100	6.02	**5.46**
VideoWeb 6-view Coverage (%)	50.78	25.80	50.21	44.74	42.36	61.91	**61.92**
VideoWeb 6-view Processing rate (%)	4.20	3.70	100	100	100	6.02	**5.58**
CarlaSim Coverage (%)	55.69	36.74	52.24	51.80	46.85	67.83	**68.65**
CarlaSim Processing rate(%)	6.50	5.40	100	100	100	14.76	**12.96**

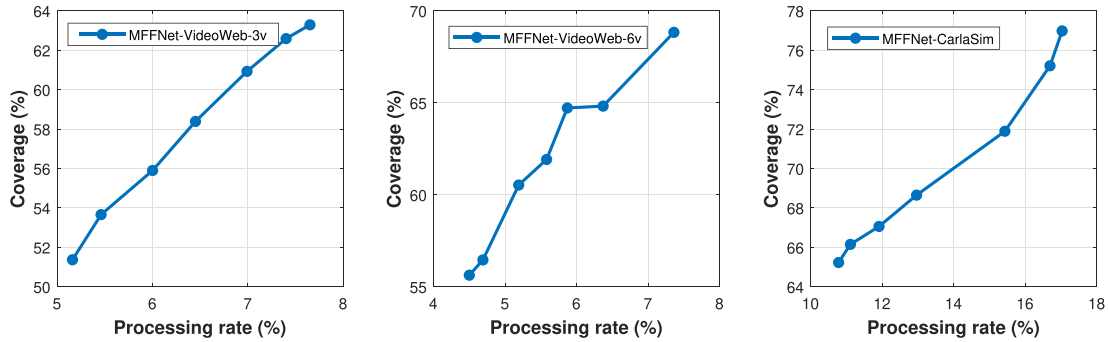


Fig. 6. Trade-off between coverage and processing rate in MFFNet for 3-view and 6-view scenarios of VideoWeb dataset. By tuning the similarity threshold ρ (marked in the figure), different levels of tradeoff can be achieved.

TABLE VII
COMPARISON OF MFFNET AND FFNET IN THE EXTREME CASE, WHERE ALL VIEWS HAVE THE SAME DATA

Methods	FFNet	MFFNet-3v	MFFNet-6v
VideoWeb Coverage(%)	54.10	71.93	75.61
VideoWeb Processing rate(%)	8.69	5.30	4.53
CarlaSim Coverage(%)	52.38	79.31	/
CarlaSim Processing rate(%)	14.76	8.09	/

DMVF achieves better coverage while MFFNet achieves better coverage on CarlaSim.

- On both datasets, MFFNet has less communication load (-44% for VideoWeb and -15% for CarlaSim) and higher frame rate (+34% on VideoWeb and +93% on CarlaSim). This is because that DMVF is a distributed method. The same information from one agent may need to be sent multiple times and the framework needs to reach a consensus on the strategy update, which leads to a higher communication load and longer communication delay.

While MFFNet has the advantages on less communication load and higher frame rate, DMVF is more flexible to utilize as it does not need a centralized infrastructure and the connections among agents can be adjusted according to system needs and agent capabilities. Both centralized and distributed methods could be suitable for improving the efficiency of a network of resource-limited agents with cameras, which can be used in tasks such as search and rescue, wide-area surveillance, and environment monitoring. Considering the advantages of each method, the choice between them depends on the practical application scenario. If we have a stable centralized infrastructure and each agent is able to reliably connect to the central controller, the centralized MFFNet might be a better choice as it can further reduce the communication load and improve the overall efficiency. However, in some cases (e.g., in an adversarial environment) we do not have a stable and capable centralized infrastructure, and some agents may not be able to reliably connect to the central node due to their physical distance or own resource limitations, in which case DMVF might be a better choice.

F. Impact of Communication on MFFNet

For a multi-agent strategy such as MFFNet, communication issues such as desynchronization or packet losses could have a major impact in practice, especially in the case of wireless communication (in [19], the impact of network connectivity on DMVF was studied). In this section, we evaluate the performance of MFFNet under the impact of such communication issues, using the VideoWeb dataset for illustration.

Desynchronization: In this experiment, we consider the desynchronization of one view with respect to the others. For instance, frame 20 from one view may be taken physically at the same time as frame 0 of other views, but is given a time tag that is the same as frame 20 of other views (this could happen due to the desynchronization of camera clocks). Fig. 7 shows the results on the 3-view scenario when one view is 20 or 100 frames desynchronized (either ahead or behind) with the other views. We can see that the desynchronization indeed has some effect on MFFNet coverage, but the drop is not too significant. Similar results can be observed for the 6-view scenario. In practice, with a decent clock synchronization scheme, we should be able to maintain the desynchronization to be under 20 frames.

Packet Losses: We consider the cases where a packet from an agent to the central controller may be lost due to communication disturbance. Each packet is the fast-forwarded segments from 100-frame raw videos at an agent. Table IX shows the coverage of MFFNet when the packet loss probability varies from 2.5% to 10%. We can see that the drop is not very significant. Moreover,

TABLE VIII
COMPARISON OF MFFNET WITH DISTRIBUTED MULTI-AGENT FAST-FORWARDING FRAMEWORK DMVF

Method-Dataset	DMVF-VideoWeb	MFFNet-VideoWeb	DMVF-CarlaSim	MFFNet-CarlaSim
coverage(%)	65.87	61.92	64.54	68.65
Processing Rate(%)	5.06	5.58	12.33	12.96
Communication p2p(GB)	0.18	/	0.20	/
Communication central (GB)	/	0.10	/	0.17
Total Communication (GB)	0.18	0.10	0.20	0.17
Summary to Server (GB)	3.59	3.22	2.27	2.46
FPS	313	419	119	230

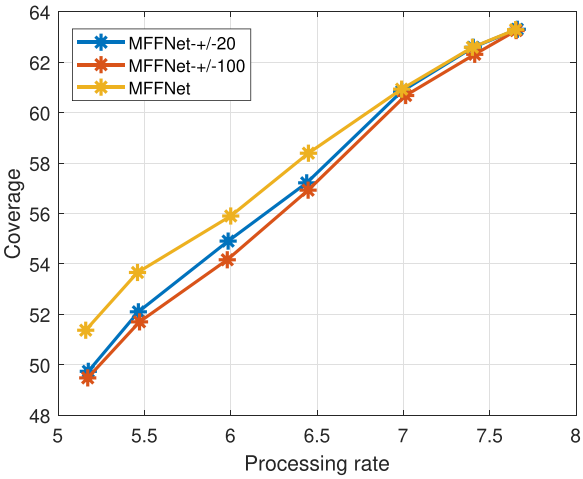


Fig. 7. Effect of desynchronization on MFFNet in 3-view VideoWeb scenario. The desynchronization has some effect on the coverage of MFFNet, but the drop is not too significant.

TABLE IX
EFFECT OF PACKET LOSSES ON MFFNET IN 3-VIEW AND 6-VIEW SCENARIOS IN VIDEOWEB. THE PERFORMANCE IS SLIGHTLY AFFECTED BY THE PACKET LOSS (LESS THAN 10% DEGRADATION IN 10% LOSS PROBABILITY)

Loss probability	2.5%	5.0%	7.5%	10.0%
3-view coverage(%)	52.00	50.98	49.43	49.00
6-view coverage(%)	60.43	60.08	59.90	57.89

TABLE X
COMPARISON OF DIFFERENT CONTROLLERS IN MFFNET

Method	MFFNet	MFFNet-DQN-0	MFFNet-DQN-1
Coverage(%)	53.66	64.80	64.24
Processing Rate(%)	5.46	7.32	7.07

most of the coverage drop is due to the loss of data itself rather than the strategy selection process.

G. Study of Central Controller Designs in MFFNet

The above results of MFFNet are based on the heuristic central controller design introduced in Section V-B, with explicit similarity computation and strategy computation. In this section, we compare such central controller design with the RL-based design introduced in Section V-C, using the VideoWeb 3-view case as an example. The results are shown in Table X, where MFFNet is the heuristic central controller based on similarity computation, and MFFNet-DQN-0 and MFFNet-DQN-1 represent two RL-based central controllers using DQN models with the trade-off factor α in (18) set to 0 and 1, respectively. From the table, we have the following observations: 1) The RL-based central controllers using DQN have higher coverage than the heuristic one based on similarity computation but also have much higher processing rate. 2) Setting the trade-off term α in the immediate reward to 1 can help lower the processing rate but also degrade the coverage. Note that the choice of which central controller to use depends on the trade-off preference on coverage or processing rate.

H. Deployment of MFFNet on Embedded Platform

We deployed MFFNet on an actual embedded platform to evaluate its efficiency. The central controller is implemented on a Dell Precision 5820 Tower workstation with a 3.6 GHz Xeon W-2123 CPU and 16 GB memory, and the agents are run on Nvidia Jetson TX2. The communication between the central controller and the agents is implemented with a wireless network using TCP. For MFFNet, the average frame rate is 661 FPS for the 3-view scenario and 419 FPS for the 6-view scenario (note that only a fraction of these frames will be actually processed), showing its capability to work efficiently and effectively with real-time speed on embedded processors.

VII. CONCLUSION

In this paper, we first summarize our previous work on the single-agent video fast-forwarding method FFNet and distributed multi-agent video fast-forwarding framework DMVF, and then present a new centralized multi-agent fast-forwarding framework MFFNet. The MFFNet framework includes a set of multi-strategy fast-forwarding agents that can adapt to different fast-forwarding paces, and a central controller that can choose the proper pace for every agent and generate a compact summary of the scene. We conducted a series of experiments on a real-world surveillance video dataset and a new simulated driving dataset, for MFFNet, DMVF, FFNet, and several methods in the literature. Experimental results demonstrate that our two collaborative multi-agent video fast-forwarding approaches, MFFNet and DMVF, can achieve better scene coverage and lower frame processing rate than applying single-agent fast-forwarding approaches on multiple agents without coordination. The experiments also demonstrate the trade-off between MFFNet and DMVF, the impact of communication disturbance, and the choice of different central controller designs.

REFERENCES

[1] E. Elhamifar and M. C. D. P. Kaluza, "Online summarization via submodular and convex optimization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1818–1826.

[2] M. Gygli, H. Grabner, and L. Van Gool, "Video summarization by learning submodular mixtures of objectives," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3090–3098.

[3] R. Panda, A. Das, Z. Wu, J. Ernst, and A. K. Roy-Chowdhury, "Weakly supervised summarization of web videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3677–3686.

[4] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *Proc. 14th Eur. Conf. Comput. Vis.*, 2016, pp. 766–782.

[5] B. Zhao and E. P. Xing, "Quasi real-time summarization for consumer videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2513–2520.

[6] Y. Fu et al., "Multi-view video summarization," *IEEE Trans. Multimedia*, vol. 12, pp. 717–729, 2010.

[7] R. Panda, A. Dasy, and A. K. Roy-Chowdhury, "Video summarization in a multi-view camera network," in *Proc. IEEE 23rd Int. Conf. Pattern Recognit.*, 2016, pp. 2971–2976.

[8] R. Panda and A. K. Roy-Chowdhury, "Multi-view surveillance video summarization via joint embedding and sparse optimization," *IEEE Trans. Multimedia*, vol. 19, pp. 2010–2021, 2017.

[9] M. Elfeki, L. Wang, and A. Borji, "Multi-stream dynamic video summarization," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, Jan. 2022, pp. 339–349.

[10] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien, "Online multi-view video summarization for wireless video sensor network," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 1, pp. 165–179, Feb. 2015.

[11] K.-Y. Cheng, S.-J. Luo, B.-Y. Chen, and H.-H. Chu, "Smartplayer: User-centric video fast-forwarding," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2009, pp. 789–798.

[12] T. Halperin, Y. Poleg, C. Arora, and S. Peleg, "EgoSampling: Wide view hyperlapse from egocentric videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 5, pp. 1248–1259, May 2018.

[13] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen, "Real-time hyperlapse creation via optimal frame selection," *ACM Trans. Graph.*, vol. 34, no. 4, 2015, Art. no. 63.

[14] N. Petrovic, N. Jojic, and T. S. Huang, "Adaptive video fast forward," *Multimedia Tools Appl.*, vol. 26, no. 3, pp. 327–344, 2005.

[15] Y. Poleg, T. Halperin, C. Arora, and S. Peleg, "EgoSampling: Fast-forward and stereo for egocentric videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4768–4776.

[16] W. L. S. Ramos, M. M. Silva, M. F. M. Campos, and E. R. Nascimento, "Fast-forward video based on semantic extraction," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 3334–3338.

[17] M. M. Silva, W. L. S. Ramos, J. P. K. Ferreira, M. F. M. Campos, and E. R. Nascimento, "Towards semantic fast-forward and stabilized egocentric videos," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 557–571.

[18] S. Lan, R. Panda, Q. Zhu, and A. K. Roy-Chowdhury, "Ffnet: Video fast-forwarding via reinforcement learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6771–6780.

[19] S. Lan, Z. Wang, A. K. Roy-Chowdhury, E. Wei, and Q. Zhu, "Distributed multi-agent video fast-forwarding," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 1075–1084.

[20] G. Denina et al., "Videoweb dataset for multi-camera activities and non-verbal communication," in *Distributed Video Sensor Networks*. Berlin, Germany: Springer, 2011, pp. 335–347.

[21] E. Elhamifar, G. Sapiro, and R. Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1600–1607.

[22] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 505–520.

[23] G. Guan et al., "A top-down approach for video summarization," *ACM Trans. Multimedia Comput. Commun., Appl.*, vol. 11, no. 1, 2014, Art. no. 4.

[24] E. Elhamifar and Z. Naing, "Unsupervised procedure learning via joint dynamic summarization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6341–6350.

[25] B. Gong, W. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2069–2077.

[26] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, "AdaFrame: Adaptive frame selection for fast video recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1278–1287.

[27] M. Rochan and Y. Wang, "Video summarization by learning from unpaired data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7902–7911.

[28] A. Khosla, R. Hamid, C.-J. Lin, and N. Sundaresan, "Large-scale video summarization using web-image priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 2698–2705.

[29] G. Kim, L. Sigal, and E. P. Xing, "Joint summarization of large-scale collections of web images and videos for storyline reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 4225–4232.

[30] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, "TVSum: Summarizing web videos using titles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5179–5187.

[31] R. Panda and A. K. Roy-Chowdhury, "Collaborative summarization of topic-related videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7083–7092.

[32] G. A. Sigurdsson, X. Chen, and A. Gupta, "Learning visual storylines with skipping recurrent neural networks," in *Proc. 14th Eur. Conf. Comput. Vis.*, 2016, pp. 71–88.

[33] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien, "Low complexity on-line video summarization with gaussian mixture model based clustering," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 1260–1264.

[34] K. A. Peker et al., "An extended framework for adaptive playback-based video summarization," *Proc. SPIE*, vol. 5242, pp. 26–33, 2003.

[35] K. A. Peker, A. Divakaran, and H. Sun, "Constant pace skimming and temporal sub-sampling of video using motion activity," in *Proc. IEEE Int. Conf. Image Process.*, 2001, pp. 414–417.

[36] J. Jiang and X.-P. Zhang, "A new player-enabled rapid video navigation method using temporal quantization and repeated weighted boosting search," in *Proc. IEEE Comput. Vis. Pattern Recognit. Workshops*, 2010, pp. 64–71.

[37] J. Jiang and X.-P. Zhang, "A smart video player with content-based fast-forward playback," in *Proc. 19th ACM Int. Conf. Multimedia*, 2011, pp. 1061–1064.

[38] W. Ramos, M. Silva, E. Araujo, L. S. Marcolino, and E. Nascimento, "Straight to the point: Fast-forwarding videos via reinforcement learning using textual data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10931–10940.

[39] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2678–2687.

[40] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2894–2902.

[41] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li, "Deep reinforcement learning-based image captioning with embedding reward," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 290–298.

[42] A. Krull et al., "Poseagent: Budget-constrained 6D object pose estimation via reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6702–6710.

[43] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Young Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2711–2720.

[44] Y. Zhang, M. Kampffmeyer, X. Zhao, and M. Tan, "Deep reinforcement learning for query-conditioned video summarization," *Appl. Sci.*, vol. 9, no. 4, 2019, Art. no. 750.

[45] L. Bu, R. Babu, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Part C (Appl. Rev.)*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[46] X. Kong, B. Xin, Y. Wang, and G. Hua, "Collaborative deep reinforcement learning for joint object search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1695–1704.

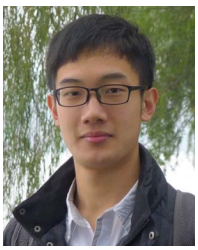
[47] L. Ren, J. Lu, Z. Wang, Q. Tian, and J. Zhou, "Collaborative deep reinforcement learning for multi-object tracking," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 586–602.

[48] W. Wu, D. He, X. Tan, S. Chen, and S. Wen, "Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6222–6231.

[49] S. Sukhbaatar et al., "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2244–2252.

[50] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.
[51] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Dept. Elect. Eng. Comput., Massachusetts Inst. Technol., Cambridge, MA, USA, 1984.
[52] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[53] A. Nedić, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.
[54] I. Matei and J. S. Baras, "Performance evaluation of the consensus-based distributed subgradient method under random communication topologies," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 754–771, Aug. 2011.
[55] A. Nedić, "Asynchronous broadcast-based convex optimization over a network," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1337–1351, Jun. 2011.
[56] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.
[57] I. Lifshitz, E. Fetaya, and S. Ullman, "Human pose estimation using deep consensus voting," in *Proc. 14th Eur. Conf. Comput. Vis.*,2016, pp. 246–260.
[58] H. Wang and D. Suter, "Background subtraction based on a robust consensus method," in *Proc. IEEE 18th Int. Conf. Pattern Recognit.*, 2006, pp. 223–226.
[59] A. T. Kamal, J. H. Bappy, J. A. Farrell, and A. K. Roy-Chowdhury, "Distributed multi-target tracking and data association in vision networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1397–1410, Jul. 2016.
[60] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.
[61] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[62] D. Arthur and S. Vassilvitskii, "K-means : The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.
[63] U. V. Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

Shuyue Lan received the bachelor's degree in automation from the University of Science and Technology of China, Hefei, China, in 2015, and the Ph.D. degree in computer engineering from Northwestern University, Evanston, IL, USA, in 2021. She spent her first two years of Ph.D. degree with UC Riverside, Riverside, CA, USA. Her research interest includes computer vision, machine learning, and cyber-physical systems, and is focusing on high-performance deep learning inference workflow.



Zhilu Wang received the B.S. degree in applied physics from the University of Science and Technology of China, Hefei, China, in 2016, and the Ph.D. degree in computer engineering from Northwestern University, Evanston, IL, USA, in 2022. He began his doctoral career with the University of California, Riverside, CA, USA, in 2016. His research interest includes formal verification, machine learning, real-time systems, and cyber-physical systems. He was the recipient of the Best Paper Award with the 2022 ACM/IEEE DATE conference and Best Thesis Award in Computer Engineering from Northwestern University.

Ermin Wei (Member, IEEE) received the undergraduate triple degree in computer engineering, finance and mathematics (with a minor in German) from the University of Maryland, College Park, MD, USA, the M.S. degree from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, and the Ph.D. degree in electrical engineering and computer Science from MIT, in 2014, advised by Professor Asu Ozdaglar. She is currently an Assistant Professor with the Electrical and Computer Engineering Department and Industrial Engineering and Management Sciences Department of Northwestern University, Evanston, IL, USA. Her research interests include distributed optimization methods, convex optimization and analysis, smart grid, communication systems and energy networks, and market economic analysis. She was the recipient of many awards, including the Graduate Women of Excellence Award, second place prize in Ernst A. Guillemen Thesis Award and Alpha Lambda Delta National Academic Honor Society Betty Jo Budson Fellowship, her team also won the 2nd place in the Grid Optimization (GO) competition 2019, an electricity grid optimization competition organized by Department of Energy.

Amit K. Roy-Chowdhury (Fellow, IEEE) received the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2002. In 2004, he joined the University of California, Riverside (UCR), CA, USA, where he is currently a Professor and Bourns Family Faculty Fellow of electrical and computer engineering, Director with the Center for Robotics and Intelligent Systems, and Cooperating Faculty with the Department of Computer Science and Engineering. He leads the Video Computing Group with UCR, working on foundational principles of computer vision, image processing, and statistical learning, with applications in cyber-physical, autonomous and intelligent systems. He has authored or coauthored more than 200 papers in peer-reviewed journals and conferences, two monographs: *Camera Networks: The Acquisition and Analysis of Videos Over Wide Areas* and *Person Re-identification with Limited Supervision*. He is on the Editorial Boards of major journals and program committees of the main conferences in his area. He was the recipient of the Doctoral Dissertation Advising/Mentoring Award 2019 from UCR and the ECE Distinguished Alumni Award from UMCP, and his students have been first authors on multiple papers that was recipient og the Best Paper Awards at major international conferences. He is also the Fellow of the IAPR.

Qi Zhu (Member, IEEE) received the B.E. degree in computer science from Tsinghua University, Beijing, China, in 2003, and the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 2008. He is currently an Associate Professor with ECE Department, Northwestern University, Evanston, IL, USA. His research interests include design automation for cyber-physical systems (CPS) and Internet of Things (IoT), safe and secure machine learning for CPS and IoT, cyber-physical security, and system-on-chip design, with applications in domains such as connected and autonomous vehicles, energy-efficient smart buildings, and robotic systems. He was the recipient of the NSF CAREER Award, IEEE TCCPS Early-Career Award, Humboldt Research Fellowship for Experienced Researchers, and the Best Paper awards at DAC 2006, DAC 2007, ICCPS 2013, ACM TODAES 2016, and DATE 2022. He is also the Conference Chair of IEEE TCCPS, and Young Professionals Coordinator with IEEE Council on Electronic Design Automation (CEDA). He is an Associate Editor for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, *ACM TCPS*, and *IET Cyber-Physical Systems: Theory & Applications*, and was a Guest Editor of the Proceedings of the IEEE, ACM TCPS, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, Elsevier*JSA*, Elsevier *Integration*, and *VLSI Journal*.