Reinforcement Learning-Driven Edge Management for Reliable Multi-view 3D Reconstruction

Motahare Mounesan, Sourya Saha, Houchao Gan, Md. Nurul Absur, Saptarshi Debroy
City University of New York
Emails: {*mmounesan,ssaha2,hgan,mabsur*}*@gradcenter.cuny.edu, saptarshi.debroy@hunter.cuny.edu*

***Abstract*—Real-time multi-view 3D reconstruction is a mission-critical application for key edge-native use cases, such as fire rescue, where timely and accurate 3D scene modeling enables situational awareness and informed decision-making. However, the dynamic and unpredictable nature of edge resource availability introduces disruptions, such as degraded image quality, unstable network links, and fluctuating server loads, which challenge the reliability of the reconstruction pipeline. In this work, we present a reinforcement learning (RL)-based edge resource management framework for reliable 3D reconstruction to ensure high quality reconstruction within a reasonable amount of time, despite the system operating under a resource-constrained and disruption-prone environment. In particular, the framework adopts two cooperative Q-learning agents, one for camera selection and one for server selection, both of which operate entirely online, learning policies through interactions with the edge environment. To support learning under realistic constraints and evaluate system performance, we implement a distributed testbed comprising lab-hosted end devices and FABRIC infrastructure-hosted edge servers to emulate smart city edge infrastructure under realistic disruption scenarios. Results show that the proposed framework improves application reliability by effectively balancing end-to-end latency and reconstruction quality in dynamic environments.**

***Index Terms*—Reinforcement Learning, Multi-view 3D Reconstruction, Edge Computing, Reliability.**

I. Introduction

Multi-view 3D reconstruction of unknown scenes is becoming a foundational capability for mission-critical smart city applications, such as emergency response and public safety, providing situational awareness for rapid, informed decision making [1], [2]. This technique aggregates images from multiple viewpoints to generate 3D representations, typically as point clouds or meshes. Reconstructing 3D scenes from overlapping views requires substantial computational resources. While data centers can easily handle such workloads, smart city deployments rely on city-scale edge servers for lower latency, cost efficiency, and data privacy. While these edge servers are geographically closer and suitable for 'near real-time' processing, they lack the capacity of remote data centers. Delivering timely results thus necessitates intelligent resource management to balance conflicting performance requirements.

The already tricky proposition of edge management for multi-view 3D reconstruction becomes even more challenging in mission-critical scenarios due to its susceptibility to disruptions and fluctuations. This comes from three factors: i) the nature of the reconstruction process, ii) the volatile environments of the underlying use cases, and iii) the resource constraints of smart city edge ecosystems. Unlike simpler video processing applications that operate on independent frame streams [3], [4], [5], 3D reconstruction requires high-quality, spatially and temporally aligned images from multiple viewpoints. When such data is degraded or partially missing, the reconstruction results can be compromised both quantitatively (e.g., incomplete surfaces and missing objects) and qualitatively (e.g., geometric distortions). This vulnerability is exacerbated in dynamic, high-risk settings, such as fire scenes, disaster zones, or urban deployments, where sensing and communication infrastructure is prone to disruption. Further, the inherently volatile nature of smart city infrastructure can also cause disruptions, including unstable network condition, limited edge connectivity, processing delays, and resource contention. Although these issues mainly affect end-to-end processing latency, degraded or unavailable input images cannot be retransmitted or recovered within the tight latency constraints of rapidly evolving environments.

Reducing end-to-end latency of multi-view 3D reconstruction pipelines [6], [7], [8] without significantly degrading reconstruction quality is thus a challenging task, especially in smart city edge deployments where network conditions and camera availability fluctuate, timing constraints are strict, and the true system state is only partially observable [2], [9]. Achieving consistent performance across these varying conditions requires a *reliable* decision-making process, one that adapts to disruptions while ensuring reconstruction results remain timely and accurate. Traditional heuristic-based approaches for edge management rely primarily on static environment and prior knowledge of system parameters — assumptions that are unrealistic in such dynamic environments. Consequently, their static design prevents them from ensuring reliable performance. Reinforcement learning (RL)-based decision-making, which enables the system to learn effective policies *online* without full knowledge of the environment in advance, can offer robust and adaptive solutions for managing the latency–quality trade-off in mission-critical applications.

In this paper, we focus on two critical decisions: selecting camera subsets, affected by visual or network disruptions, and choosing edge servers, influenced by computational load and network conditions. We develop an RL-based framework that enables the system to make these decisions adaptively. We

This material is based upon work supported by the National Science Foundation (NSF) under Award Number CNS-1943338

978-3-903176-75-1 ©2025 IFIP

employ two cooperative Q-learning agents, one for camera selection and one for server selection, learning adaptive policies that improve responsiveness and reliability under unpredictable disruptions. The agents learn entirely online during system operation, thus requiring practical implementation to expose the agents to representative system dynamics. Thus, we implement and evaluate our framework on a realistic testbed composed of resource-constrained end devices in the lab, along with FABRIC network and server infrastructure that emulates network and edge-server components, operating under diverse simulated disruption scenarios. Results demonstrate that the camera selection strategy achieves up to 15% higher reliability, while the adaptive server selection strategy achieves up to a 50% improvement over baselines. Overall, these findings highlight the potential of RL to enable robust, disruption-aware decision making in real-time edge systems.

The remainder of this paper is organized as follows. Section II reviews the background and related work. Section III describes the system model and details the proposed reinforcement learning–based framework for camera and server selection. Section IV presents our testbed setup, evaluation methodology, and experimental results under various disruption scenarios. Finally, Section V summarizes the key findings.

II. Background and related works

Here, we present an overview of multi-view 3D reconstruction and discuss related works.

A. Multi-view 3D reconstruction

Multi-view 3D reconstruction generates 3D representations of objects or scenes from RGB images or occasionally sketches. It uses images captured from multiple perspectives, either as sequential frames or sparse views. Outputs include voxel grids, point clouds, and meshes [10], with point clouds and meshes particularly versatile, scalable, and accurate. Geometry-based reconstruction follows a two-step pipeline: Structure-from-Motion (SfM) and Multi-View Stereo (MVS). OpenMVG/openMVS [6], [7] is a popular library implementing this pipeline. Deep learning methods estimate object geometry using prior knowledge [11], [12], but often yield lower accuracy and generalization than geometry-based approaches. *In this work, we adopt the geometry-based pipeline for its robustness and reconstruction quality.*

B. Evaluating reconstruction quality

A key objective of latency-optimized edge management for 3D reconstruction is ensuring high-quality outputs, which requires reliable techniques to evaluate generated point clouds. Quality can be assessed via three main methods: Geometry-Oriented [13], Color-Oriented [14], and Combined [15], all of which rely on *ground-truth* point clouds. However, ground truth for unknown 3D scenes in a smart city environment is typically unavailable. In its absence, point cloud quality must be evaluated using no-reference methods, including traditional [16] and learning-based approaches [17]. Traditional methods assess internal geometric consistency or appearance cues derived from projections, while learning-based methods predict perceptual metrics such as the Mean Opinion Score (MOS) using neural networks trained on annotated datasets. *Since creating and managing large annotated datasets for city-scale indoor/outdoor scenes is impractical, we employ a latency-efficient geometry-based technique leveraging known intrinsic and extrinsic camera parameters (focal length, distortion, pose) to produce perspective-aligned images that preserve geometric structure and visual appearance. This generates high-fidelity projections that enable robust quality assessments even without ground truth.*

C. Reinforcement Learning for edge resource management

Edge resource management for complex, time-sensitive applications such as 3D reconstruction is challenging. Reinforcement learning has emerged as a promising approach for sequential decision-making in dynamic edge environments [18], [19], [20]. RL has been applied to optimize resource allocation for 3D pose reconstruction in robotic minimally invasive surgery [21], manage edge server orchestration for augmented reality workloads [22], and support adaptive wireless access in emergency response [23]. DRL-based frameworks like DRJOA [24] jointly optimize task offloading and wireless resources in MEC scenarios. *However, many RL-based methods rely on offline training or assume relatively static environments. Real-time, disruption-prone, resource-constrained settings like ours require online learning and rapid adaptation. Our work addresses this gap.*

D. Managing disruptions in distributed systems

Disruption management in distributed systems is becoming a critical area of research aimed at improving system reliability and security. Recent studies explore various aspects of this challenge. For example, the ReSeT system [25] focuses on reducing service disruption by optimizing handover times for mobile nodes. Work, such as [26] addresses latency in real-world applications by proposing a two-phase scheduling strategy to mitigate disruptions in edge environments. Additionally, [27] explores spatio-temporal disruption patterns that can impact video processing quality and applies a portfolio theory-inspired approach to model these disruptions. Video processing in edge systems is exposed to different types of disruptions beyond general system reliability. These include environmental and infrastructure constraints such as obstructions, occlusions [28], and mission-critical disruptions [29] that directly affect image quality and coverage. *Unlike these, in our work, we aim to explore a combination of factors, including environmental and infrastructure constraints, alongside coverage, accuracy, and computational complexity. We propose a resource selection strategy that is specifically tailored to address the unique challenges of complex 3D video processing tasks to ensure high-quality reconstruction.*

III. Problem Formulation and Solution Strategy

Here, we discuss the proposed system model, RL based problem formulation, and Q-learning based solution strategy.

A. System model

We consider a smart city environment running a 3D reconstruction application using city-hosted edge servers. The key components are: a set of edge servers for computation, a fleet of camera-equipped end devices (e.g., drones, robots) for scene capture, and a central edge controller or resource manager for overall resource allocation.

Edge servers ($\mathcal{E}$) – We consider a set of m smart city edge servers, each responsible for performing computationally intensive 3D reconstruction tasks. These servers operate under dynamically changing workloads due to processing demands from a variety of smart city applications running simultaneously. The available computational capacity and processing latency of each server fluctuate over time, and are not directly controlled by the controller.

Camera-equipped end devices ($\mathcal{C}$) – We consider a fleet of n end devices (e.g., drones, robots), each equipped with a high-resolution camera of identical hardware specifications but capturing the scene from different viewpoints. These devices continuously stream visual data to the edge system. However, environmental factors such as smoke, fog, or fire may degrade the quality of captured images, while wireless communication disruptions can prevent timely delivery of images to the controller. Both types of disruptions affect the availability and usefulness of each camera's data during processing runtime.

Controller – We assume a centralized decision-making module operating at the network edge responsible for overall resource management of the entire smart city environment. In the context of the 3D reconstruction use case which serves as the problem at hand, the controller's objective is to make latency- and quality-aware decisions, adapting to disruptions in sensing, communication, and computation. In this context, we reiterate that such systems operate in a disruption-prone environment, where among issues, unpredictable fluctuations in sensing quality, communication reliability, and computation capacity impact the *end-to-end latency and reconstruction quality balance* the most.

While the system model can accommodate multiple resource allocations (e.g., network paths, wireless channels), this work focuses on camera selection and server assignment under dynamic conditions. The primary disruptions either block data from a camera, reducing reconstruction quality, or overload a server, increasing latency. We study the controller's ability to adaptively select cameras and servers at each timestep in response to the evolving system state.

B. Defining 3D reconstruction reliability

A 3D reconstruction application is considered reliable if it consistently produces usable reconstructions within a predefined near real-time latency budget under dynamic conditions. Formally, we define the reliability ($\mathbb{R}_t$) at timestep t as a binary variable: TRUE if both quality and latency constraints are satisfied, and FALSE otherwise. In other words, a camera and server selection is reliable if it meets the minimum quality threshold and maximum latency constraint. Thus:

$$\mathbb{R}_t = \begin{cases} 1, & \text{if } Q_t \geq \Theta \text{ and } L_t \leq \Phi \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where:

- Q_t denotes the point cloud quality score at timestep t,
- L_t denotes the total end-to-end latency which includes transmission and processing (i.e., reconstruction) delays,
- Θ is the minimum acceptable reconstruction quality,
- Φ is the maximum allowable latency.

Long-term reliability is the proportion of timesteps with $\mathbb{R}_t = 1$, directly reflecting task success under predefined quality and latency constraints and supporting real-time evaluation.

C. RL-based problem formulation

As reliability is defined as satisfaction of two constraints: (i) reconstruction quality must remain above a minimum threshold, and (ii) reconstruction latency must not exceed a predefined upper bound, the problem is formulated as a constrained optimization task to select camera subsets and edge servers that maximize reliability under operational constraints.

We model the camera and server selection problem as a Markov Decision Process (MDP) to enable RL-based optimization under uncertain and variable environmental conditions. While some system components are partially observable, the MDP framework allows decision-making as a sequence of actions guided by measurable outcomes such as reconstruction quality and latency. This supports policies that adapt over time and generalize across deployment scenarios where handcrafted heuristics fail and prior learning is infeasible. To manage complexity and enable efficient online learning, we decompose the process into two cooperative agents: one for camera selection and one for server selection. *This separation is motivated by the large combinatorial space of camera selection, which complicates online adaptation if coupled with server decisions. Server selection depends on broader system context, including chosen cameras and projected resource load. Decoupling these roles allows each agent to optimize its subproblem independently with tailored rewards.*

Camera selection – The camera selection task is formulated as a *stateless Markov Decision Process (MDP)*. To support real-time inference and reduce model complexity, the agent operates without an explicit state representation. It selects actions based solely on its learned policy, without access to dynamic environment variables such as camera availability, quality degradation, or network conditions. As a result, the environment is treated as stateless from the agent's perspective, and learning is driven purely by interaction and feedback.

At each timestep t, the agent selects a subset of cameras from a pool of N candidates, based on the current set of valid combinations provided by the environment. The action space $\mathcal{A}_\mathcal{C}$ is defined as:

$$\mathcal{A}_\mathcal{C} = \left\{ \mathbf{a} \in \{0,1\}^N \mid k_{\min} \leq \|\mathbf{a}\|_1 \leq k_{\max} \right\} \tag{2}$$

where each binary vector $\mathbf{a}$ encodes which cameras are selected (1) or not (0), and the ℓ_1 norm constrains the number of selected cameras within predefined bounds.

The reward received by the camera agent at each timestep t reflects the system's reliability objective and is shaped to encourage selections that meet quality and latency constraints. Specifically, the reward $\mathcal{R}_t^{\mathcal{C}}$ is computed as:

$$\mathcal{R}_t^{\mathcal{C}} = w_1 \cdot S_Q^{(t)} + w_2 \cdot S_L^{(t)} \tag{3}$$

where $S_Q^{(t)}$ and $S_L^{(t)}$ represent the normalized reconstruction quality and latency scores at time t, and $w_1, w_2 \in [0, 1]$ are weighting coefficients summing one. Latency accounts solely for reconstruction delay, ignoring transmission overhead.

The quality score $S_Q^{(t)}$ is defined by normalizing the point cloud quality (Q) at timestep t with respect to a reference threshold Θ:

$$S_Q^{(t)} = \min\left(1, \frac{Q_t}{\Theta}\right) \tag{4}$$

The latency score $S_L^{(t)}$ captures how closely the reconstruction latency approaches the system's real-time threshold Φ, and is given by:

$$S_L^{(t)} = \max\left(0, 1 - \frac{L_t}{\Phi}\right) \tag{5}$$

This reward formulation enables the agent to receive partial credit for decisions that approach but do not fully satisfy the reliability constraints, thereby supporting progressive learning. The stateless design ensures lightweight, fast decision-making compatible with real-time deployment, while still promoting constraint-aware behavior in dynamic and partially observable environments.

Server selection – The server selection task is formulated as an MDP. At each timestep t, after the camera subset is selected, the server agent receives a state s_t defined as:

$$s_t = \left(\|\mathbf{a}_t\|_1, a_{t-1}^{\mathcal{E}}\right) \tag{6}$$

where:

- $\|\mathbf{a}_t\|_1$ denotes the number of cameras selected at time t,
- $a_{t-1}^{\mathcal{E}}$ is the server selected at the previous timestep.

The act of including the number of selected cameras in the server agent's state allows the agent to condition its server choice on the expected computational load associated with the camera selection decision. This coordination between the camera and server selection agents enables more effective adaptation to the dynamic conditions of the system.

The agent then selects an action $a_t^{\mathcal{E}} \in \mathcal{A}_{\mathcal{E}}$, where $\mathcal{A}_{\mathcal{E}}$ denotes the set of available servers. The reward for the server agent at timestep t, denoted $\mathcal{R}_t^{\mathcal{E}}$, is based solely on the end-to-end latency of the 3D reconstruction process, computed as:

$$\mathcal{R}_t^{\mathcal{E}} = S_L^{(t)} \tag{7}$$

where the latency score $S_L^{(t)}$ is defined as:

$$S_L^{(t)} = \max\left(0, 1 - \frac{L_t}{\Phi}\right) \tag{8}$$

This formulation enables the server agent to learn dynamic server assignment strategies that minimize end-to-end latency under varying server loads and network disruptions.

D. Q-learning based solution methodology

To solve the decision-making problems formulated as MDPs, we employ *tabular Q-learning* for both the camera selection and server selection agents.Q-learning is a model-free reinforcement learning algorithm that estimates an action-value function $Q(s, a)$, which represents the expected return of taking action a in state s and following the current policy thereafter. At each timestep, the agent observes the current state s_t, selects an action a_t, receives a reward r_t, observes the next state s_{t+1}, and updates the Q-value as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \Big[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \Big] \tag{9}$$

This temporal-difference update rule allows the agent to iteratively refine its value estimates without requiring a model of the environment. Q-learning is particularly well-suited to our setting due to the discrete action spaces and its ability to handle delayed rewards, which are common in scenarios where the consequences of decisions (e.g., reconstruction quality or end-to-end latency) may manifest several steps later.

Both agents employ an ϵ-greedy strategy for exploration: at each timestep, the agent selects a random action with probability ϵ_t and otherwise selects the action that maximizes the current Q-value. This balances exploration of new actions with exploitation of known good actions:

$$a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon_t \\ \arg\max_a Q(s_t, a) & \text{with probability } 1 - \epsilon_t \end{cases} \tag{10}$$

To improve adaptability under non-stationary conditions, we extend Q-learning with adaptive adjustment of both the learning rate α_t and the exploration rate ϵ_t, resulting in *Adaptive Q-learning*. When performance degradation is detected , both parameters are temporarily increased to encourage faster adaptation:

$$\epsilon_{t+1} = \min(\epsilon_t \cdot \eta_{\text{inc}},\ \epsilon_{\max}), \quad \alpha_{t+1} = \min(\alpha_t \cdot \lambda_{\text{inc}},\ \alpha_{\max}) \tag{11}$$

During steady-state operation, both are gradually decayed to promote convergence:

$$\epsilon_{t+1} = \max(\epsilon_t \cdot \eta_{\text{dec}},\ \epsilon_{\min}), \quad \alpha_{t+1} = \max(\alpha_t \cdot \lambda_{\text{dec}},\ \alpha_{\min}) \tag{12}$$

Here, $\eta_{\text{inc}} > 1$, $\eta_{\text{dec}} < 1$, $\lambda_{\text{inc}} > 1$, and $\lambda_{\text{dec}} < 1$ are adaptation factors that control the degree of responsiveness and stabilization. This mechanism enables the agent to remain flexible in the face of environmental volatility while converging efficiently when conditions are stable. In this work, we adopt standard Q-learning based camera selection strategy and adaptive Q-learning based server selection approach.

IV. Evaluation and Results

In this section, we evaluate our proposed ad-hoc camera selection and resource management framework through testbed experiments conducted in a realistic Edge-AI environment. We begin by describing the testbed setup, including system components and network design, followed by evaluation metrics, baselines, datasets, and implementation details. We then present and analyze the experimental results.

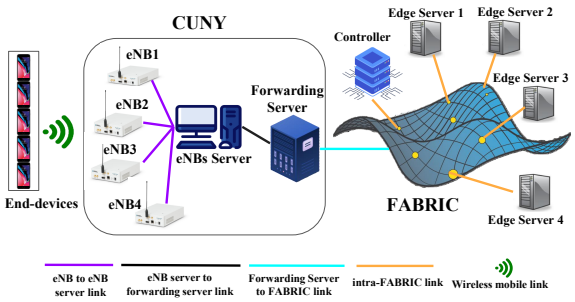

Fig. 1: Smart city testbed implementation

A. Testbed Setup and Experiment Design

The smart city testbed, illustrated in Fig. 1, comprises of multiple components:

1) End devices: To replicate camera-enabled end-devices, we use 5 Nokia 2.2 smartphones with that has multi-view video dataset already pre-stored. The devices connect to the forwarding server through an LTE network and are synchronized to simultaneously transmit the pre-stored video frames to the forwarding server.

2) Base stations: To enable reliable communication between end devices and forwarding server(s), we deploy eNB-based base stations. These stations handle data transmission over the LTE network, ensuring low-latency connectivity for video frame streaming.

3) Forwarding server: To manage data flow between edge devices and end servers, we deploy a forwarding server that receives video streams from the base stations and forwards them to the appropriate edge servers based on controller's decisions. The end devices, base stations, and forwarding servers are implemented within lab hardware at CUNY.

4) End servers: The smart city edge servers are simulated using four virtual servers from FABRIC testbed [30], each equipped with a 16-core CPU (2.8 GHz), 32 GB of RAM, 100 GB of disk space, and an NVIDIA Quadro RTX 6000 GPU for accelerated processing. State-of-the-art 3D reconstruction pipeline are pre-installed on the edge servers.

5) Controller: The controller or resource manager is deployed on a machine within FABRIC, equipped with a 20-core CPU (2.8 GHz), 32 GB RAM, and 100 GB disk space. It runs the proposed RL agents, making dynamic decisions on camera and server selection based on system state.

6) Network Implementation: To create a real-world setting, we establish an LTE network for end-device connectivity to the forwarding server using the srsRAN4G library, enabling a controlled and adaptable LTE environment. The LTE network is supported by multiple eNodeBs, each simulated using USRP-2901 devices with sub-1 GHz antennas. The Evolved Packet Core (EPC), responsible for managing high-level network functions such as user session management, mobility, security, and external data routing, is implemented using *Open5GS*. It is co-located on the same machine as the eNodeBs, creating a compact and integrated smart city environment setup.

7) Multi-view 3D reconstruction datasets: We use our own multi-view dataset [2] that represents different degrees of indoor dynamic scenes, e.g., Pickup, Walk and Handshake. Each scene captures dynamic indoor environments using five synchronized Raspberry Pi cameras, providing long sequence recordings for 3D reconstruction. The scenes are structured with diverse objects, controlled lighting, and a central focal point to ensure consistency. Videos are recorded at 25-30 FPS and are later processed into sequential images for 3D reconstruction. This dataset enables a detailed analysis of 3D reconstruction performance.

8) Simulating disruption: Camera disruption probabilities are simulated over a 4000-frame timeline using a predefined correlation matrix to capture interdependencies. Cameras 1 and 2, and Cameras 3 and 5, are highly correlated, reflecting scenarios where nearby cameras or shared wireless channels fail together (e.g., due to fire, smoke, or interference). Additionally, 10 random bump events—each lasting 50 frames—are injected across camera groups $\{1, 2\}$, $\{3, 5\}$, and Camera 4 independently, introducing temporary spikes in failure probability with a threshold of 0.6, producing binary traces (0 = disruption, 1 = normal). Server-side disruptions are modeled over the same timeline with four independent servers, each having a baseline transmission latency of 150 ms with small fluctuations. Ten random latency spike events are injected independently, adding 400–1200 ms for 50 frames per event, emulating temporary server overloads while preserving independent behaviors.

9) 3D reconstruction Pipeline: Due to the time-sensitive nature of 3D reconstruction, generating the final dense point cloud using the openMVG/openMVS pipeline creates a latency bottleneck for timely decision-making. On our testbed edge server, this process takes 5 seconds, exceeding near real-time requirements. However, sparse point clouds generated as intermediate outputs are leveraged to update the RL model's policy before dense reconstruction completes. In the absence of ground truth, reconstruction quality is quantified by projecting sparse 3D points back onto the original 2D images and measuring the average reprojection error.

10) Evaluation metrics: We evaluate our RL-based system in terms of observed performance, focusing on reliability, quality, and latency. *Reliability* is the primary metric, defined as the proportion of frames meeting both reconstruction quality and end-to-end latency thresholds (Section III-A); *latency* captures the average end-to-end delay per frame from image capture to reconstruction; and *reconstruction quality* is measured using a projection-based metric.

11) RL agent training parameters: We evaluate both fixed and adaptive variants of tabular Q-learning for camera and server selection. All Q-tables are initialized to zero, and training occurs online during system execution. For the camera selection task, the fixed agent uses a learning rate of $\alpha = 0.9$, a discount factor of $\gamma = 0.1$, and a fixed exploration rate of $\epsilon = 0.1$. The adaptive camera agent starts with a base learning rate of 0.5 and an initial exploration rate of $\epsilon = 1.0$, which decays to a minimum of 0.05 based on runtime conditions. For server selection, the fixed Q-learning agent is configured with $\alpha = 0.9$, $\gamma = 0.1$, and $\epsilon = 0.1$. The adaptive server agent

TABLE I: Camera selection performance comparison

Camera Selector	Avg PQ	Avg Recon Lat. (s)	Avg Total Lat. (s)	Reliability
Q-learning	582	0.79	2.54	62.53%
Greedy-3	542	0.67	2.45	59.92%
Epsilon-Greedy Bandit	443	0.71	3.45	24.69%
Adaptive Q-learning	378	0.70	2.47	36.38%
Random	386	0.69	2.26	25.05%

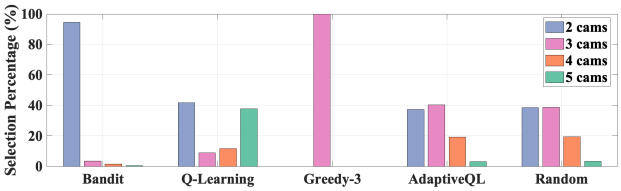

Fig. 2: Distribution of selected camera subsets

uses an initial learning rate of 0.3, an initial exploration rate of $\epsilon = 0.2$, and a discount factor of $\gamma = 0.95$.

12) Baseline approaches: We compare our proposed Q-learning and Adaptive Q-learning strategies to baselines for camera and server selection, spanning non-learning, heuristic, and learning-based approaches.

Camera selection baselines: We compare our RL-based strategies against three baselines: (i) *Random*, which uniformly samples a valid camera subset at each timestep as a non-adaptive lower bound; (ii) *Greedy-3*, which always selects the 3-camera subset predicted to yield the highest reconstruction quality, ignoring latency and variability; and (iii) *Epsilon-Greedy Bandit*, which treats camera subset selection as a multi-armed bandit and balances exploration and exploitation via an ϵ-greedy policy.

Server selection baselines: We compare against two non-learning baselines: (i) *Round-Robin*, which cycles through servers in a fixed order without considering load or latency, serving as a simple deterministic benchmark; and (ii) *Latency-Greedy*, which selects the server with the lowest estimated latency based on an exponentially weighted moving average, adapting to recent trends but without anticipating future variations or accounting for camera quality.

B. Testbed Results and Discussions

1) Camera selection performance: To evaluate different camera selection strategies, we fix the server policy to a *Round-Robin* scheduler so all strategies operate under identical server conditions. Each camera strategy dynamically selects a subset of available cameras at each timestep to optimize quality and responsiveness. Reliability is defined as the percentage of frames meeting all of the following: (1) at least 400 matching points per view, (2) total latency under 3 sec, and (3) reconstruction latency under 1 sec.

Tab. I shows that the proposed *Q-learning* achieves the highest reliability (62.53%) and overall point cloud quality, demonstrating its ability to learn policies over time. Interestingly, *Greedy-3*, despite being a simple deterministic policy, performs competitively, achieving a reliability of 59.92% with lower average latency than the Q-learning agent. This suggests that consistently selecting a strong camera subset can serve as a robust baseline in stable environments. The *Epsilon-Greedy Bandit* and *Random* strategies perform poorly in terms of both reliability (24.69% and 25.05%, respectively) and average point cloud quality. These results reflect their inability to account for delayed feedback and their tendency to make suboptimal choices under variable network conditions. The *Adaptive Q-learning* agent, while designed to respond to environmental changes, exhibits instability and lower reliability (36.38%), likely due to overreaction to short-term fluctuations. Fig. 2 visualizes the selection distribution of camera subsets for each strategy. While *Q-learning* and *Greedy-3* agents concentrate on a limited set of high-quality subsets, the *Adaptive Q-learning* and *Random* policies show more dispersed behavior. These findings highlight the importance of balancing adaptability and consistency in view selection for real-time 3D reconstruction.

TABLE II: Server selection performance comparison

Server Agent	Total Lat. (s)	Recon Lat. (s)	Reliability
Round-Robin	3.20	0.91	31.5%
Latency-Greedy	5.32	2.14	4.1%
Q-Learning	3.47	2.72	16.9%
Adaptive Q-Learning	**3.62**	**2.13**	**55.0%**

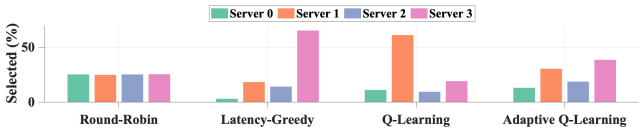

Fig. 3: Distribution of selected servers

2) Server selection performance: To isolate the impact of server selection policies, all experiments use the *Greedy-3* camera selection strategy. Reliability is defined as the percentage of frames meeting all of the following: (1) at least 500 matching points per view, (2) total latency under 3 sec, and (3) reconstruction latency under 1 sec.

Tab. II shows that *Round-Robin* achieves strong performance with a reliability of 31.5%, outperforming both *Latency-Greedy* and *Q-learning*. While *Q-learning* is designed to adapt based on feedback, it suffers from delayed response to performance degradation and tends to repeatedly select servers that were previously fast, leading to overload. In contrast, the fixed *Round-Robin* policy maintains a balanced server distribution — assigning approximately 25% of frames to each server, which results in more stable latency under dynamic conditions. The uniform selection pattern is clearly observable in the server selection results (Fig. 3). This selection behavior also influences system latency, which varies across the four strategies as shown in Fig. 4. *Round-Robin* exhibits the highest median latency and widest inter-quartile range (IQR) among the more stable strategies, indicating its consistent but non-optimized performance. In contrast, both *Latency-Greedy* and *Adaptive Q-learning* show lower medians and narrower IQRs, suggesting more efficient server choices under typical conditions. However, the standard *Q-learning* agent displays a much wider spread, with a long box and numerous high-latency outliers. This indicates instability and delayed responsiveness to changing network conditions. These results highlight the importance of balancing adaptiveness with robustness in server selection policies. All evaluation related codes and data are available through Github [31].

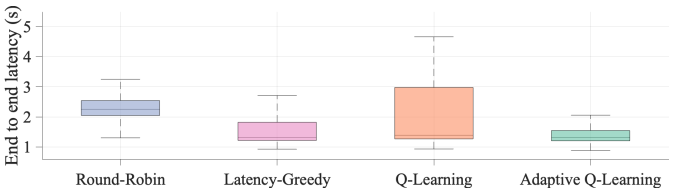

Fig. 4: Distribution of end-to-end latency

V. Conclusions

Latency-sensitive and edge-native multi-view 3D reconstruction is crucial for mission-critical applications, where meeting strict latency and quality constraints enables effective decision-making, often in the presence of disruptions. We present an RL-based framework that dynamically selects camera subsets and allocates server resources based on observed conditions. The method learns adaptive policies that enhance responsiveness and reliability under unpredictable disruptions. Evaluation in simulated and physical environments shows substantial gains: the camera selection agent achieves up to 15% higher reliability than random selection and 2% over Greedy-3, while the adaptive server agent improves up to 24% over round-robin and 50% over latency-greedy baselines. These results demonstrate RL's potential for robust, disruption-aware decision-making in latency-sensitive edge systems.

References

[1] C. Schönauer, E. Vonach, G. Gerstweiler, and H. Kaufmann, "3d building reconstruction and thermal mapping in fire brigade operations," in *Proceedings of the 4th Augmented Human International Conference*, AH '13, (New York, NY, USA), p. 202–205, Association for Computing Machinery, 2013.

[2] X. Zhang, H. Gan, A. Pal, S. Dey, and S. Debroy, "On balancing latency and quality of edge-native multi-view 3d reconstruction," in *Proceedings of the Eighth ACM/IEEE Symposium on Edge Computing*, SEC '23, (New York, NY, USA), p. 1–13, Association for Computing Machinery, 2024.

[3] X. Zhang, A. Pal, and S. Debroy, "Effect: Energy-efficient fog computing framework for real-time video processing," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 493–503, 2021.

[4] S. Yousefi, M. Mounesan, and S. Debroy, "Advar-dnn: Adversarial misclassification attack on collaborative dnn inference," in *2025 IEEE 50th Conference on Local Computer Networks (LCN)*, pp. 1–9, 2025.

[5] M. Zneit, X. Zhang, M. Mounesan, and S. Debroy, "Adversarial autoencoder based model extraction attacks for collaborative dnn inference at edge," in *NOMS 2025-2025 IEEE Network Operations and Management Symposium*, pp. 01–09, 2025.

[6] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, "Openmvg: Open multiple view geometry," in *International Workshop on Reproducible Research in Pattern Recognition*, pp. 60–74, Springer, 2016.

[7] D. Cernea, "OpenMVS: Multi-view stereo reconstruction library." 2020.

[8] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 767–783, 2018.

[9] X. Zhang, M. Li, A. Hilton, A. Pal, S. Dey, and S. Debroy, "End-to-end latency optimization of multi-view 3d reconstruction for disaster response," in *2022 10th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 17–24, IEEE, 2022.

[10] A. Farshian, M. Götz, G. Cavallaro, C. Debus, M. Nießner, J. A. Benediktsson, and A. Streit, "Deep-learning-based 3-d surface reconstruction—a survey," *Proceedings of the IEEE*, vol. 111, no. 11, pp. 1464–1501, 2023.

[11] T. Samavati and M. Soryani, "Deep learning-based 3d reconstruction: a survey," *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9175–9219, 2023.

[12] X.-F. Han, H. Laga, and M. Bennamoun, "Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 5, pp. 1578–1604, 2019.

[13] F. Lin, Y. Yue, S. Hou, X. Yu, Y. Xu, K. D. Yamada, and Z. Zhang, "Hyperbolic chamfer distance for point cloud completion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14595–14606, October 2023.

[14] I. Viola, S. Subramanyam, and P. Cesar, "A color-based objective quality metric for point cloud contents," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, 2020.

[15] R. Diniz, P. G. Freitas, and M. C. Q. Farias, "Color and geometry texture descriptors for point-cloud quality assessment," *IEEE Signal Processing Letters*, vol. 28, pp. 1150–1154, 2021.

[16] G. Zhang and Y. Chen, "A metric for evaluating 3d reconstruction and mapping performance with no ground truthing," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 3178–3182, 2021.

[17] Q. Liu, H. Yuan, H. Su, H. Liu, Y. Wang, H. Yang, and J. Hou, "Pqa-net: Deep no reference point cloud quality assessment via multi-view projection," *IEEE transactions on circuits and systems for video technology*, vol. 31, no. 12, pp. 4645–4660, 2021.

[18] M. Mounesan, M. Lemus, H. Yeddulapalli, P. Calyam, and S. Debroy, "Reinforcement learning-driven data-intensive workflow scheduling for volunteer edge-cloud," in *2024 IEEE 8th International Conference on Fog and Edge Computing (ICFEC)*, pp. 79–88, 2024.

[19] M. Mounesan, X. Zhang, and S. Debroy, "Edgerl: Reinforcement learning-driven deep learning model inference optimization at edge," in *2024 20th International Conference on Network and Service Management (CNSM)*, pp. 1–5, 2024.

[20] M. Mounesan, X. Zhang, and S. Debroy, "Infer-edge: Dynamic dnn inference optimization in just-in-time edge-ai implementations," in *NOMS 2025-2025 IEEE Network Operations and Management Symposium*, pp. 1–9, 2025.

[21] K. Fan, Z. Chen, Q. Liu, G. Ferrigno, and E. D. Momi, "A reinforcement learning approach for real-time articulated surgical instrument 3-d pose reconstruction," *IEEE Transactions on Medical Robotics and Bionics*, vol. 6, no. 4, pp. 1458–1467, 2024.

[22] W. Qian and R. W. L. Coutinho, "A reinforcement learning-based orchestrator for edge computing resource allocation in mobile augmented reality systems," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2023.

[23] H. Zhou, X. Wang, M. Umehira, X. Chen, C. Wu, and Y. Ji, "Wireless access control in edge-aided disaster response: A deep reinforcement learning-based approach," *IEEE Access*, vol. 9, pp. 46600–46611, 2021.

[24] Y. Chen, S. Chen, K. C. Li, *et al.*, "DRJOA: intelligent resource management optimization through deep reinforcement learning approach in edge computing," *Cluster Computing*, vol. 26, pp. 2897–2911, 2023.

[25] J. Jeong, J. Ha, and M. Kim, "Reset: Reducing the service disruption time of follow me edges over wide area networks," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 159–166, 2019.

[26] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4225–4234, 2019.

[27] M. N. Absur, S. Brahma, and S. Debroy, "Poster: Reliable 3d reconstruction for ad-hoc edge implementations," 2024.

[28] K. He, C. Sui, C. Lyu, Z. Wang, and Y. Liu, "3d reconstruction of objects with occlusion and surface reflection using a dual monocular structured light system," *Appl. Opt.*, vol. 59, pp. 9259–9271, Oct 2020.

[29] Q. Lu, Y. Pan, L. Hu, and J. He, "A method for reconstructing background from rgb-d slam in indoor dynamic environments," *Sensors*, vol. 23, no. 7, 2023.

[30] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47.

[31] GitHub, "Github repository." https://github.com/dissectlab/Edge-CNSM2025.git, 2025. Accessed: Sep 21, 2025.