

# Occlusion-Aware Camera Selection in Vehicular Networks

Ruiqi Wang, *Student Member, IEEE*, and Guohong Cao, *Fellow, IEEE*,

**Abstract**—Camera sensors are widely used to perceive traffic environments, understand traffic condition, and help avoid accidents. Since most sensors are limited by line-of-sight, the perception data from vehicles can be uploaded and shared through the edge server. To reduce bandwidth, storage and processing cost, we propose an edge-assisted camera selection system that selects only necessary camera images for sharing. The selection process is based on camera metadata which describes each cameras' coverage in terms of locations, orientations, and fields of view. Different from existing work, our metadata-based approach can detect and locate occlusions by leveraging depth sensors, and then precisely and quickly calculate the actual camera coverage and identify the coverage overlap. Based on camera metadata, we study a camera selection problem that aims to select a limited number cameras to maximize total coverage, and solve it with an efficient algorithm. To further reduce bandwidth consumption, we first introduce similarity-based redundancy suppression and sector-based selection techniques. We then propose a *Redundancy-Aware Sector Selection* algorithm, which incorporates image redundancy into the sector selection process to improve bandwidth efficiency. Extensive evaluations demonstrate that our algorithms can effectively maximize coverage with bandwidth constraint.

**Index Terms**—Vehicular Networks, Camera Sensor, Edge Computing

## I. INTRODUCTION

Today's advanced driver assistance systems rely on various sensors to perceive the vehicle's surrounding environments, allowing them to understand traffic conditions and prevent accidents. However, the perception data collected by sensors from a single vehicle has limitations because most sensors are restricted by line of sight. Consider the example shown in Fig. 1, the pedestrian behind truck *B* is crossing the road, while car *A* is passing through the intersection. Car *A* can not see the pedestrian because its view is blocked by truck *B* which is waiting to turn at the intersection. In this scenario, only relying on sensors on car *A* is not able to detect the crossing pedestrian, and hence resulting in an accident.

This accident can be avoided by sharing the collected traffic information among vehicles. A specific way of information sharing is to offload the perception data to the edge server. The edge server collects and analyzes information from vehicles and then builds a real-time map that tracks all objects in traffic [1]–[3]. Based on the map, the edge server can send messages to vehicles about the environment beyond their line of sight

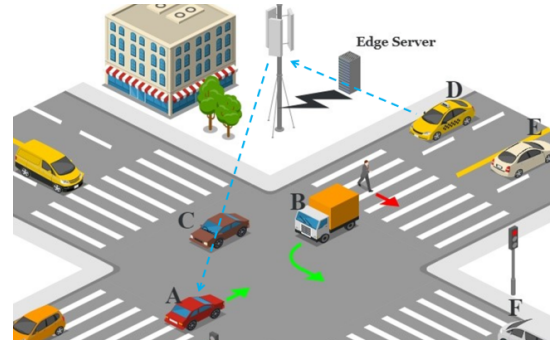


Fig. 1: A traffic scenario where the pedestrian behind truck *B* is not visible to car *A*.

[4], [5]. Specifically, in Fig. 1, the crossing pedestrian can be observed by the cameras on car *D*, which can upload the information about the crossing pedestrian to the edge server. Then, the edge server can send a message to *A* and help *A* avoid such a possible accident.

There has been some research on edge-assisted information sharing [1], [2], and they generally follow two approaches and both face some challenges. One is to upload all camera images to the server, but this may cause network congestion especially when many vehicles exist in a busy intersection. In the other approach, vehicles perform some local object detection, and only upload detected objects to the edge server; however, the object detection may not be accurate due to the resource limitation on vehicles as detailed in [6]. Moreover, only uploading detected objects may cause a lack of generality [7], for example street reconstruction needs more details about the street view rather than just detected objects.

To address these problems, we propose an edge-assisted camera selection system that selects only necessary camera images to upload. The selection is based on the camera metadata which describes the coverage of the cameras represented with GPS locations, orientations, and field of views. Similar ideas have been studied in a different context in [8]–[10]. However, they do not consider occlusions and object (vehicle) movements, and then the calculated camera coverage may not be accurate when directly applied to vehicular networks. We leverage depth images generated by depth sensors to detect and locate occlusions, allowing for precise and efficient calculation of actual camera coverage under resource constraints.

Due to bandwidth limitations, the number of selected cameras must be restricted. To have a better understanding of the traffic conditions, the area covered by the selected cameras should be as large as possible. Based on the camera metadata,

The authors are with School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA 16802.  
E-mail: {ruw400, gxc27}@psu.edu.

we formulate the *Max-Coverage* problem, which focuses on selecting a limited number of cameras to maximize total coverage. To reduce the bandwidth consumption, we incorporate redundancy suppression and sector-based selection as supporting techniques. Specifically, redundancy suppression filters out highly similar images to save bandwidth, while sector-based selection divides camera images into non-overlapping sectors, enabling more flexible and efficient area coverage. Although these techniques can save bandwidth, the conserved bandwidth remains unused, resulting in low bandwidth efficiency. To address this, we propose a redundancy-aware sector selection approach that reallocates saved bandwidth to select additional sectors. Specifically, when an image is deemed redundant, the saved bandwidth is reallocated to other sectors, improving overall bandwidth efficiency. We formulate this as a Constrained Markov Decision Process (CMDP), and propose a heuristic algorithm to solve it effectively.

In summary, the paper has the following main contributions. First, we propose an edge-assisted camera selection system that only selects the necessary camera images based on metadata. Different from existing work, our metadata based approach considers camera occlusions by leveraging depth sensors. Second, we study the camera selection problem which aims to maximize the total camera coverage under bandwidth constraints, and solve it with an efficient algorithm. We also introduce similarity based redundancy suppression and sector-based selection techniques to reduce bandwidth consumption. Third, we formulate and address the redundancy-aware sector selection problem, which incorporates image redundancy into the sector selection process to further improve bandwidth efficiency.

The rest of the paper is organized as follows. In Section II, we describe how to detect occlusions and represent camera coverage with metadata. In Section III, we present the camera selection problem, and propose an efficient solution. Section IV presents redundancy suppression techniques to further improve performance. In Section V, we divide camera coverage into sectors and study the redundancy-aware sector selection problem. We present evaluation results in Section VI. Section VII reviews related work and Section VIII concludes the paper.

## II. CAMERA METADATA

To accurately represent the camera coverage with metadata, we need to detect occlusions in vehicular networks. Thus, we first introduce techniques for occlusion detection and then present the details about camera metadata.

### A. Occlusion Detection

The coverage of a camera can be modeled as a sector, where the center is the location of the camera, the radius is the maximum range of the camera, and the field of view (FoV) is an angle that shows how wide the camera can cover. However, in the real scenario, the camera view can be blocked by objects such as buildings or vehicles, and these blocked areas are not visible to the cameras. For example, Fig. 2 (a) shows a scenario at an intersection. In the image, cars *A*, *B*, *C* and *D* block

different part of the camera view and nothing behind them are covered. Thus, to show the real coverage of the camera, these occlusions should be detected and the blocked areas should be removed.

**Local object detection:** One natural solution to identify occlusion is by running object detection algorithms on the collected camera image. As shown in Fig. 2 (a), by detecting the vehicles, the occlusions can be identified and the blocked areas can be removed. However, most object detection models are based on deep neural networks which are computational intensive. As illustrated in Table. I, the running time of a typical object detection algorithm Faster R-CNN [11] is over 300 ms on a single  $1242 \times 375$  image on NVIDIA Jetson TX2 [12]. Hence, it is impossible to generate real-time traffic map.

As another solution, we can use light-weight object detectors such as the Single Shot Detector (SSD) [13]. Although the light-weight detector has low delay (*i.e.*, 43 ms), it sacrifices accuracy where some important objects may be missed. As shown in Fig. 2 (b), two vehicles in red boxes are detected by SSD, but others are missed. Then, the area blocked by undetected vehicles will be considered as covered, which is not correct.

**Depth sensor:** Instead of identifying occlusions with inaccurate local object detection, we propose to utilize depth sensors. Most cameras on modern vehicles are equipped with depth sensors capable of generating depth images, enabling them to capture the distance information between vehicles and other objects. Each pixel in the depth image corresponds to a point in the real world, while the value of each pixel represents the distance between the depth sensor and the corresponding real-world point. Therefore, based on the 3D projection rules, pixels in depth images can be projected to the real world with precise coordinates. Then, with such information, we can identify the occlusions. This property makes depth sensors the best choice to locate occlusions and measure the distances between the occlusions and vehicles.

Depth images typically consist of millions of pixels, but only part of them represent occlusions. The rest may depict the ground plane, sky, or noise beyond the sensor's maximum range, rendering those pixels is unnecessary. To reduce the computational complexity, we propose the following preprocessing steps to remove less useful pixels. First, we remove the pixels of sky and out-of-range noise by comparing the depth of pixels to the maximum sensing range. If the depth is larger than the maximum sensing range, the pixel should be removed. Second, we exclude the ground plane from consideration, since it is not considered as occlusions. Given the height  $h_{dep}$  of the depth sensor above the ground, the height of the corresponding real-world point of any pixel  $(u, v)$  can be calculated based on the 3D projection rules:

$$h = h_{dep} - (v - H/2) \cdot d_{(u,v)} / f$$

where  $H$  denotes the height of the depth image size,  $d_{(u,v)}$  is the depth of the pixel  $(u, v)$ , and  $f$  denotes the focal length. With the heights of pixels, we use the Random Sample Consensus (RANSAC) algorithm [14] to identify and remove the pixels of the ground plane.



Fig. 2: (a) Image of a vehicle's front camera in which other vehicles block the view. (b) Objects detected by SSD.

	Depth sensor	SSD	Faster R-CNN
<b>Recall</b>	77.21%	25.17%	83.16%
<b>Time (ms)</b>	5	43	318

TABLE I: Comparisons of occlusion detection techniques.

Finally, each occlusion corresponds to a cluster of pixels, with the width of the cluster reflecting the size of the occlusion. Large objects such as vehicles typically result in wide clusters and are therefore considered as occlusions. Conversely, smaller objects like utility poles and traffic signs produce narrower clusters that do not obstruct the view significantly, and can be removed.

For example in Fig. 3 (a), we show the depth image captured by a vehicle. By leveraging our proposed techniques to remove less useful pixels, we only keep the pixels that correspond to the occlusions, shown in Fig. 3 (b). As a result, after applying these processing steps, we can effectively detect the occlusions, *i.e.*, the four vehicles as shown in Fig. 2 (a).

To compare the performance of occlusion detection using depth sensors versus occlusion detection using local object detection approaches (Faster R-CNN and SSD), we randomly select 1,000 camera images and the corresponding depth images from the KITTI dataset [15]. These images are then used to evaluate the performance on NVIDIA Jetson TX2 [12]. The running time and recall are shown in Table I. We choose recall as the performance metric because it indicates the percentage of true positive detections over all existing occlusions. Each occlusion corresponds to a blocked area, which may contain important information such as pedestrians. Missing occlusions result in false negatives, where the blocked area is considered covered, potentially ignoring risks in that area. Therefore, to assess the accuracy of different approaches, we aim to detect as many occlusions as possible while minimizing false negatives."

From the table, we can see that Faster R-CNN achieves the highest recall of 83.16% [16], outperforming the other two approaches. However, it also has the longest running time, *i.e.*, 318 ms per image. Such a delay could render the traffic scenario quickly outdated due to the rapid movement of vehicles. By leveraging depth sensors, the execution time for occlusion detection can be reduced to 5 ms, which is much faster than SSD and Faster R-CNN. Moreover, it achieves a comparable recall to Faster R-CNN, and much higher recall than SSD. Therefore, we use depth sensors to detect occlusions.

### B. Camera Metadata

After identifying the occlusions, we need to represent the real coverage of the camera. Since the pixels in depth image can be projected into real world coordinate, the occlusions

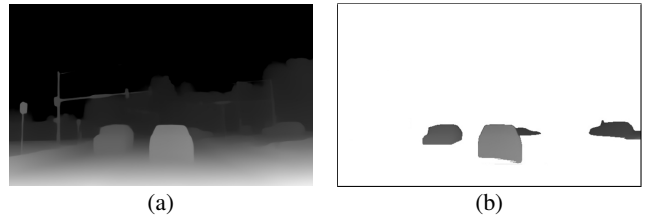


Fig. 3: (a) Depth image of the vehicle. (b) Occlusions after removing the pixels of sky, ground plane and tiny objects.

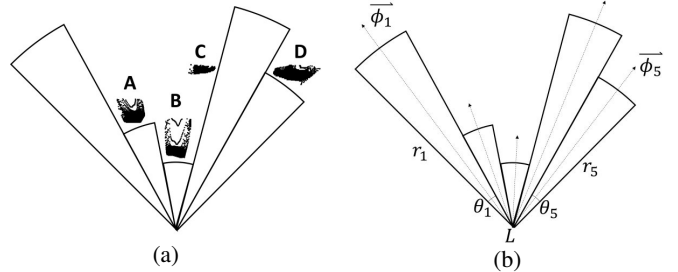


Fig. 4: (a) Sectors split by occlusions. (b) Camera metadata.

are represented as point cloud. Then, based on the point cloud, we can identify the location of the detected occlusions, and split the coverage area into multiple sectors, which can be represented efficiently. As shown in Fig. 4 (a), the point cloud of the occlusions split the camera coverage into 5 small sectors. Each sector is represented by a 3-tuple  $(r_i, \vec{\phi}_i, \theta_i)$ , illustrated in Fig. 4 (b), where  $r_i$  denotes the radius of the sector.  $\vec{\phi}_i$  indicates the orientation of each sector, and  $\theta_i$  is the central angle of the sector.

We introduce a new data structure called *camera metadata* to represent the true coverage of the camera. Specifically, the metadata of a camera consists of the GPS location  $L$ , the camera ID, and the sectors represented by 3-tuples. For example, the camera metadata in Fig. 4 (b) should be:  $(L, id, (r_1, \vec{\phi}_1, \theta_1), (r_2, \vec{\phi}_2, \theta_2), (r_3, \vec{\phi}_3, \theta_3), (r_4, \vec{\phi}_4, \theta_4), (r_5, \vec{\phi}_5, \theta_5))$ .

Based on the metadata, the camera coverage can be calculated. Consider the graph in Fig. 5 which is constructed based on the metadata of the vehicle cameras. It represents the scenario shown in Fig. 2 (a) where vehicles are represented by  $A, B, C, D$  and  $E$ , which is the one taking the picture. Each vehicle has a front camera, the GPS information in the camera metadata locates the camera in the graph, and the geometric information in 3-tuples specifies the area covered by the camera. The camera coverage is represented in different colors. As shown in the figure, these cameras have large overlaps, and uploading all these images to the edge server could waste a large amount of bandwidth, storage and processing capability. Thus, we present two camera selection algorithms which only select the necessary camera images based on the metadata in the following two sections.

## III. ACHIEVING MAXIMUM COVERAGE

In this section, we consider the scenario that the edge server has a limited bandwidth which can only support the uploading of  $k$  camera images. We formulate the *Max-Coverage* problem and then propose a greedy algorithm to solve it.

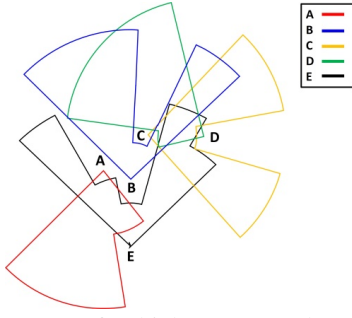


Fig. 5: The coverage of vehicle cameras shown in Fig. 2 (a), constructed with the camera metadata.

### A. Problem Statement

**Definition 1 (Max-Coverage).** Given a set of  $n$  cameras  $\mathcal{C} = \{c_1, c_2, \dots, c_n\}$  and the camera metadata of each camera. Our goal is to select  $k$  cameras such that the total coverage area is maximized.

**Theorem 1.** This Max-Coverage problem is NP-hard.

*Proof.* We prove the NP-hardness of the *Max-Coverage* problem via a reduction from the weighted maximum coverage problem [17]. In the weighted maximum coverage problem, there is a collection of sets, and each element in a set has a weight. The problem is to select a sub-collection of  $k$  sets such that the total weight of elements in the union of the sub-collection is maximized.

For any instance of the weighted maximum coverage problem, we can construct an instance of the *Max-Coverage* problem. Given a set of  $n$  cameras  $\mathcal{C} = \{c_1, c_2, \dots, c_n\}$ , the coverage of these cameras are partitioned into a set of small, non-overlapping regions,  $O = \{o_1, o_2, \dots\}$ . Each  $o_i$  is a region with a weight of its area, and each camera  $c_i$  covers multiple regions, and the coverage forms a set  $O_{c_i} \subseteq O$ . The collection,  $\{O_{c_1}, O_{c_2}, \dots, O_{c_n}\}$ , constructs the set collection in the weighted maximum coverage problem. The number  $k$  in the *Max-Coverage* problem is the same as the  $k$  in the weighted maximum coverage problem, corresponding to the size of the sub-collection.

A solution  $S$  to this instance of the *Max-Coverage* problem is the  $k$  selected cameras that maximizes the coverage area. When regions are seen as weighted elements,  $S$  maximizes the total weight of elements. Thus,  $S$  is also a solution to the weighted maximum coverage problem. This completes the reduction and hence the proof.  $\square$

### B. The Max-Coverage Algorithm

Since the *Max-Coverage* problem is NP-hard, we propose a greedy algorithm as shown in Algorithm 1 to solve it. Starting with an empty set of selected cameras  $S = \emptyset$ , our algorithm selects cameras from the cameras set  $\mathcal{C}$  and adds them to  $S$  round by round. In each round, the algorithm chooses the camera  $c_i$  with the largest coverage and moves it to the selected cameras set  $S$  from the cameras set  $\mathcal{C}$ . After selecting the camera  $c_i$ , the area covered by this camera is removed. This algorithm repeats the above steps until  $k$  cameras are selected.

For example, as shown in Fig. 6, there are three vehicles A, B and C with five cameras,  $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5\}$ . Each

---

### Algorithm 1 The Max-Coverage Algorithm

---

**Input:** Cameras set:  $\mathcal{C}$ , required number of cameras  $k$

**Output:** Selected cameras set:  $S$

- 1:  $S \leftarrow \emptyset$
  - 2: **while**  $|S| < k$  **do**
  - 3:   Find a camera  $c_i \in \mathcal{C}$  with the maximum coverage
  - 4:    $S \leftarrow S \cup c_i$
  - 5:    $\mathcal{C} \leftarrow \mathcal{C} \setminus c_i$
  - 6:   Update the coverage by removing the overlap  
 $(\cup_{c_j \in \mathcal{C}} O_{c_j}) \cap O_{c_i}$
  - 7: **end while**
  - 8: **return**  $S$
- 

camera has a  $100^\circ$  FoV and 30 meters range. Cameras  $c_2$  and  $c_4$  are blocked by vehicle C and camera  $c_5$  is blocked by vehicle B. The coverage of these cameras is partitioned into regions,  $O = \{o_1, o_2, \dots, o_{11}\}$ .

With  $k = 3$ , our *Max-Coverage* algorithm will select three cameras in following steps. In the first round, camera  $c_1$  is selected and added to  $S$  because it has the largest coverage. Then, the coverage of  $c_1$  is removed, including the region  $o_2$  which overlaps with camera  $c_3$ , and the coverage area of camera  $c_3$  should be updated by subtracting the area of  $o_2$ . Since the coverage has arcs, the calculation of the exact overlap area is hard. We approximate arcs with line segments such that the coverage and overlaps can be handled as polygons. After using polygons to represent  $c_1$ 's coverage and  $c_3$ 's coverage, the Greiner-Hormann clipping algorithm [18] is applied to extract the region  $o_2$ , and the area of  $o_2$  can be calculated in  $O(m)$  time where  $m$  is the number of lines on the polygon [19]. In the second round, the regions covered by the remaining cameras are as follows:  $O_{c_2} = \{o_4\}$ ,  $O_{c_3} = \{o_3\}$ ,  $O_{c_4} = \{o_6, o_8, o_9, o_{11}\}$  and  $O_{c_5} = \{o_5, o_7, o_9, o_{10}\}$ . Since  $c_4$  covers the largest area, it is selected and added to  $S$ . Then, all regions in  $O_{c_4}$  are removed, the overlapped region  $o_9$  is extracted, and the area is subtracted from  $c_5$ 's coverage. In the third round,  $c_2$  is selected and the algorithm terminates.

Since we approximate cameras coverage with polygons and calculate the overlap area based on those polygons, we need to know the accuracy. To find a good tradeoff between accuracy and calculation time, we randomly sample 10 cameras on a road and calculate all overlapping areas using Greiner-Hormann clipping algorithm [18] and using the equations in [19]. We calculate the exact area by using lines to replace arcs every  $0.1^\circ$  which is small enough. Then, we evaluate how the degree of arcs affect the accuracy and time. We repeat the experiment 100 times on an Intel Core i7 3.80GHz machine, and the result suggests a  $5^\circ$  interval which has an area error of 0.05% and 1.7 milliseconds running time. Thus, in this paper, arcs are replaced by lines every  $5^\circ$ , for example  $c_1$ 's arc has a  $100^\circ$  FoV and it will be approximated by 20 lines.

The following theorem shows the approximation ratio of our algorithm.

**Theorem 2.** Let  $\mathcal{A}(\cdot)$  denote the coverage area of the selected cameras. Let  $S_{our}$  be the set of  $k$  cameras selected by our algorithm, and let  $S^*$  be the  $k$  cameras selected by the optimal

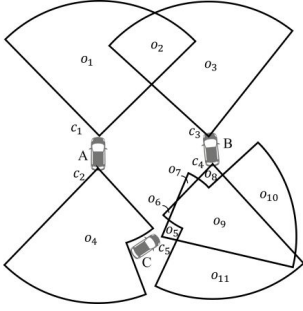


Fig. 6: The coverage of 5 cameras is partitioned into non-overlapping regions.

*solution.* Then,  $\mathcal{A}(S_{our}) \geq (1 - \frac{1}{e})\mathcal{A}(S^*)$

*Proof.* As we have proved in Theorem 1, the total weight of the solution is maximized if and only if the coverage area of selected cameras is maximized. Since the  $k$  selected cameras construct a solution to the weighted maximum coverage problem, some selected cameras must cover at least  $1/k$  fraction of the remaining uncovered area in the optimal solution.

Formally, let  $i$  be the  $i$ -th round of our proposed algorithm, and  $x_i$  be the new area covered by the  $i$ -th selected camera. The remaining uncovered area after the  $i$ -th round is  $z_i = \mathcal{A}(S^*) - \sum_1^i x_i$ . Then, for the  $(i+1)$ -th round, the new area  $x_{i+1}$  must cover  $1/k$  fraction of the remaining area  $z_i$ , and thus we have  $x_{i+1} \geq z_i/k$ . Building on this foundation, we prove the approximation ratio inductively.

**Base case:** When  $i = 1$ , the first selected camera covers at least  $1/k \cdot \mathcal{A}(S^*)$  area, and thus, the remaining uncovered area after the 1-st round has  $z_1 \leq (1 - 1/k) \cdot \mathcal{A}(S^*)$ .

**Inductive step:** Assume that for some  $i \geq 1$ , we have  $z_i \leq (1 - 1/k)^i \cdot \mathcal{A}(S^*)$ . Then,

$$\begin{aligned} z_{i+1} &\leq z_i - x_{i+1} \\ &\leq (1 - 1/k) \cdot z_i \\ &\leq (1 - 1/k) \cdot (1 - 1/k)^i \mathcal{A}(S^*) \\ &\leq (1 - 1/k)^{i+1} \cdot \mathcal{A}(S^*) \end{aligned}$$

Thus,  $z_i \leq (1 - 1/k)^i \cdot \mathcal{A}(S^*)$  holds for arbitrary  $i$ . Additionally, since  $z_i = \mathcal{A}(S^*) - \sum_1^i x_i$ , we have

$$\begin{aligned} (1 - 1/k)^i \cdot \mathcal{A}(S^*) &\geq z_i \\ 1/e \cdot \mathcal{A}(S^*) &\geq \mathcal{A}(S^*) - \sum_0^i x_i \\ \sum_1^i x_i &\geq (1 - 1/e) \cdot \mathcal{A}(S^*) \end{aligned}$$

Since  $\sum_1^i x_i$  represents all the covered area after the  $i$ -th round of our *Max-Coverage* algorithm, we have  $\sum_1^i x_i = \mathcal{A}(S_{our})$ . Hence,  $\mathcal{A}(S_{our}) \geq (1 - \frac{1}{e})\mathcal{A}(S^*)$ .  $\square$

#### IV. SIMILARITY BASED REDUNDANCY SUPPRESSION

After cameras and sectors are selected, the corresponding images should be uploaded to the edge server. Since a camera may take multiple images per second (*i.e.*, 30 frames per second for video), transmitting all of them will consume a

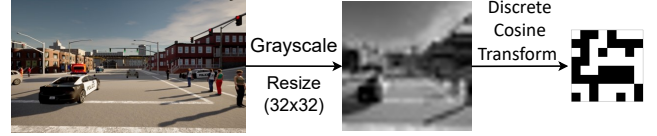


Fig. 7: pHash generates the fingerprint for the camera image.

large amount of bandwidth. The newly captured image may be similar to the previous one because vehicles cannot move too far in a short time (*i.e.*, 33 ms) and hence the changes in the images are negligible. The similarity of the consecutive images contains significant redundancy while the redundancy provides less useful information about the traffic. Thus, we can further reduce the bandwidth consumption by not uploading these similar images, and the edge server can reuse the previous uploaded image for analysis. Specifically, whenever a new image is taken, the system determines whether the image should be uploaded or not by measuring its similarity with the previous one. If the similarity is larger than a threshold  $\alpha$ , it will not be uploaded and the server should use the previous image. Otherwise, the new image is uploaded.

A natural solution to implement this idea and estimate the similarity is to use the *Intersection over Union (IoU)* metric which measures the overlap of the camera coverage. Let  $C_{pre}$  denote the previous camera coverage, and let  $C_{new}$  denote the new coverage of the same camera. Then, the IoU is calculated as follow,  $IoU = (C_{pre} \cap C_{new}) / (C_{pre} \cup C_{new})$

With the camera metadata, IoU can be easily obtained by comparing the new camera metadata with the previous one. However, IoU may fail to make correct decisions in some cases. For example, suppose  $\alpha = 0.9$ . In Fig. 8, a camera takes two images while waiting at an intersection. Fig. 8 (a) is the first frame taken at time  $t_1$ , and Fig. 8 (b) is the second frame taken at time  $t_2$ . Apparently, a new pedestrian can be detected from the second frame (in Fig. 8 (b)), which provide new information, and this image should be uploaded. However, as shown in Fig. 8 (c), the camera coverage has large overlap, and IoU is 0.96 which is larger than the  $\alpha = 0.9$ . Thus, the second frame will not be uploaded, which is a wrong decision. As another example shown in Fig. 9, two images have nearly identical views, but they have low IoU of 0.82 because the camera location changes. Since the IoU is lower than  $\alpha$ , the image will be uploaded even though it is redundant.

To address this problem, we use *perceptual hashing (pHash)* [20]–[22] to measure the similarity of the two images. The pHash generates fingerprints for images based on the content, and the procedure is shown in Fig. 7. The pHash method first converts the camera image to a  $32 \times 32$  gray-scale thumbnail, shown as the middle image in Fig. 7. Then, Discrete Cosine Transform (DCT) is applied to transform pixels in the thumbnail from the spatial domain to the frequency domain, and obtain a  $32 \times 32$  coefficient matrix. Let  $\mathcal{F}$  denote the coefficient matrix, and let  $\mathcal{I}$  denote the thumbnail. Each element of the coefficient matrix  $\mathcal{F}_{(x,y)}$  is calculated as follow.

$$\mathcal{F}_{(x,y)} = \sum_{i=0}^{32} \sum_{j=0}^{32} \mathcal{I}_{(i,j)} \cos \left[ \frac{(2i+1)\pi}{2 \times 32} x \right] \cos \left[ \frac{(2j+1)\pi}{2 \times 32} y \right]$$

The pHash only considers the low frequency features which

corresponds to the  $8 \times 8$  matrix from the upper left corner of the coefficient matrix. In Fig. 7, the right image shows the  $8 \times 8$  matrix, and then it is converted to a vector of length 64 ( $8 \times 8$ ). Next, the pHash compares the vector with its mean value to build the 64-bit fingerprint. If the value in the vector is larger than the mean, the corresponding bit in the fingerprint is 1; otherwise, the bit is 0. Then, given two fingerprints  $\mu$  and  $\nu$ , the similarity can be calculated with the Hamming distance as follows, which counts the number of bits where the corresponding values are different.

$$\text{Similarity} = \frac{1}{64} \cdot \|\{i : \mu_i \neq \nu_i, i = 1, 2, \dots, 64\}\|$$

For the example shown in Fig. 8, the pHash similarity of the two images is 0.84, compared with  $\alpha = 0.9$  the new image in Fig. 8 (b) will be uploaded as expected. On the other hand, the two similar images in Fig. 9 have a higher pHash similarity of 0.98, and then the new image will not be uploaded.

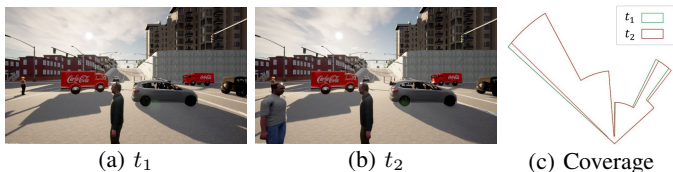


Fig. 8: The similarity: IoU = 0.96, pHash=0.84

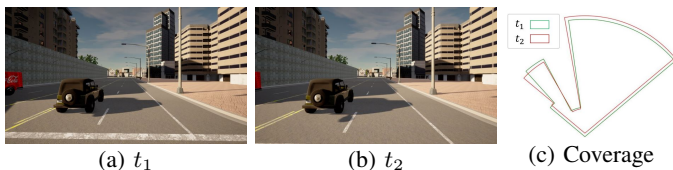


Fig. 9: The similarity: IoU = 0.82, pHash=0.98

## V. REDUNDANCY-AWARE SECTOR SELECTION

In this section, we first divide camera coverage into sectors to further reduce bandwidth consumption, and then study a redundancy-aware sector selection problem to maximize coverage while improving bandwidth efficiency.

### A. Sector-Based Selection

The *Max-Coverage* problem focuses on camera-based selection, where decisions are binary: once a camera is selected, its entire image is uploaded to the edge server. However, this approach may lead to significant coverage overlaps between selected cameras, which waste bandwidth. For example, Fig. 10 shows the coverage of cameras *A* and *E* from Fig. 5. With camera-based method, both cameras would be selected, despite the overlap in their coverage areas, represented by the shaded regions. This overlap will result in redundant data transmission and waste bandwidth. To address this problem, we propose a sector-based approach, where we divide the camera coverage into non-overlapping sectors based on the camera metadata (Section II-B), and then select the sectors to maximize the coverage. Then, we crop the camera image based on sectors, and only upload the corresponding images of the selected sectors, referred to as *sector images*<sup>1</sup>. As

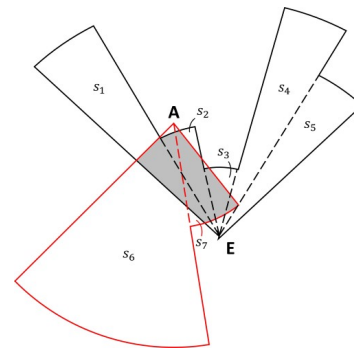


Fig. 10: Motivation of sector-based selection.

shown in Fig. 10, the coverage of camera *E* is divided into sectors  $s_1, s_2, s_3, s_4, s_5$ , and camera *A* is divided into  $s_6, s_7$ . By selecting sectors, overlap can be significantly reduced. For example, since there is significant overlap between sectors  $s_2, s_3$ , and  $s_7$ , we only select  $s_7$  and then discard  $s_2$  and  $s_3$ . This reduces the need to upload the sector images of  $s_2$  and  $s_3$ , thereby conserving bandwidth.

It is important to note that Fig. 10 shows only the bird's-eye view of sectors, while bandwidth consumption depends on the size of sector images. For example, in Fig. 10, the sector images of  $s_2$  and  $s_3$  comprise half of the image size. By excluding these parts, we can achieve a 50% bandwidth saving. Details on how to obtain sector images will be presented in Section V-D.

This sector-based selection approach can be applied to the *Max-Coverage* algorithm with appropriate extensions. In the original *Max-Coverage* algorithm (Algorithm 1), selection is based on entire cameras, where the coverage of a selected camera is selected in each iteration. To adapt to sector-based selection, the camera set  $\mathcal{C}$  is replaced with a sector set  $\mathcal{S}$ , and the selected cameras  $c_i$  are replaced with sectors  $s_i$ . Specifically, in line 3 of the algorithm, the selection step becomes: “Find a sector  $s_i \in \mathcal{S}$  with the maximum coverage.” For example, in Fig. 10,  $s_6$  is selected first since it has the largest coverage, and its coverage is removed. Then, we select sectors  $s_4, s_5, s_1$  and  $s_7$  in the following iterations. Since the coverage of sectors  $s_2$  and  $s_3$  have large overlap with sector  $s_7$ , they are not selected, reducing bandwidth consumption. After sector selection, similarity-based redundancy suppression can be applied to sector images to identify and exclude redundant content. If a sector's image contains significant redundancy, it can be excluded from upload, further reducing bandwidth consumption.

### B. Redundancy-Aware Sector Selection

Although redundancy suppression can reduce bandwidth consumption by filtering out redundant images, the saved bandwidth is often left unused, leading to inefficient resource utilization. To address this issue, we propose to consider redundancy during the selection process. By doing so, bandwidth saved from redundant sectors can be reallocated to select additional sectors, thereby improving coverage and maximizing resource efficiency. Specifically, when selecting sectors, the redundancy of sector images is evaluated. If the sector image is redundant, bandwidth can be saved by not uploading it. This

<sup>1</sup>An example of sector image is shown as the red box in Fig. 12 (c).

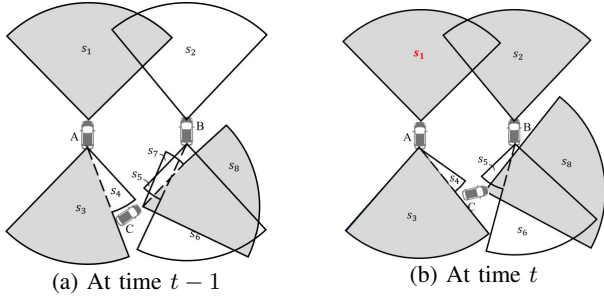


Fig. 11: Redundancy-aware sector selection

saved bandwidth can be reallocated to other sectors, enabling their selection and further enhancing overall coverage.

For example in Fig. 11, we show the sectors defined by the camera metadata. The gray sectors are selected at different time steps under bandwidth constraints. Sectors  $s_1, s_3$  and  $s_8$  are selected at time  $t-1$  and  $t$ . Since car  $C$  moves forward, the coverage of  $s_3$  and  $s_8$  changes at time  $t$ , causing their sector images to update significantly. This results in low redundancy based on the pHash value, and their sector images should be uploaded to keep the edge server updated. On the other hand, sector  $s_1$  does not change, making its sector image redundant; thus, it is not uploaded, saving bandwidth. The saved bandwidth can be used to select additional sectors, such as  $s_2$ , which has the largest coverage. This approach enables more efficient bandwidth utilization and enhances overall coverage.

To determine which sectors should be selected while considering redundancy, we formulate the redundancy-aware sector selection problem which aims to maximize coverage under bandwidth constraints. Since sector selection depends on the state of sectors (i.e., coverage and image redundancy) as well as previous selections, we model the problem as a Constrained Markov Decision Process (CMDP) [23].

**State:** The state includes all observable information relevant to the sector selection process, designed to fully and informatively represent the status of the sectors. Given that sector selection decisions are influenced by the previous selections at time  $t-1$ , image redundancy, and bandwidth, we define the state space as  $\{\mathcal{X}^{t-1}, \mathcal{H}^t, B\}$ . Specifically,  $\mathcal{X}^{t-1} = \{x_1^{t-1}, x_2^{t-1}, \dots, x_n^{t-1}\}$  represents the sectors selected at time  $t-1$ ,  $\mathcal{H}^t = \{h_1^t, h_2^t, \dots, h_n^t\}$  represents the pHash value of the sector images at time  $t$ , and  $d_i^t \in [0, 1]$ , and  $B$  represents the bandwidth constraint.

**Action:** Based on the observed state, the system decides which sectors to select. The action space is defined as  $\mathcal{X}^t = \{x_1^t, x_2^t, \dots, x_n^t\}$ , which represents the selection of sectors at time  $t$ , and  $x_i^t \in \{0, 1\}$  indicates whether sector  $i$  is selected ( $x_i^t = 1$ ) or not ( $x_i^t = 0$ ) at time  $t$ .

**Reward:** The objective is to maximize overall coverage, so the reward is defined as the area covered by the selected sectors. Given that  $\mathcal{X}^t$  represents the selected sectors, the reward is expressed as  $\mathcal{A}(\mathcal{X}^t)$ , where  $\mathcal{A}(\cdot)$  is a function that calculates the coverage.

**Constraint:** Since only sector images with less redundancy are uploaded, there are two kinds of uploads.

*Type I.* When a sector is newly selected at time  $t$ , its sector image should be uploaded. Specifically, let sector  $i$  be selected

at time  $t$ , but not selected at time  $t-1$ , we have  $x_i^t = 1, x_i^{t-1} = 0$ . Let  $z_i$  be the bandwidth required to upload the sector image of sector  $i$ , then the total required bandwidth for this kind of uploads is

$$b_1 = \sum_i z_i \cdot x_i^t \cdot \mathbb{1}(x_i^{t-1} = 0)$$

where  $\mathbb{1}(\cdot)$  is an indicator that returns 1 if the condition is satisfied; otherwise, it is 0.

*Type II.* If a sector is selected at both time  $t-1$  and  $t$  ( $x_i^t = 1, x_i^{t-1} = 1$ ), whether its sector image should be uploaded or not depends on the redundancy. Specifically, let sector  $i$  be selected, the redundancy of its sector image is  $d_i^t$ , which is calculated based on the pHash. If  $d_i^t$  is smaller than the threshold  $\alpha$  (defined in Section IV), the sector image is not redundant and will be uploaded. Formally, the total required bandwidth for this kind of uploads is

$$b_2 = \sum_i z_i \cdot x_i^t \cdot \mathbb{1}(x_i^{t-1} = 1) \cdot \mathbb{1}(d_i^t < \alpha)$$

Therefore, given the bandwidth constraint  $B$ , we have  $b_1 + b_2 \leq B$ .

### C. Proposed Method

Although reinforcement learning algorithms [24] have been used to solve the CMDP problem, most of them are designed for scenarios with a fixed action space. However, in the redundancy-aware sector selection problem, the rapid movement of vehicles causes the number of sectors to change dynamically, resulting in a highly variable action space. This makes existing reinforcement learning algorithms unsuitable for direct application. Additionally, due to real-time requirements, the selection algorithm must execute quickly to ensure that images are uploaded and analyzed in a timely manner. Reinforcement learning algorithms often involve substantial computation, which can be time-consuming and cannot satisfy the real time requirement. To address these challenges, we propose the *Redundancy-Aware Sector Selection* algorithm to solve this problem.

Our algorithm runs at each time step, selecting sectors round by round. For each round, the sector covering the largest area is selected and its coverage is removed. Based on the previous selection and the image redundancy, we determine whether the sector image of the selected sector should be uploaded. The algorithm stops when no sector can be selected under bandwidth constraint. Specifically, as shown in Algorithm. 2, at each time  $t$ , the algorithm first initializes the set  $\mathcal{X}^t$  of size  $n$  with all 0 and the total bandwidth consumption  $b = 0$  (lines 1-2), indicating no sectors are selected at the beginning. Then, the algorithm selects sectors round by round. In each round, the algorithm selects the largest sector  $i$  and set it as selected,  $x_i^t = 1$  (line 4-5). If the sector is newly selected, i.e.,  $x_i^{t-1} = 0$ , the sector image should be uploaded, and hence we add the required bandwidth to the total bandwidth consumption (line 6-7); Otherwise, the sector  $i$  is already selected previously, and we calculate the image redundancy  $d_i^t$  of its sector image (line 9). Specifically, if the sector image is not redundant, i.e.,  $d_i^t < \alpha$ , it will be uploaded, and the required bandwidth is added to

---

**Algorithm 2** The *Redundancy-Aware Sector Selection* Algorithm
 

---

**Input:** Time  $t$ ; sector set  $\mathcal{S}$ ; previous selection  $\mathcal{X}^{t-1}$ ; bandwidth constraint  $B$

**Output:** Selected sectors:  $\mathcal{X}^t$

- 1:  $\mathcal{X}^t \leftarrow \{0, 0, \dots, 0\}$
- 2:  $b \leftarrow 0$
- 3: **while**  $b < B$  **do**
- 4:   Find a sector  $i$  with the maximum coverage
- 5:    $x_i^t \leftarrow 1$
- 6:   **if**  $x_i^{t-1} = 0$  **then**
- 7:      $b \leftarrow b + z_i$   
     //  $z_i$  is the bandwidth required to upload the sector image of sector  $i$
- 8:   **else**
- 9:     Calculate the redundancy  $d_i^t$
- 10:    **if**  $d_i^t < \alpha$  **then**
- 11:      $b \leftarrow b + z_i$
- 12:    **end if**
- 13:   **end if**
- 14:   Update the coverage by removing the overlap  
      $(\cup_{j \in \mathcal{S} O_j}) \cap O_i$
- 15: **end while**
- 16: **return**  $\mathcal{X}^t$

---

the total bandwidth consumption (line 10-12); In contrast, if the sector image is redundant, it will not be uploaded to save bandwidth. After each selection, we update the coverage by removing the overlap (line 14). The algorithm stops when no sectors can be further selected due to bandwidth constraint, and all elements with  $x_i^t = 1$  in  $\mathcal{X}^t$  are selected sectors.

For example, as shown in Fig. 11 (b), our algorithm first selects the largest sector  $s_1$ . Since  $s_1$  is selected at time  $t - 1$ , we calculate and check the redundancy  $d_1^t$ . Since the sector image of  $s_1$  does not change too much, it is redundant and will not be uploaded. Then, we select  $s_3$  and  $s_8$  in the following rounds, and we set  $x_3^t = 1$  and  $x_8^t = 1$ . Due to the movement of car  $C$ , the sector images of  $s_3$  and  $s_8$  have changed, and their redundancies  $d_3^t$  and  $d_8^t$  are below the threshold  $\alpha$ . Thus, both sector images are uploaded. Since the sector image of  $s_1$  is not uploaded, the saved bandwidth can be reallocated to select another sector. Our algorithm selects sector  $s_2$ , as it has the largest coverage among unselected sectors. As a result,  $\{s_1, s_2, s_3, s_8\}$  in Fig. 11 (b) are selected. Note that although there are 4 selected sectors, only the sector images of  $\{s_2, s_3, s_8\}$  are uploaded while  $s_1$ 's sector image is not since it is redundant.

#### D. Dividing Images Based on Camera Metadata

To extract the sector images, we should accurately divide camera images based on the sectors. However, cameras produce 2D images, and sectors represent coverage from a distinct dimension, specifically the bird's-eye view. The challenge arises from the fact that there is no straightforward method to directly map sectors to images. This difficulty stems from the inherent differences in perspective and representation between

the 2D images captured by the cameras and the bird's-eye view sectors.

To fill the gap between sector representations and the sector images, we propose to utilize the camera projection rules. We first introduce the pinhole camera model [25], which illustrates the principle of most modern cameras. In Fig. 12 (a), we show the principle of pinhole camera model. To simplify, we use two red lines to represent the light from the top and bottom of the tree. As the light passes through the pinhole, the tree is projected onto a 2D image. During this projection, the location of the tree on the image depends solely on the direction of the light (i.e., from the top or bottom), with the distance of the tree having no effect on the projection. Therefore, when extracting the sector images of different sectors, we only need to consider the sectors' orientations. For example, in Fig. 12 (b), we show the sectors of the image in Fig. 2 (a) and a pinhole camera model at the center which projects the scene to the image. It is important to note that in this pinhole camera model, the image is flipped (as shown in Fig. 12 (a)) so that the right boundary of the image is displayed on the left in Fig. 12 (b). For each sector, the sector image is located by the sector's orientation while the width of the sector image is determined by the central angle. An example can be seen in Fig. 12 (b) where the scene within the second sector is projected onto the image, illustrated in red. Since the camera specifications are known to the vehicles, i.e., camera orientation  $\vec{\phi}$  and focal length  $f$ , the specific left and right boundaries of sector image of sector  $i$  can be calculated based on its orientation  $\vec{\phi}_i$  and central angle  $\theta_i$ .

$$\begin{aligned} \text{left} &= \frac{W}{2} - f \cdot \tan\left(\vec{\phi}_i - \vec{\phi} + \frac{\theta_i}{2}\right) \\ \text{right} &= \frac{W}{2} - f \cdot \tan\left(\vec{\phi}_i - \vec{\phi} - \frac{\theta_i}{2}\right) \end{aligned}$$

where  $W$  is the width of the whole camera image, and *left* and *right* denote the left and right boundaries of the sector image.

Using this formula, we calculate the boundaries of the sector image for the second sector (with orientation  $\vec{\phi}_2$  and central angle  $\theta_2$ ) in Fig. 12 (b), and the sector image is shown in the red bounding box in Fig. 12 (c).

#### E. Redundancy of Sector Images

Although the basic idea of calculating the redundancy of sector images remains the same as the camera-based method which compares the pHash values, dealing with sectors is more complicated. In camera-based selection, camera images can be distinguished by camera identities, allowing us to compare pHash values only among images with the same camera identities. However, in sector selection, the coverage of a camera can be divided into multiple sectors, each representing a different part of the same camera image. Therefore, before comparing the pHash of sector images, we should first make sure that the sectors represent the same part of the image.

Specifically, given a sector  $s_1$  selected at time  $t_1$  and another one  $s'_1$  selected at time  $t_2$  ( $t_1 < t_2$ ), we first check whether these sectors are from the same camera. If they are not from the same camera, they cannot represent the same part of image.

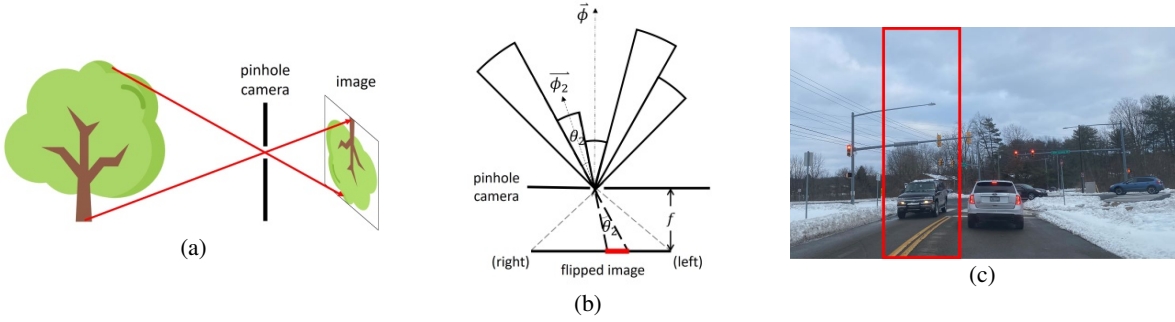


Fig. 12: (a) Principle of pinhole camera model. (b) Sectors and the corresponding parts of the image. (c) The corresponding image (red box) of the second sector.

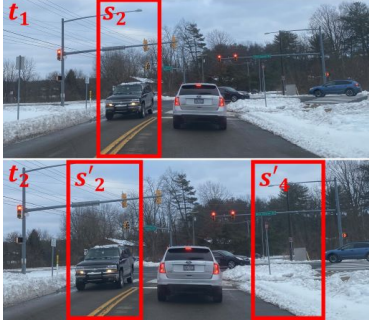


Fig. 13: Images of the same camera taken at time  $t_1$  (upper) and  $t_2$  (below).

On the other hand, if both sectors are from the same camera, we proceed to calculate the image boundaries of both  $s_1$  and  $s'_1$  based on the equations in Section V-D. Then, we check whether their sector images overlap. If there is any overlap, sector  $s'_1$  might have a similar sector image as  $s_1$  does. Then, the pHash values of their sector images are compared to calculate the redundancy. On the contrary, if there is no image overlap, it indicates that the sector image of  $s'_1$  is different from  $s_1$ 's, and hence the redundancy should be 0.

An example can be seen in Fig. 13. There are two images from the same camera at time  $t_1$  and  $t_2$ . The red boxes represent the sector images of sectors  $s_2$  (at  $t_1$ ),  $s'_2$  and  $s'_4$  (at  $t_2$ ). For the sector images of  $s'_2$  and  $s'_4$ , we first compare their overlaps. Since the sector images of  $s_2$  and  $s'_2$  overlap, their pHash values should be compared to calculate the redundancy. On the other hand, there is no overlap between the sector image of  $s'_4$  and others, then its redundancy is 0.

## VI. PERFORMANCE EVALUATIONS

In this section, we first use a case study to show the effectiveness of the proposed solution, and then evaluate its performance with extensive simulations.

### A. Evaluation Setup

Although there are some existing datasets for autonomous driving, like KITTI [15], they only collect perception data from a single vehicle. In contrast, the evaluation of information sharing requires data from multiple vehicles at the same time. Therefore, we use CARLA [26] to evaluate the performance of our proposed system. CARLA is an open source simulator for autonomous driving, which provides several high-definition

(HD) maps of different towns and cities, allowing us to simulate busy traffic in the maps. CARLA also provides a range of sensors, such as GPS, cameras and depth sensors to perceive the surrounding environment accurately. Thus, we setup traffic scenarios in CARLA and collect sensory data from multiple vehicles concurrently.

Specifically, we select two intersections in CARLA and simulate dense traffic scenarios by randomly spawning vehicles at the intersections. Vehicle are equipped with two depth sensors and two cameras, installing at the front and rear of the vehicles. Each camera and depth sensor has a  $100^\circ$  FoV and a maximum range of 30 meters by default, and they generate RGB and depth images at 30 fps with size  $1280 \times 720$ .

We evaluate the performance of our proposed algorithms with respect to different constraints, *i.e.*, selecting  $k$  cameras in *Max-Coverage*. Our algorithms are compared to a random selection algorithm that selects cameras randomly. To ensure fairness, we run the random selection algorithm 10 times for each simulation scenario to obtain averaged results.

### B. Case Study

We begin by illustrating the efficacy of our solution through a case study. Fig.14(a) depicts the traffic situation at an intersection during frame  $\mathcal{T}$ . A red boundary delineates the road area requiring coverage, with coverage calculations limited to within this boundary. This restriction is due to the fact that camera coverage outside the boundary does not contribute to obtaining traffic information. Occlusion caused by the boundary results in reduced coverage sectors for vehicle cameras, as sectors beyond the boundary are blocked. To assess our algorithm's performance, we employ the *coverage ratio*, defined as the area covered by selected cameras divided by the total area within the boundary. For simplicity, we may use *coverage* and *coverage ratio* interchangeably.

Within the boundary, 10 vehicles and several pedestrians are present. Each vehicle is equipped with a front and a rear camera, resulting in a total of 20 cameras. Notably, the pedestrian marked with the yellow box remains invisible to vehicle 1 due to being obscured by vehicle 2. Therefore, effective camera selection should aim to maximize coverage and ensure coverage of obscured areas. In this case study, we address the *Max-Coverage* problem. Assuming bandwidth supports the upload of images from  $k = 5$  cameras, we demonstrate that our *Max-Coverage* algorithm selects 5 cameras with much better coverage, providing comprehensive views of the

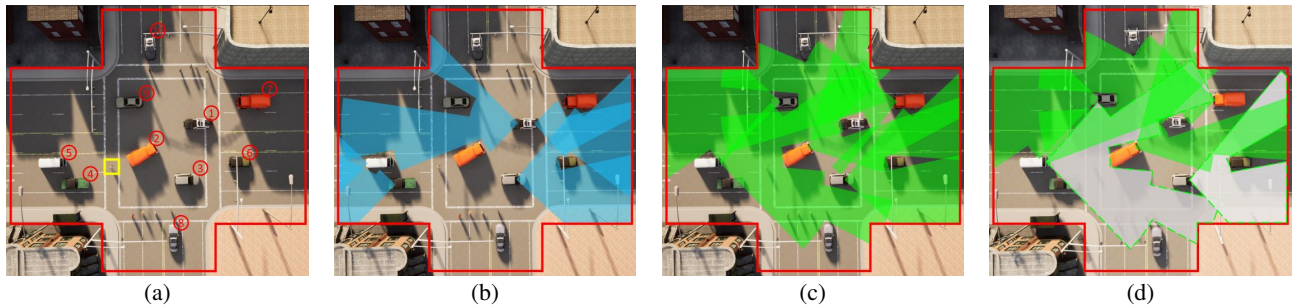
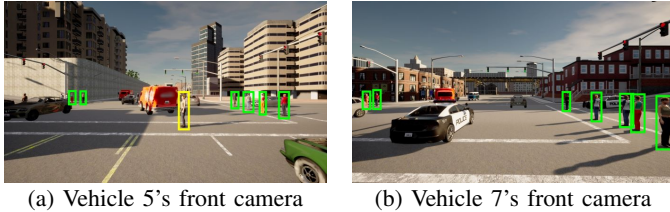


Fig. 14: (a) Case study: a scenario with 10 vehicles and 10 pedestrians in an intersection. (b) Random selection (five selected cameras achieve 43.1% coverage). (c) *Max-Coverage* selection (five selected cameras achieve 70.8% coverage). (d) Applying redundancy suppression in the next frame (saving 42.0% bandwidth in frame  $\mathcal{T} + 1$ ) to deal with vehicle movements.



(a) Vehicle 5's front camera (b) Vehicle 7's front camera  
Fig. 15: The images of two cameras that are selected by the *Max-Coverage* algorithm at frame  $\mathcal{T}$ .



Fig. 16: Vehicle 5's front camera at frame  $\mathcal{T} + 1$

intersection and covering vehicles and pedestrians as many as possible.

The random selection algorithm may not perform well. For instance, as depicted in Fig. 14 (b), the blue areas represent the coverage of five randomly selected cameras, which only covers 43.1% of the area. The crossing pedestrian behind vehicle 2 is not covered by any camera in the random selections, and hence the random selection algorithm fails to eliminate the risk of potential traffic accidents. In contrast, our *Max-Coverage* algorithm's selection, as shown in Fig.14(c) with green areas, covers 70.8% of the road area within the red boundary. This coverage is only 8% less than the optimal coverage achievable with all 20 cameras (78.5%). Moreover, the crossing pedestrian is captured by vehicle 5's front camera, enhancing safety measures.

For deeper insight into our system's effectiveness, Fig.15 presents the front camera views of vehicles 5 and 7 selected by our algorithm. These images offer diverse perspectives of the intersection, aiding in comprehensive traffic monitoring. The edge server can utilize these images to detect the 10 pedestrians delineated by bounding boxes. Notably, the previously mentioned crossing pedestrian (in Fig.14 (a)), highlighted in the yellow bounding box, is successfully identified in vehicle 5's front camera. Consequently, the edge server can issue a warning message to vehicle 1, mitigating potential accidents. In contrast, cameras selected by the random selection algorithm may fail to provide actionable information, resulting in resource wastage.

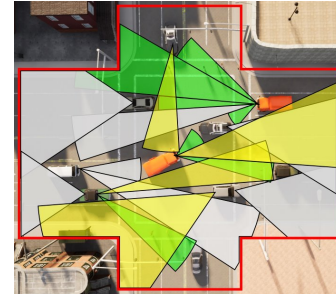


Fig. 17: *Redundancy-Aware Sector Selection* achieves 83.6% coverage.

We evaluate the computational performance of the *Max-Coverage* algorithm on a machine equipped with an Intel Core i7 3.80GHz and an NVIDIA RTX 3080. The mean computational time is 13 ms, comfortably supporting 30 fps. Additionally, we utilize a brute force algorithm to determine the optimal camera selection. With a complexity of  $O(n^k)$ , where  $n$  is the total number of cameras and  $k$  is the number of selected cameras (here,  $n = 20$  and  $k = 5$ ), the brute force algorithm requires 8.005 seconds for computation. However, its coverage reaches only 74.0%. Compared to the 70.8% coverage achieved by our *Max-Coverage* algorithm, this marginal increase of 3.2% is not significant.

**Dealing with vehicle movements:** Due to the dynamic nature of vehicle movements, our system must update the traffic conditions at a rate of 30 fps. However, consecutive images often contain significant redundancy, motivating us to propose redundancy suppression techniques to further reduce bandwidth consumption. As illustrated in Fig.14(c) and Fig.14(d), with a short time difference of 33 ms, vehicles do not move considerably, resulting in similar camera coverage. Through our redundancy suppression techniques, bandwidth consumption can be decreased by 42.0%. Specifically, the gray areas in Fig.14(d) denote redundant camera images (i.e., those from vehicle 3's and 5's front cameras).

To illustrate the redundancy further, Fig.16 presents the image captured by vehicle 5's front camera at frame  $\mathcal{T} + 1$ . Comparing it with the previous image (frame  $\mathcal{T}$  shown in Fig.15 (a)), the differences are negligible, with a pHash value of 0.92, exceeding the threshold  $\alpha = 0.9$ . Consequently, these two images are deemed similar, and the new image will not be uploaded to save bandwidth.

**Redundancy-Aware Sector Selection:** We show the result of applying *Redundancy-Aware Sector Selection* algorithm,

and this approach involves selecting sectors while considering the redundancy. In Fig. 17, the yellow sectors represent the newly selected sectors that are not selected before; the green sectors indicate that the sector images have less redundancy and are uploaded; the gray sectors represent the redundant sectors whose images are not uploaded.

In general, the *Redundancy-Aware Sector Selection* algorithm achieves a higher coverage (83.6%) compared to the camera-based selection (70.8% with  $k = 5$ ). Additionally, more cameras are involved and more sectors are selected, achieving larger overall coverage. This observation stems from the fact that redundancy-aware sector selection can utilize the available bandwidth more efficiently. Specifically, sectors with significant redundancy are not selected, and the corresponding images are not uploaded. This bandwidth reduction enables the uploading of other images that cover more areas. For example, images of gray sectors are not uploaded, and the saved bandwidth is reallocated to support the uploading of images of other newly selected sectors, i.e., yellow sectors. As a result, this better utilization of bandwidth contributes to an increase in the overall coverage.

### C. Simulation Results

We conduct extensive experiments to evaluate the performance of our algorithms at a larger scale. We spawn 30 vehicles at intersections to simulate the busy traffic. Since some of the vehicles may not participate in our system due to hardware limitations or other reasons, we also evaluate how the number of participating vehicles ( $p$ ) affects the performance.

1) *Results on Max-Coverage*: In this part, we evaluate the performance of our “Max-Coverage” algorithm and compare it with the random selection algorithm (“Random”). Since obtaining the optimal selections in practice is infeasible, we present the performance upper bound with the “Best Achievable” coverage. This metric represents the total coverage that can be achieved by utilizing all vehicle cameras.

We first study how the number of selected cameras ( $k$ ) affects the coverage. Fig. 18 (a) shows the coverage of different algorithms as a function of  $k$ , when there are 20 participating vehicles ( $p$ ), the camera range  $r$  is 30 meters, and the FoV is  $100^\circ$ . The figure illustrates that the coverage of both “Max-Coverage” and the “Random” increases as more cameras are selected. Importantly, “Max-Coverage” consistently outperforms the “Random”. Since each vehicle has two cameras, there are 40 cameras in total and the best achievable coverage is obtained with all these 40 cameras, which is 0.89. When  $k = 14$ , which means that 14 cameras are selected, “Max-Coverage” can achieve a comparable coverage to the best achievable while utilizing significantly less wireless bandwidth, i.e., transferring 14 camera images instead of 40.

Fig. 18 (b) shows the coverage when the number of participating vehicles ( $p$ ) changes. With an increase in  $p$ , there are more selection choices available in “Max-Coverage” and the “Best achievable”, leading to an increase in the coverage. For example, when  $p = 20$  (vs.  $p = 5$ ), “Max-Coverage” can select the best 8 cameras from these  $20 \times 2$  (vs.  $5 \times 2$ ) cameras to achieve a coverage of 0.82 (vs. 0.68), while the “Best achievable” will select all these  $20 \times 2$  (vs.  $5 \times 2$ ) cameras to achieve a

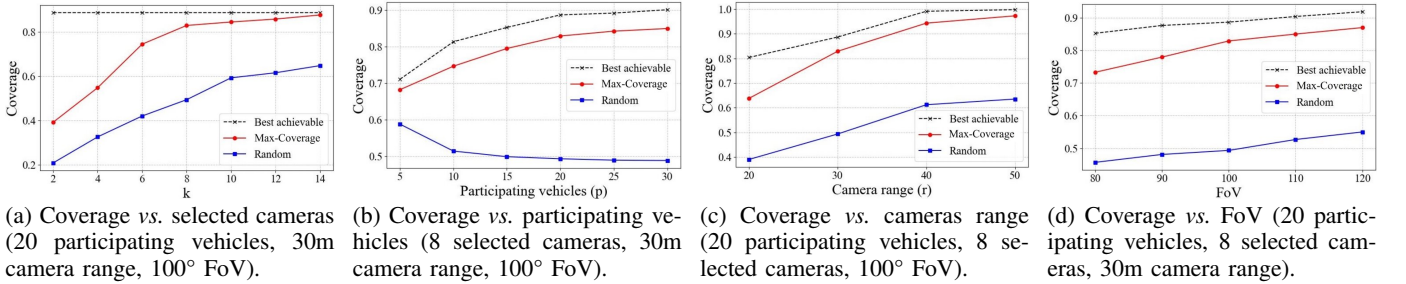
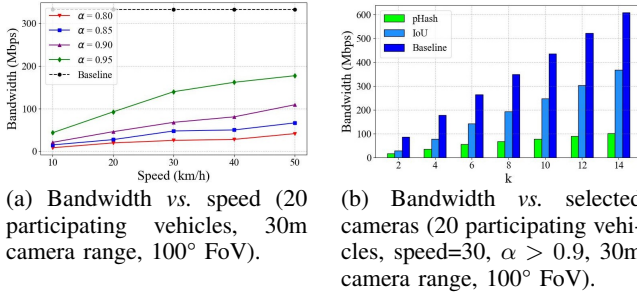
coverage of 0.89 (vs. 0.71). This demonstrates the adaptability of “Max-Coverage” in selecting cameras from a larger pool, resulting in improved coverage with an increasing number of participating vehicles. On the other hand, in “Random”, unpredictability and variability can be observed, resulting in fluctuating performance. For example, as the number of participating vehicles  $p$  increases from 5 to 10, the coverage is decreased, and this can be explained as follows. When  $p = 5$ , eight cameras are randomly selected from these  $5 \times 2$  cameras, and the likelihood of selecting cameras with better coverage is higher. However, when  $p = 10$ , eight cameras are randomly selected from these  $10 \times 2$  cameras, diminishing the chances of selecting cameras with better coverage. As the number of participating vehicles continues to increase, the coverage remains relatively stable since the probability of selecting cameras with good coverage does not change significantly.

We also study how camera specifications affect the coverage of selected cameras. In Fig. 18 (c), we compare the three approaches (“Max-Coverage”, “Random” and “Best achievable”) when the camera range changes, with  $k = 8$ ,  $p = 20$ , and FoV =  $100^\circ$ . The figure reveals that the coverage increases with the expansion of the camera range. Besides, “Max-Coverage” always outperforms the “Random” algorithm, and approaches to the “Best achievable” as the camera range increases. Next, in Fig. 18 (d), we fix the camera range as 30 meters,  $p = 20$ , and  $k = 8$ , and compare the coverage of these three approaches when the FoV changes from  $80^\circ$  to  $120^\circ$ . As shown in the figure, the coverage increases as the FoV increases. “Max-Coverage” always outperforms random selection, and comparable to the “Best achievable”.

**Redundancy Suppression:** The proposed similarity based redundancy suppression techniques only depend on the selected cameras, allowing for direct application to the *Max-Coverage* algorithm to filter out redundant images. For our study, we consider the case where  $k = 8$ , with a fixed camera range of 30 meters and a field of view (FoV) of  $100^\circ$ . In the “Baseline” scenario, cameras generate images at 30 frames per second (fps), resulting in the upload of  $8 \times 30$  images to the edge server per second.

Fig. 19 (a) evaluates the bandwidth requirement in pHash when the vehicle speed changes. An increase in vehicle speed results in rapid changes in the camera view, leading to lower similarity between camera images. Hence, more images will be uploaded, consuming additional bandwidth. The figure also illustrates that a higher threshold value  $\alpha$  correlates with increased bandwidth consumption. Notably, a pronounced increase in bandwidth is observed when  $\alpha$  is raised from 0.9 to 0.95. To strike a balance between keeping information up-to-date and achieving a substantial reduction in bandwidth, we set the threshold  $\alpha = 0.9$ . This choice ensures that we maintain current information while realizing a significant reduction in bandwidth consumption.

Next, we set the vehicle speed to 30 km/hour, which aligns with the average speed of vehicles in cities [27]. The performance of different similarity metrics, specifically IoU and pHash, is then compared to the baseline when varying numbers of cameras are selected. In general, applying the redundancy suppression technique demonstrates a notable reduction in

Fig. 18: Results on *Max-Coverage* problem.Fig. 19: Results on *Max-Coverage* with redundancy suppression.

bandwidth consumption, as shown in Fig. 19 (b). For instance, pHash reduces the bandwidth usage by 80% ~ 85% compared to the baseline, while pHash also significantly outperforms the other similarity metric, IoU, underlining its effectiveness in reducing bandwidth consumption.

2) *Results on Redundancy-Aware Sector Selection*: In this subsection, we evaluate the effectiveness of our *Redundancy-Aware Sector Selection* algorithm. We compare the performance with two other approaches, the “Camera-based” approach and the “Sector-based” approach. Specifically, the “Camera-based” approach is based on *Max-Coverage*, which uploads the entire images of selected cameras; The “Sector-based” approach is an extension of *Max-Coverage* which selects sectors. “Ours” represents the *Redundancy-Aware Sector Selection* algorithm. Throughout the evaluation, we maintain a fixed camera range of  $r = 30$  meters and a FoV of 100°. Also, we set the redundancy threshold  $\alpha = 0.9$ , ensuring that only images with significant changes are uploaded.

We first evaluate the overall coverage. In the “Camera-based” approach, the *Max-Coverage* algorithm selects  $k$  cameras, whereas in the “Sector-based” approach, the constraint of  $k$  cameras is not directly applicable. Since  $k$  is determined by the available bandwidth, we use bandwidth as the constraint. As shown in Fig. 20 (a), x-axis represents bandwidth constraints, and y-axis shows coverage. All approaches achieve larger coverage as bandwidth increases, and “Ours” can achieve the same coverage as the best achievable baseline when the bandwidth is higher than 30 Mbps. The “Sector-based” approach consistently outperforms the “Camera-based” approach since selecting sectors provides more flexibility to cover the road area. “Ours” further considers redundancy during sector selection, and thus achieves higher coverage than the other two approaches. This superiority stems from more efficient utilization of bandwidth, where the saved bandwidth is reallocated to enable the selection of additional sectors. As a

		Number of sectors					
		50	60	70	80	90	100
Convex	Time (ms)	34.4	55.7	63.4	85.1	147.9	169.9
	std	8.7	10.5	11.0	12.5	24.1	61.1
Ours	Time (ms)	1.4	1.7	1.8	2.0	2.8	3.9
	std	0.1	0.2	0.2	0.6	0.8	1.2

TABLE II: Processing time of sector selection.

result, in case of insufficient bandwidth, “Ours” demonstrates significantly higher efficiency. For instance, with a 10 Mbps bandwidth constraint, “Ours” achieves a coverage of 0.63, which is 12% and 26% higher than that achieved by “Sector-based” and “Camera-based”, respectively.

We also evaluate the coverage under varying numbers of participating vehicles, as shown in Fig. 20 (b). As the number of vehicles increases, the number of sectors grows, providing more selection options and resulting in improved coverage. Since all approaches aim to maximize the total coverage, their performance improves with more vehicles. However, “Ours” consistently outperforms others since it efficiently reallocates the saved bandwidth to enable the selection of additional sectors.

Fig. 20 (c) and (d) present the performance results with different camera specifications, specifically camera range and field of view (FoV). Larger camera ranges and wider FoVs increase the coverage of each sector, leading to an overall increase in total coverage for all approaches. Due to the superior bandwidth efficiency, “Ours” can select more sectors, achieving significantly better coverage compared to other approaches under the same constraints.

We also compare the percentage of selected sectors with the “Sector-based” approach to show the efficiency of our algorithm. In Fig. 21, we show the percent of sectors that are selected by different approaches with 30 Mbps bandwidth constraint. As redundant images are not uploaded, the *Redundancy-Aware Sector Selection* algorithm reallocates the saved bandwidth to select additional sectors, thus “Ours” selects more sectors than “Sector-based”, which does not consider redundancy during sector selection. The results show that “Ours” selects 80% sectors, representing a 37% improvement over “Sector-based”. This efficiency directly translates into improved coverage, with “Ours” achieving a coverage of 0.85 in Fig. 20, which is close to the best achievable performance.

Next, we evaluate the computational overhead of the *Redundancy-Aware Sector Selection* algorithm (*Ours*), and show the computation time as a function of the number of sectors. We compare our algorithm with a traditional convex optimization algorithm, *Convex*, provided in [28], and the results are shown in Table II. As expected, the processing

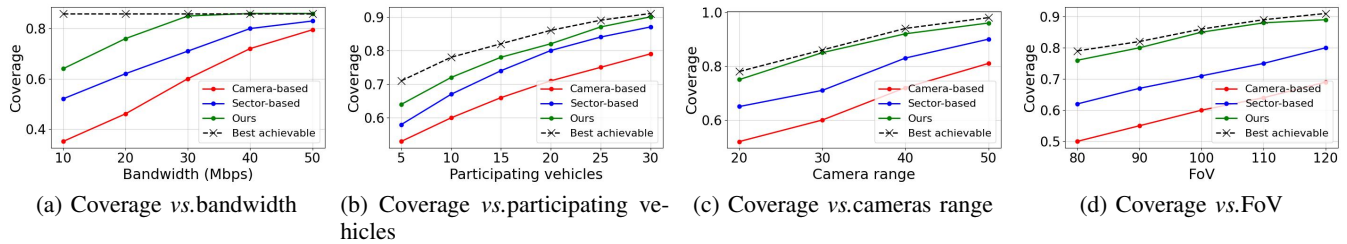
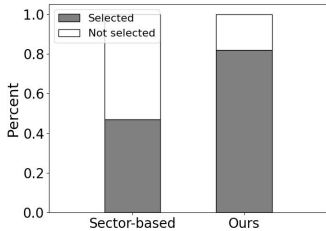
Fig. 20: Results on *Redundancy-Aware Sector Selection*.

Fig. 21: Percent of selected sectors.

time for both methods increases with the number of sectors due to the growing complexity of the selection problem and the increased overlap between sectors. However, *Convex* incurs a significantly higher computational cost, with processing times exceeding 33 ms, making it impractical for real-time applications. In contrast, our algorithm has significantly low computation time, which is less than 4 ms even for 100 sectors, ensuring that sufficient time remains for image uploading and processing in real-time operations. From the table, we also observe that the standard deviation increases as the number of sectors grows. This is because a higher number of sectors introduces redundancy, leading to variations in the computational load required to handle redundant images, resulting in fluctuations in computation time.

## VII. RELATED WORK

Our work intersects with the fields of cooperative perception in vehicular networks, mobile edge computing, and camera selection based on metadata.

**Cooperative Perception:** Considerable research has been devoted to cooperative perception in vehicular networks [29]–[33]. Qiu *et al.* [34] developed a system enabling the sharing of visual information among neighboring vehicles to detect objects blocked by preceding vehicles. Wang *et al.* [35] proposed broadcasting object features obtained by LiDAR to nearby vehicles to enhance environmental perception. Liu *et al.* [36] introduced a system facilitating the sharing of perception data among vehicles, leading to a collaborative synthesis of traffic information. Similarly, Liu *et al.* [2] presented a system for sharing camera images among vehicles, followed by object detection and tracking in various traffic scenarios. In [5], Wang *et al.* developed an edge-assisted relevance-aware perception dissemination system that collects perception data from multiple vehicles and selectively disseminates only the necessary data to appropriate vehicles. However, these efforts do not consider the overall coverage of the perception data, potentially leaving some areas uncovered and undetected.

**Mobile Edge Computing:** With the rise of mobile edge computing, extensive research has explored offloading com-

putational tasks to edge servers [6], [37]–[40]. In vehicular networks, edge servers are frequently utilized for information sharing and perception data processing. For instance, Du *et al.* [6] proposed offloading part of videos to edge servers to enhance detection accuracy using advanced deep neural networks. LiveMap [1] employs the edge server to collect and analyze camera images uploaded from vehicles, constructing a dynamic map of road traffic. LiveMap adaptively offloads object detection tasks to the edge server, optimizing bandwidth and computational resources. Offloading is scheduled among a few vehicles based on their camera coverage, although occlusions are not factored in. In [7], Zhang *et al.* proposed an edge-assisted system enabling multiple vehicles to share raw LiDAR sensor data with the edge server for perception processing. Different from existing approaches, our system utilizes depth images to detect and pinpoint occlusions. Consequently, we leverage camera metadata to represent these occlusions, enabling the edge server to make informed camera selection decisions. This innovative strategy effectively reduces overall bandwidth consumption.

**Camera Metadata and Selection:** Our work draws inspiration from previous studies on utilizing metadata to represent camera coverage. For instance, SmartPhoto [8] quantifies the quality of crowdsourced photos based on camera metadata, aiding in photo selection for covering target points. Wu *et al.* [9] extended this approach to select crowdsourced photos covering target areas under various resource constraints. VideoMec [10] employs metadata for selecting crowdsourced videos under bandwidth and energy constraints. However, none of these works consider object occlusions when utilizing camera metadata.

The camera selection problem addressed in this paper is related to the area coverage problems in sensor networks, extensively studied in prior research [41]–[45]. Huang *et al.* [41] investigated the  $k$ -coverage problem in sensor networks and proposed algorithms to determine whether an area is  $k$ -covered. He *et al.* [45] proposed approximation algorithms for selecting the minimum number of cameras to achieve full-view coverage. However, none of these studies consider occlusions or address challenges related to vehicle and camera movements.

In contrast, we leverage camera metadata to tackle area coverage problems, utilizing depth sensors to identify occlusions and applying metadata concepts for camera selection in vehicular networks. Additionally, we account for fast vehicle movements and introduce redundancy suppression techniques to conserve bandwidth.

## VIII. CONCLUSION

In this paper, we proposed an edge-assistant camera selection system which selects only the necessary camera images based on camera metadata, to reduce the bandwidth, storage and processing cost at the edge server. By leveraging depth sensors, we detect and locate occlusions, enabling an accurate representation of actual camera coverage by excluding blocked areas through camera metadata. Based on camera metadata, we developed an efficient algorithm to solve the *Max-Coverage* camera selection problem, which selects a limited number of cameras to maximize the total coverage. To reduce bandwidth consumption, we introduced redundancy suppression and sector-based selection techniques. However, we observed that the saved bandwidth often remains unused, leading to inefficient bandwidth utilization. To address this problem, we proposed a redundancy-aware sector selection algorithm that reallocates saved bandwidth to select additional sectors. Specifically, when an image is identified as redundant, the saved bandwidth is reallocated to other sectors, improving the overall bandwidth efficiency. Extensive evaluations demonstrate that the proposed algorithms effectively maximize coverage with bandwidth constraints while ensuring efficient resource utilization.

## IX. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grants 2125208 and 2215043.

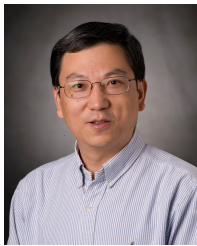
## REFERENCES

- [1] Q. Liu, T. Han, J. L. Xie, and B. Kim, "LiveMap: Real-Time Dynamic Map in Automotive Edge Computing," in *IEEE INFOCOM*, 2021.
- [2] L. Liu and M. Gruteser, "EdgeSharing: Edge Assisted Real-time Localization and Object Sharing in Urban Streets," in *IEEE INFOCOM*, 2021.
- [3] R. Wang and G. Cao, "Edge-Assisted Camera Selection in Vehicular Networks," in *IEEE INFOCOM*, 2024.
- [4] H. Qiu, P.-H. Huang, N. Asavisanu, X. Liu, K. Psounis, and R. Govindan, "AutoCast: Scalable Infrastructure-Less Cooperative Perception for Distributed Collaborative Driving," in *ACM MobiSys*, 2022.
- [5] R. Wang and G. Cao, "Edge-Assisted Relevance-Aware Perception Dissemination in Vehicular Networks," in *IEEE ICDCS*, 2024.
- [6] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *ACM SIGCOMM*, 2020.
- [7] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao, "EMP: Edge-assisted Multi-vehicle Perception," in *ACM MobiCom*, 2021.
- [8] Y. Wu, Y. Wang, W. Hu, and G. Cao, "SmartPhoto: A Resource-Aware Crowdsourcing Approach for Image Sensing with Smartphones," *IEEE Transactions on Mobile Computing*, 2016.
- [9] Y. Wu, Y. Wang, and G. Cao, "Photo Crowdsourcing for Area Coverage in Resource Constrained Environments," in *IEEE INFOCOM*, 2017.
- [10] Y. Wu and G. Cao, "VideoMec: A Metadata-enhanced crowdsourcing system for mobile videos," *IPSN*, 2017.
- [11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
- [12] "Jetson TX2 Module." <https://developer.nvidia.com/embedded/jetson-tx2/>.
- [13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *European Conference on Computer Vision (ECCV)*, 2016.
- [14] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, 1981.
- [15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, 2013.
- [16] "The KITTI Vision Benchmark Suite." [http://www.cvlibs.net/datasets/kitti/eval\\_object.php?obj\\_benchmark=2d](http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d).
- [17] D. S. Hochbaum, ed., *Approximation Algorithms for NP-Hard Problems*. USA: PWS Publishing Co., 1996.
- [18] G. Greiner and K. Hormann, "Efficient Clipping of Arbitrary Polygons," *ACM Trans. Graph.*, vol. 17, no. 2, p. 71–83, 1998.
- [19] R. Nürnberg, "Calculating the area and centroid of a polygon in 2d." <https://www.ma.imperial.ac.uk/~rn/centroid.pdf>, 2013.
- [20] L. Du, A. T. Ho, and R. Cong, "Perceptual hashing for image authentication: A survey," *Signal Processing: Image Communication*, vol. 81, p. 115713, 2020.
- [21] F. Khelifi and A. Bouridane, "Perceptual Video Hashing for Content Identification and Authentication," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 1, pp. 50–67, 2019.
- [22] C. Zauner, "Implementation and benchmarking of perceptual image hash functions." [https://www.phash.org/docs/pubs/thesis\\_zauber.pdf](https://www.phash.org/docs/pubs/thesis_zauber.pdf), 2010.
- [23] E. Altman, *Constrained Markov Decision Processes*. Chapman and Hall, 1999.
- [24] A. Gattami, Q. Bai, and V. Aggarwal, "Reinforcement Learning for Constrained Markov Decision Processes," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2021.
- [25] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2003.
- [26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.
- [27] "INRIX Global Traffic Scorecard." <https://inrix.com/scorecard/>.
- [28] S. Diamond and S. Boyd, "CVXPY: A Python-Embedded Modeling Language for Convex Optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
- [29] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-Cooper: Feature Based Cooperative Perception for Autonomous Vehicle Edge Computing System Using 3D Point Clouds," in *ACM/IEEE Symposium on Edge Computing*, 2019.
- [30] Q. Chen, S. Tang, Q. Yang, and S. Fu, "Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds," in *IEEE ICDCS*, 2019.
- [31] S. Kassar and G. de Veciana, "Opportunistic Collaborative Estimation for Vehicular Systems," in *IEEE INFOCOM*, 2023.
- [32] J. Zhao and G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," *IEEE INFOCOM*, 2006.
- [33] J. Zhao, Y. Zhang, and G. Cao, "Data Pouring and Buffering on the Road: A New Data Dissemination Paradigm for Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3266–3277, 2007.
- [34] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan, "AVR: Augmented Vehicular Reality," in *ACM MobiSys*, 2018.
- [35] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun, "V2VNet: Vehicle-to-vehicle communication for joint perception and prediction," in *European Conference on Computer Vision (ECCV)*, 2020.
- [36] H. Liu, P. Ren, S. Jain, M. Murad, M. Gruteser, and F. Bai, "FusionEye: Perception Sharing for Connected Vehicles and its Bandwidth-Accuracy Trade-offs," in *IEEE SECON*, 2019.
- [37] L. Liu, H. Li, and M. Gruteser, "Edge Assisted Real-time Object Detection for Mobile Augmented Reality," in *ACM MobiCom*, 2019.
- [38] P. Zhou, T. Braud, A. Zavodovski, Z. Liu, X. Chen, P. Hui, and J. Kangasharju, "Edge-Facilitated Augmented Vision in Vehicle-to-Everything Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 12187–12201, 2020.
- [39] F. Ahmad, H. Qiu, R. Eells, F. Bai, and R. Govindan, "CarMap: Fast 3d feature map updates for automobiles," in *Usenix Conference on Networked Systems Design and Implementation (NSDI)*, 2020.
- [40] J. Li, L. Liu, H. Xu, S. Wu, and C. J. Xue, "Cross-Camera Inference on the Constrained Edge," in *IEEE INFOCOM*, 2023.
- [41] C.-F. Huang and Y.-C. Tseng, "The Coverage Problem in a Wireless Sensor Network," in *ACM WSN*, 2003.
- [42] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Journal of Wireless Ad-hoc and Sensor Networks*, vol. 1, no. 1-2, pp. 89–124, 2005.
- [43] Y. Wang and G. Cao, "On full-view coverage in camera sensor networks," in *IEEE INFOCOM*, 2011.

- [44] Y. Wu and X. Wang, "Achieving Full View Coverage with Randomly-Deployed Heterogeneous Camera Sensors," in *IEEE ICDCS*, pp. 556–565, 2012.
- [45] S. He, D.-H. Shin, J. Zhang, J. Chen, and Y. Sun, "Full-View Area Coverage in Camera Sensor Networks: Dimension Reduction and Near-Optimal Solutions," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7448–7461, 2016.



**Ruiqi Wang** is a PhD candidate at Pennsylvania State University. He received the BE degree from University of Electronic Science and Technology of China, the MS degree in computer science from University of Delaware. His research interests include mobile computing, edge computing, and vehicular networks. He is a student member of the IEEE.



**Guohong Cao** received his B.S. degree in computer science from Xi'an Jiaotong University, and his Ph.D. in computer science from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Distinguished Professor. Dr. Cao directs the Mobile Computing and Networking Lab and co-directs the Institute for Networking and Security Research. He has published more than 250 papers in the areas of wireless networks, mobile computing,

machine learning, wireless security and privacy, and Internet of Things, which have been cited over 25000 times. He has served on the editorial board of *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Wireless Communications*, and *IEEE Transactions on Vehicular Technology*, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS, MASS, and INFOCOM. Dr. Cao has received several best paper awards, the IEEE INFOCOM Test of Time award, and the NSF CAREER award. He is a Fellow of the AAAS and a Fellow of the IEEE.