SPATIAL KEYFRAME EXTRACTION OF MOBILE VIDEOS FOR EFFICIENT OBJECT DETECTION AT THE EDGE

George Constantinou *Cyrus Shahabi* *Seon Ho Kim*

Integrated Media Systems Center
University of Southern California
Los Angeles, CA 90089

ABSTRACT

Advances in federated learning and edge computing advocate for deep learning models to run at edge devices for video analysis. However, the captured video frame rate is too high to be processed at the edge in real-time with a typical model such as CNN. Any approach to consecutively feed frames to the model compromises both the quality (by missing important frames) and the efficiency (by processing redundantly similar frames) of analysis. Focusing on outdoor urban videos, we utilize the spatial metadata of frames to select an optimal subset of frames that maximizes the coverage area of the footage. The *spatial keyframe* extraction is formulated as an optimization problem, with the number of selected frames as the restriction and the maximized coverage as the objective. We prove this problem is NP-hard and devise various heuristics to solve it efficiently. Our approach is shown to yield much better hit-ratio than conventional ones.

Index Terms— spatial keyframe extraction, weighted coverage, spatial metadata, object detection

1. INTRODUCTION

Recent advancements in processing power and explosive growth of IoT devices (20.4 billion by 2020 [1]) enabled the continuous development of edge computing (EC) systems. With EC paradigm, the processing cost of information is offloaded close to the edge devices, where data is generated. Shifting computation to the edge has several benefits: a) applications do not suffer from latency and communication bandwidth restrictions because they are able to process and store data locally in real-time, b) it reduces the costs of processing the collected data at a cloud-based or centralized server, and c) it provides a privacy mechanism so that raw sensitive data (e.g., fingerprints) does not need to be directly shared outside the device.

In addition to the wide applicability of EC, the improvements in neural networks have made it possible to run deep learning models locally for classification or object detection tasks. Image-based Machine Learning (ML) applications for smart cities already made their appearance, e.g., detect road damage [2, 3] to improve the infrastructure, monitor street cleanliness [4] to prioritize sanitation efforts, detect surface for graffiti removal [5, 6] to improve quality of life and reduce gang crime, and bicycle/pedestrians counters [7, 8, 9] to improve transportation, to name a few.

Cameras embedded on edge devices are nowadays able to capture videos at a very high frame rate (e.g., 24-60 FPS). The only way for a Convolutional Neural Network (CNN) (e.g., YOLOv3 [10], Caffe [11]) to process such visual data volume at this rate is by using powerful CPUs and GPUs which cannot be found on edge devices. At the edge, frame acquisition rate is much higher from what a deployed ML model can handle, hence only a subset of frames can be processed in real-time. *Then, how do we decide which frames to feed to a CNN?* Naively, applications use the CNN processing rate after sampling some frames, i.e., feed the newly captured frame to the CNN as soon as the processing of an older frame is completed, which can result in missing important visual content, especially when using a mobile device.

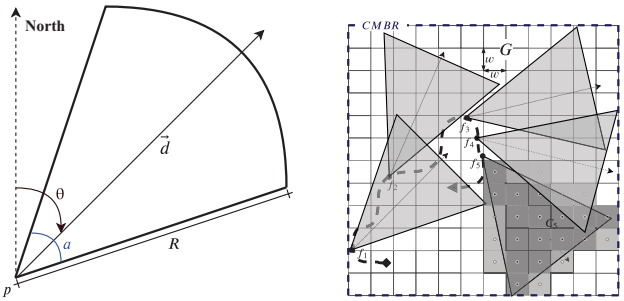
In this work, we focus on efficient frame selection from urban mobile videos while considering the limited resources of edge devices in image ML applications. Instead of directly analyzing the raw video frames which requires lots of computing power, our algorithms leverage the geospatial metadata of images (captured at recording time by devices' sensors) to reduce image processing cost by maximizing the spatial coverage while minimizing the number of selected frames. In particular, we partition an area of interest into grid cells and use the number of unique cells covered by each frame (using its geospatial metadata) as the preference criteria to select or drop a frame. We formally define this coverage problem, prove its NP-hardness, and propose heuristics to efficiently solve it. We compare our heuristics with traditional keyframe selections based on visual content. Experimental results using real dataset show our hit-ratio is at least 25% better than conventional approaches.

2. RELATED WORK

Video frame extraction can be classified in three broader categories based on the methods used to extract frames: 1) visual, 2) machine learning, and 3) spatial metadata analysis.

Visual-based Video Frame Extraction: Dimitrova *et al.* [12] presented a method to compare the luminance (Y) and chrominance (Cr and Cb) of sequential video frames in order to identify significant differences and then used image histograms to filter keyframes. Ejaz *et al.* [13] proposed a strategy that initially samples frames with a skip factor λ. Afterwards it calculates a *contributing value* of consecutive frames based on color and structural properties, and extract frames with contributing value larger than a threshold τ.

ML-based Video Frame Extraction: The first ML approach to extract video frames utilized unsupervised clustering-based techniques. The main idea is to create clusters of visually similar images based on image content (e.g., color histogram) and select a frame from each cluster based on a similarity metric (e.g., frame closer to the cluster mean). De Avila *et al.* [14] presented *VSUMM* (Video SUMMarization) mechanism which uses pre-sampling to reduce the number of frames, constructs a 16-bin histogram from the HSV color space to obtain the feature vector and uses k-means algorithm to extract keyframes by selecting the frame closer to the cluster's



(a) The 2-D FOV model. (b) 5 FOVs and their $CMBR$.

Fig. 1: Field-Of-View (FOV) model and Coverage MBR.

centroid. Wu *et al.* [15] presented AdaFrame, an ML framework that adaptively selects frames for video recognition. AdaFrame contains a LSTM network augmented with a global memory that provides context information for searching which frames to use over time.

Spatial Metadata-based Video Frame Extraction: Kim *et al.* [16] exploits the spatial metadata of frames to automatically generate panoramic images from crowdsourced mobile videos. Each video frame is augmented with a field-of-view [17] to calculate the coverage and overlap ratio.

3. SPATIAL KEYFRAME EXTRACTION

3.1. Preliminaries

Definition 3.1. *Field-Of-View:* A video v is represented as a set of individual video frames $\mathcal{F} = \{f_1, f_2, \ldots, f_i, \ldots, f_n\}$ ordered by the time t_i at which the frame was captured. We use the Field-Of-View (FOV) model [17] as shown in Figure 1a to represent the coverage of the viewable scene of f_i. Hereafter, we denote with f_i the video frame and its FOV, interchangeably. The FOV f_i is in the form of $\langle p, \theta, R, \alpha \rangle$, where p is the camera position consisting of the latitude and longitude coordinates read from the GPS sensor in a mobile device, $0^\circ \leq \theta < 360^\circ$ is the angle of the camera viewing direction with respect to the North obtained from the digital compass sensor, R is the maximum visible distance, and $0^\circ < \alpha < 360^\circ$ denotes the visible angle obtained from the camera lens property at the current zoom level. We use the dot notation to access properties, i.e., $f_i.p$ refers to the camera point of the FOV f_i.

Definition 3.2. *Coverage Minimum Bounding Rectangle:* Given a set of FOVs $\mathcal{F}$, the Coverage Minimum Bounding Rectangle $CMBR$ is defined as the minimum bounding box which contains all FOVs[1] as illustrated in Figure 1b.

Definition 3.3. *Coverage Grid and Cell Set:* Given a $CMBR$ and cell size w, we partition the $CMBR$ into a set of square cells $\mathcal{G} = \{c_1, c_2, \ldots, c_m\}$ of width w forming the Coverage Grid. Given a set of FOVs $\mathcal{F}$ and the grid $\mathcal{G}$, the Coverage Cell Set $\mathcal{C} \subseteq \mathcal{G}$ contains all the cells which are covered by at least one FOV. In addition, we define with $\mathcal{C}_i \subseteq \mathcal{C}$, the subset of the cells which are covered by f_i. For example, in Figure 1b, the coverage cell set $\mathcal{C}_5$ for f_5 is highlighted in dark grey color. For simplicity, the cell is considered as covered if any FOV covers its center. The set $\mathcal{C}_f = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ contains all such subsets. Hence, $\bigcup_{i=1}^{n} \mathcal{C}_i = \mathcal{C}$. Symmetrically, we define with $\mathcal{F}_j \subseteq \mathcal{F}$ the subset of FOVs which cover cell $c_j \in \mathcal{C}$. The set $\mathcal{F}_c = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m\}$ contains all such subsets. Thus, $\bigcup_{j=1}^{m} \mathcal{F}_j = \mathcal{F}$.

3.2. Proposed Algorithms

User-generated urban videos contain a lot of overlapping regions. Even if the camera is moving, due to the variations in direction, a cell can be covered by multiple FOVs. To model the importance of regions and FOVs, we introduce a cell spatial weight which depends on the FOV that covers them.

Cell Spatial Weight: The spatial weight of an FOV f_i and a cell $c_j \in \mathcal{C}_i$ is defined as:

$$w_{i,j} = \begin{cases} 1 - \frac{d(f_i.p, c_j.p)}{f_i.R}, & \text{if } d(f_i.p, c_j.p) \leq f_i.R \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The function $d(f_i.p, c_j.p)$ calculates the distance[2] between the camera location $f_i.p$ and the cell center $c_j.p$. The distance is then normalized by the maximum viewable distance $f_i.R$. Cells closer to camera location are assigned higher weight, whereas cells outside the FOV's coverage area are assigned zero weight, i.e., $d(f_i.p, c_j.p) > f_i.R$. This ensures that the weight is $0 \leq w_{i,j} \leq 1$.

Cell Overlap Weight Function: We define a function $f : X \rightarrow Y$, where $X = \{x \in \mathbb{R}^{|\mathcal{F}'_j|} | x = w_{i,j}, f_i \in \mathcal{F}'_j \subseteq \mathcal{F}_j, c_j \in \mathcal{C}_i\}$ and $Y = \{y \in \mathbb{R} | 0 \leq y \leq 1\}$. The function f accepts as input the weights $X = w_{i,1}, w_{i,2}, \ldots, w_{i,|\mathcal{F}'_j|}$ for cell c_j for the selected FOVs $\mathcal{F}'_j$ which cover it, and assigns a new weight Y. f defines what is the new spatial weight of the cell when multiple FOVs cover it and are selected by the solution. This function is application-specific. For example, a new weight can be the average sum of all weights contributed by the selected FOVs.

Residual Overlap Weight: For a current frame selection S, the residual overlap weight for f_i and its cells C_i is computed as follows: for cells $c_j \in C_i$ not covered by any other FOV already in $\mathcal{S}$, the residual weight $w^r_{i,j}$ is equal to $w_{i,j}$. Otherwise, we use the Cell Overlap Weight Function f to calculate a new overlap weight $w^o_{i,j}$ if $\mathcal{S}$ is to include f_i and calculate the difference with the current selection, i.e., $w^r_{i,j} = w^o_{i,j} - w_{i,j}$. The intuition is when a new FOV is added, it increases the total weight by only the weight difference.

Problem 1. *Maximum Weighted Overlap Coverage Problem:* Given a set of FOVs $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, the weights $w_{i,j}$, the cell overlap weight function f, the set of covered cells $\mathcal{C}_j = \{c_1, c_2, \ldots, c_m\}$ for each FOV, the maximum budget for frames B, the Maximum Weighted Overlap Coverage Problem (MWOCP) finds a subset $\mathcal{F}'$, s.t. $|\mathcal{F}'| \leq B$ which maximizes the weighted sum of covered cells in the sets $\mathcal{F}'_j$.

Theorem 1. The MWOCP is NP-Hard.

Proof. The proof for $MWOCP$'s hardness comes from the reduction of the Maximum Coverage Problem (MCP), i.e., $MCP \leq_p MWOCP$. Given a finite set of elements, called the universe $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$, a collection of sets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}, \mathcal{S}_i \subseteq \mathcal{X}$, whose union equals the universe, i.e., $\bigcup_{i=1}^{n} \mathcal{S}_i = \mathcal{X}$, and a budget value k the Maximum Coverage is the problem of finding a subset of $\mathcal{S}' \subseteq \mathcal{S}$, s.t., $|\mathcal{S}'| \leq k$ and $|\bigcup_{\mathcal{S}_i \in \mathcal{S}'} \mathcal{S}_i|$ is maximized.

For any MCP instance, we reduce it to an instance of $MWOCP$ in polynomial time. The reduction is straight-forward. The number of maximum frames k is passed as input budget B to the $MWOCP$. For each element x_j in the MCP, we create a cell c_j in the $MWOCP$, hence $\mathcal{X} = \mathcal{G}$. This construction takes $\mathcal{O}(m)$ time. Additionally, each set $\mathcal{S}_i$ in MCP is mapped to an FOV

[1] To reduce the computational complexity of the pie slice shape [17], we represent the FOVs as triangles.

[2] There are several distance approximation functions between points on Earth surface: euclidean distance [18], haversine or Vincenty's [19] formula.

Algorithm 1 Greedy-SKE

```
1: procedure GREEDY(F, C, W, f, B)
2:     S = ∅                                              ▷ Solution set
3:     U = C                                              ▷ Cell Universe
4:     F' = F                                             ▷ Candidate FOVs
5:     while |S| ≤ B do                                   ▷ Up to B FOVs
6:         bestFOV = bestCellSet = null, bestWeight = 0
7:         for all f_i ∈ F' do
8:             FOVCells = getFOVCells(f)                  ▷ Cell Set C_i
9:             uncoveredCells = FOVCells ∩ U
10:            weight = computeWeight(f, uncoveredCells)
11:            coveredCells = FOVCells \ U
12:            if coveredCells ≠ ∅ then
13:                weight = computeResidual(f, S, coveredCells)
14:            end if
15:            if weight > bestWeight then
16:                bestFOV = f
17:                bestCellSet = FOVCells
18:                bestWeight = weight
19:            end if
20:        end for
21:        if bestFOV == null then break
22:        end if
23:        S = S ∪ bestFOV
24:        F' = F' \ bestFOV
25:        U = U \ bestCellSet
26:    end while
27:    return S                                           ▷ The greedy solution to MWOCP
28: end procedure
```

Coverage Set $\mathcal{C}_i$ in $MWOCP$, assigning a weight $w_{i,j} = 1$ when element $x_j \in \mathcal{S}_i$, and $w_{i,j} = 0$ otherwise. The cell weight function f is deliberately set to return 1 when c_j (thus x_j) is not part of the current solution, otherwise 0 to prevent double-counting an element. This mapping takes $\mathcal{O}(nm)$ time. The construction ensures that elements covered by $\mathcal{S}_i$ are represented as covered cells by FOV f_i in $\mathcal{C}_i$ ($\mathcal{S} = \mathcal{C}_f$). Therefore, $MWOCP$'s output of FOVs is exactly $\mathcal{S}'$ in MCP, which completes the proof. □

As $MWOCP$ is NP-Hard, we propose a polynomial-time greedy algorithm to solve it efficiently. Our approach extends the Generalized Maximum Coverage Problem [20] (GMCP), to support overlaps (an element can belong to multiple bins).

Spatial Keyframe Extraction (SKE) algorithm: Algorithm 1 shows our proposed greedy algorithm **Greedy-SKE** to solve the MWOCP. Lines 2-4 initialize the solution set, the cell universe and the list of candidate FOVs $\mathcal{F}'$. The algorithm iterates until the budget of frames B (Line 5) is exhausted or no new FOV contributes to the total weight of the solution (Line 21). At each step of the main loop, the greedy algorithm tries to find the best FOV which has a higher weight and updates the current best at Lines 15-19. For each candidate FOV f, the total weight is measured as the sum of weights from uncovered and covered cells. Uncovered cells are the cells which are not covered by any FOV in the current solution $\mathcal{S}$, while covered cells are the cells which have at least one FOV in the existing solution $\mathcal{S}$. For uncovered cells, the weight is calculated as described in Section 3.2. For covered cells, with the help of a subroutine $computeResidual$, we calculate the residual overlap weight. When an FOV is selected, it is added to the current solution S, is removed from candidate frame set $\mathcal{F}'$ and the cell universe U of uncovered cells is updated at Lines 23-25.

3.3. Baselines for Comparison

Using Visual Features: A straightforward method of selecting distinct video frames is by comparing visual features. **Clustering**-based extraction utilizes VSUMM's [14] technique. Initially, a set of frames are sampled every half-second. Then, a histogram of 50 bins is constructed from the HSV color space (20-20-10 bins for each component, respectively) for each frame image which are used as the feature vectors. Finally, given a budget value B as the maximum number of frames that the algorithm can extract from each video, the k-means algorithm is applied to select the closest frame from each cluster centroid.

Using Frame FOV metadata: The previous baseline requires to analyze image content in order to decide whether there are significant visual changes between frames. This is a chicken-egg problem because the whole goal is to avoid analyzing all frames while detecting objects within a video. Approaches that require visual analysis are expensive both in terms of processing time and battery consumption, especially when analyzing video at the edge. The techniques introduced here do not require to *see* the image frame content. Instead, they solely rely on the spatial metadata captured along with the video.

Temporal selects frames based on a predefined sampling rate t_s (e.g., 2 frames per second). t_s can be adjusted in a way, such that it matches the processing capacity of an edge device. This method can be fast, but it may introduce unnecessary redundant frame processing in case of fast sampling or capturing similar frames when the camera is slowly moving.

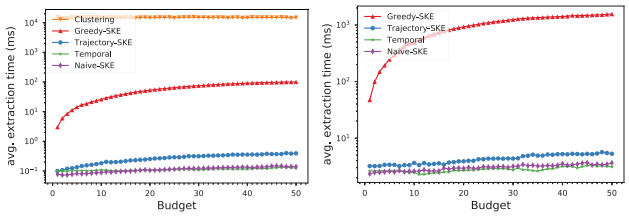
Trajectory-SKE selects frames based on the camera location of the FOV metadata. For each frame of a video, sorted by captured time, a predefined radius threshold t_r is used to determine whether the camera location of the frame is farther than the previous frame's radius. Sampling the trajectory by adjusting t_r ensures that the selected frames are captured at different locations, hence partially addresses the problem of processing redundant content of frames. However, it may still process duplicate frames when t_r is very small or the FOV cover a particular region.

Naive-SKE uses a max-heap to get cells in order based on their cumulative spatial weight of all FOVs. For the current cell c_j, a random FOV $f_i \in \mathcal{F}_j$ is selected and added to the solution $\mathcal{S}$. Additionally, all cells $c_l \in \mathcal{C}_i$ are removed from the heap. The algorithm stops when B is reached or the heap is emptied.

4. EXPERIMENTS

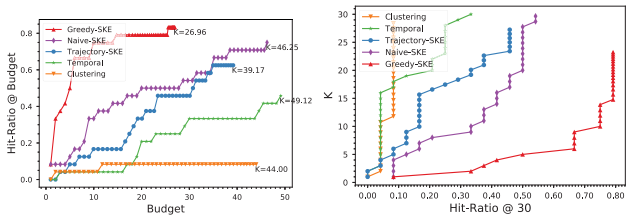
To demonstrate the effectiveness of our proposed techniques, we have conducted experiments on a real dataset. With MediaQ mobile app [21], we collected 25 FHD videos at 30 FPS generating 69K frames (2872 frames per video on average) along with their FOV metadata. All videos recorded so that they intentionally contain frames that capture a Starbucks coffee shop. The experimental setup is how to efficiently detect Starbucks logos from the collected videos using the discussed frame selection approaches. For actual logo detection, we used Google Vision API to detect the Starbucks logo for each frame in every video and log the detected frames with a confidence $\geq$ 70%, resulting to 5.5K frames with the detected logo. Experiments were performed on two types of devices: a powerful Ubuntu OS 16.04 equipped with a Quad Core Intel(R) Xeon(R) CPU E3-1240 v5 at 3.50GHz and 64GB of RAM and a less capable Raspberry Pi (RPI) 3 Model B device.

In our analysis, the *budget* B is the maximum number of frames an algorithm can extract from each video and is given as an input parameter. It is application/device specific value based on the constrained resources on edge device such as power consumption and data transfer bandwidth capacity [22]. The actual number of extracted frames is denoted by $K \leq B$. Note that it is possible that



(a) Desktop performance. (b) Raspberry Pi performance.

Fig. 2: Avg. Extraction Time per Video.



(a) Hit-Ratio @ Budget. (b) Frames for Hit-Ratio@30.

Fig. 3: Hit-Ratio performance.

$K < B$ because for **Greedy-SKE** the weighted overlap coverage is reached and adding a new FOV does not increase the total weight, or for the baselines the sampling rate results in fewer frames for short videos. We set the grid width $w = 15$ meters (similar trends were observed when we varied the width $w \in \{5, 10, 15, 20, 25\}$, hence results are omitted). Additionally, for **Temporal** we fixed $t_s = 500ms$ and for **Trajectory-SKE**, we empirically found that the sampling radius $t_r = 2m$ performs best, to select enough frames given the length of the videos. **Greedy-SKE** uses haversine as distance function $d(f_i.p, c_j.p)$ to calculate the spatial weight (results were similar with euclidean distance). After testing both avg and max, we chose max (performs slightly better) as the residual overlap function f, which assigns the maximum weight to the cell contributed by the selected FOVs that cover it.

4.1. Performance

Figure 2 shows the logarithmic scale computation time in msec of the various approaches while we vary B on both desktop (Figure 2a) and RPI (Figure 2b). The **Clustering** approach suffers the most (i.e., 2+ orders of magnitude slower than others) because of the time complexity to create the frame's histogram bins and to run k-means algorithm. ML-based frame extraction techniques, due to the processing cost, are not suitable to run on edge devices and is omitted from Figure 2b. The techniques share the same patterns on both the desktop and the RPI. The RPI, due to its limited resources, requires 1 order of magnitude more time. For example, **Greedy-SKE** needs 1sec to select 50 frames on RPI, compared to 100msec on the desktop, which is practical, since the majority of edge devices are not able to process frames at 50 FPS. **Greedy-SKE** requires to keep track of the residual weight each time a new FOV is selected; however, it linearly scales with the maximum number of frames B. **Temporal** and **Naive-SKE** are extremely fast. **Temporal** solely relies on the temporal information of each frame (i.e., extracts a frame per half second). **Naive-SKE** efficiently uses a max-heap to pick the next uncovered cell with the highest total weight and chooses any FOV that covers it. **Trajectory-SKE** is slightly slower than **Temporal** and **Naive-SKE** due to the calculation of the haversine distance to identify whether a new FOV was already sampled within t_r.

4.2. Impact of Spatial Weight

To demonstrate the effectiveness of assigning spatial weight to each cell, we constructed a field experiment for automated logo visibility analysis. We compare the selected frames by the approaches to the actual frames which contain Starbucks logo. If at least one of the selected frames contain the logo it is considered as a hit.

Figure 3a depicts the hit-ratio while varying $B \in \{1, 2 \ldots 50\}$. The graph plots the average selected frames K. Notice that $K < B$. For example, the **Greedy-SKE** at $K = 26.96$ frames it saturates, i.e., no other frame adds any value to the total weight, so it stops. Although, the **Naive-SKE** algorithm is fast to execute, it is inferior to **Greedy-SKE**. The **Naive-SKE** considers the cell total weight contributed from all FOVs, whereas **Greedy-SKE** smartly adjusts the overlapping cell weight while new frames are selected. The **Trajectory-SKE**, due to t_r, requires more frames to reach the same hit-ratio. **Greedy-SKE** does not require any user input and is able to reach 80% with only 15 frames (compared to 41% and 18% for **Naive-SKE** and **Trajectory-SKE**, respectively). **Trajectory-SKE** has the limitation that it needs to set a t_r such that the whole trajectory is sampled, which is not a realistic assumption when the video length is unknown.

Figure 3b illustrates the output number of selected frames K (y-axis) while varying the target hit-ratio at a fixed budget $B = 30$ (x-axis), which is reasonable given the average length of our video dataset. Our **Greedy-SKE** algorithm outperforms all other approaches. It is able to detect the logo in 66% of the videos with 6 frames in 17ms, 75% of the videos with 10 frames in 26ms, and reaches 80% with 15 frames in 39ms. None of the other approaches reach beyond 54%, despite that all videos contain frames which capture the Starbucks logo. **Naive-SKE** requires 4 times the number frames of **Greedy-SKE** (20 vs 5, respectively) to reach a hit-ratio of 50%. Despite consuming more time, **Clustering** does not achieve high hit-ratio in urban videos because it is unaware of the coverage area. Frames that are spatially close, are not necessarily visually close due to background changes and directional differences. **Temporal** yields worse results than **Trajectory-SKE** because it selects near-duplicate or miss important frames due to the sampling rate t_s. The **Greedy-SKE** considers the overlap coverage area due to the effect of the spatial weight and selects the optimal frames well ahead of any other technique.

5. CONCLUSION

We presented a variety of approaches to extract frames from urban mobile videos to detect objects. The **Greedy-SKE** highlights the importance of using spatial metadata and optimizes the frame selection based on the coverage area of the video. Experiments show that it outperforms other approaches by achieving a higher hit-ratio with less number of selected keyframes in a short amount of time. **Greedy-SKE** can be applied in many domains; for example, to enhance the situation awareness in case of disasters [23].

Acknowledgement

This research has been supported in part by the USC Integrated Media Systems Center, the Annenberg Foundation, and unrestricted cash gifts from Google and Oracle.

6. REFERENCES

[1] Gartner, "Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016," 2017.

[2] Hiroya Maeda, Yoshihide Sekimoto, Toshikazu Seto, Takehiro Kashiyama, and Hiroshi Omata, "Road damage detection and classification using deep neural networks with smartphone images," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 12, pp. 1127–1141, 2018.

[3] A. Alfarrarjeh, D. Trivedi, S. H. Kim, and C. Shahabi, "A deep learning approach for road damage detection from smartphone images," in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, Dec 2018, pp. 5201–5204, IEEE.

[4] A. Alfarrarjeh, S. H. Kim, S. Agrawal, M. Ashok, S. Y. Kim, and C. Shahabi, "Image classification to determine the level of street cleanliness: A case study," in *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*, Xi'an, China, Sep. 2018, pp. 1–5, IEEE.

[5] E. K. Tokuda, R. M. Cesar, and C. T. Silva, "Quantifying the presence of graffiti in urban environments," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Kyoto, Japan, Feb 2019, pp. 1–4, IEEE.

[6] A. Alfarrarjeh, D. Trivedi, S. H. Kim, H. Park, C. Huang, and C. Shahabi, "Recognizing material of a covered object: A case study with graffiti," in *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, Sep. 2019, pp. 2491–2495, IEEE.

[7] G. Somasundaram, V. Morellas, and N. Papanikolopoulos, "Counting pedestrians and bicycles in traffic scenes," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, St. Louis, MO, USA, Oct 2009, pp. 1–6, IEEE.

[8] M. K. Kocamaz, J. Gong, and B. R. Pires, "Vision-based counting of pedestrians and cyclists," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Placid, NY, USA, March 2016, pp. 1–8, IEEE.

[9] Jie Shen, Xin Xiong, Zhiyuan Xue, and Yinglong Bian, "A convolutional neural-network-based pedestrian counting model for various crowded scenes," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 10, pp. 897–914, 2019.

[10] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," 2018.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, New York, NY, USA, 2014, MM '14, pp. 675–678, ACM.

[12] Nevenka Dimitrova, Thomas McGee, and Herman Elenbaas, "Video keyframe extraction and filtering: A keyframe is not a keyframe to everyone," in *Proceedings of the Sixth International Conference on Information and Knowledge Management*, New York, NY, USA, 1997, CIKM '97, pp. 113–120, ACM.

[13] Naveed Ejaz, Tayyab Bin Tariq, and Sung Wook Baik, "Adaptive key frame extraction for video summarization using an aggregation mechanism," *J. Vis. Comun. Image Represent.*, vol. 23, no. 7, pp. 1031–1040, Oct. 2012.

[14] Sandra Eliza Fontes de Avila, Ana Paula Brandão Lopes, Antonio da Luz, and Arnaldo de Albuquerque Araújo, "Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method," *Pattern Recognition Letters*, vol. 32, no. 1, pp. 56 – 68, 2011, Image Processing, Computer Vision and Pattern Recognition in Latin America.

[15] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S. Davis, "Adaframe: Adaptive frame selection for fast video recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, IEEE.

[16] Seon Ho Kim, Ying Lu, Junyuan Shi, Abdullah Alfarrarjeh, Cyrus Shahabi, Guanfeng Wang, and Roger Zimmermann, "Key frame selection algorithms for automatic generation of panoramic images from crowdsourced geo-tagged videos," in *Web and Wireless Geographical Information Systems*, Dieter Pfoser and Ki-Joune Li, Eds., Berlin, Heidelberg, 2014, pp. 67–84, Springer Berlin Heidelberg.

[17] Sakire Arslan Ay, Roger Zimmermann, and Seon Ho Kim, "Viewable scene modeling for geospatial video search," in *Proceedings of the 16th ACM International Conference on Multimedia*, New York, NY, USA, 2008, MM '08, pp. 309–318, ACM.

[18] John P Snyder, *Flattening the earth: two thousand years of map projections*, University of Chicago Press, Chicago, USA, 1997.

[19] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey Review*, vol. 23, no. 176, pp. 88–93, 1975.

[20] Reuven Cohen and Liran Katzir, "The generalized maximum coverage problem," *Information Processing Letters*, vol. 108, no. 1, pp. 15 – 22, 2008.

[21] Seon Ho Kim, Ying Lu, Giorgos Constantinou, Cyrus Shahabi, Guanfeng Wang, and Roger Zimmermann, "Mediaq: Mobile multimedia management system," in *Proceedings of the 5th ACM Multimedia Systems Conference*, New York, NY, USA, 2014, MMSys '14, p. 224–235, Association for Computing Machinery.

[22] G. Constantinou, G. Sankar Ramachandran, A. Alfarrarjeh, S. H. Kim, B. Krishnamachari, and C. Shahabi, "A crowd-based image learning framework using edge computing for smart city applications," in *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, Singapore, Sep. 2019, pp. 11–20, IEEE.

[23] H. To, S. H. Kim, and C. Shahabi, "Effectively crowdsourcing the acquisition and analysis of visual data for disaster response," in *2015 IEEE International Conference on Big Data (Big Data)*, Oct 2015, pp. 697–706.