Scene Cache for Video Capture Data Reduction

Benjamin Simpson and Yuchen Liu
{bssimps, lyuchen} @umich.edu

Abstract

Continuous machine vision systems have a wide range of applications but high power consumption hinders wide adoption. This work tries to reduce power consumption of machine vision tasks without hardware or application redesign. We propose scene cache, a system that reduces power consumption during image acquisition phase. It exploits temporal redundancy between frames by only updating blocks that are different from the previous frame. It exploits spatial redundancy by row/column skipping (decimation) before determining regions of difference. Tests show that the scene cache can achieve 88% accuracy with 4% data transferred.

1 Introduction

Machine vision systems are key components of many important current and upcoming technologies such as facial recognition and autonomous vehicles. These systems are not highly optimized for low-power consumption, limiting their potential applications in mobile, battery-powered devices. Biological sensing systems, however, are highly optimized to reduce energy consumption [1].

The power consumption in machine vision systems is in part proportional to the number of pixels sensed. Current image processing pipelines are designed to capture high resolution data in all areas of an image. The human vision system, however, uses spatially varying resolution, with high resolution concentrated on a small area in the center known as the fovea and low resolution in the periphery. Previous work by Lubana and Dick has shown that replicating this non-uniform structure in hardware can significantly reduce the number of pixels processed by a machine vision algorithm with only a slight reduction in accuracy. This leads to a corresponding decrease in power consumption [2].

We expand upon the work done by Lubana and Dick by taking advantage of temporal redundancies to reduce data transferred. By combining their digital foveation techniques with a cache of previous images, the amount of data transferred can be further reduced. The resulting scene cache is general-purpose and can be applied to any machine vision algorithm that takes full-frame video as an input.

Our work is directly based on digital foveation [2]. This work simulates the non-uniform sampling of the human retina through the use of digital foveation primitives. These primitives are simply commands given to the image sensor that can extract arbitrary rectangles of the image at different resolutions. They use these primitives in a multi-round image processing method. The image is first sampled at a low resolution. Regions of interest are identified, and if more resolution is required in those regions for the final task, digital foveation primitives are used to sample just those regions at higher resolution. By applying this approach to a license plate recognition system, digital foveation reduced energy consumption by 81.3% while reducing accuracy by less than 1%. This work differs from digital foveation by incorporating a temporal scene cache. Our system is designed to be general-purpose but uses digital foveation primitives to update itself.

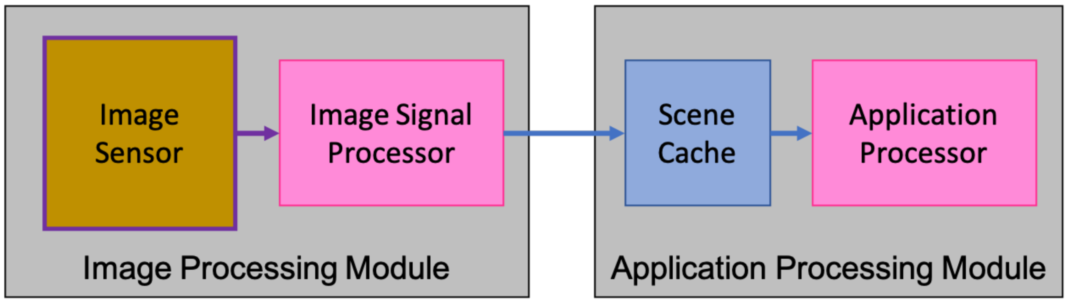


Figure 1: Scene cache in image signal processing pipeline

2 Scene Cache Design

Figure 1 shows our scene cache in the context of the image processing pipeline. Light is captured by the image sensor, the image signal processor processes the raw sensor data into a standard format, and the application processor performs a machine learning task such as object detection. The scene cache sits between the image signal processing step and the application processing step. These steps generally reside on separate hardware modules, and the scene cache could be placed on either. The figure depicts the scene cache on the application processor hardware since it is more general-purpose than the ISP module allowing the scene cache to be implemented more easily (e.g., as firmware).

The scene cache itself acts as a pipeline between the signal processor and the application processor, requesting image frames or pieces thereof from the image signal processor, caching that data, and sending the output to the application processor.

2.1 Design Philosophy

Our scene cache design strives for two main goals: generality and low energy consumption. An ideal scene cache would only collect and pass relevant information to the application processor. This approach, however, would require a knowledge of what data are important to the machine vision application. As there are a wide variety of machine vision applications, this would require a different scene cache for each. Alternatively, the application could feed back data to the scene cache to help it select important data, but this would require each application to be aware of the cache. In order to impact a variety of machine vision tasks without modifying them, we instead would prefer a general scene cache. This general scene cache should output a full-sized video frame at the frame rate used by the pipeline thus allowing the the machine vision algorithm to treat its output the same as that of a standard image signal processor.

Additionally, the primary goal of the scene cache is to save energy. The main way it does this is by reducing the data transferred from the image sensor thus reducing the computational energy of the image signal processor. By this same reasoning, the computational requirements of the scene cache should not be excessive. This rules out the use of many machine learning methods, especially neural networks, in the cache due to their large compute requirements.

2.2 Scene Cache Implementation

To comply with our design philosophy, our scene cache relies on motion detection to determine what pixels to sample at each new frame while unchanged pixels are already stored in the cache, thus reducing the amount of data transferred through the ISP. The cost function of this general scene cache trades off output frame quality vs. data transferred from the ISP.

Our general scene cache stores two images: a low-resolution image for computing where to sample and a full-resolution output that is fed to the application processor. On all but the first frame[1], the scene cache's update process has multiple steps:

1. Low-resolution sample
2. Difference calculation
3. Bounding box generation
4. High-resolution update

These steps are shown in figure 2 and described below.

Low-resolution sample The scene cache samples a low-resolution image from the ISP by skipping rows and columns. The number of rows/columns skipped is referred to as the downsample rate and must be an integer. We explore tuning this parameter in Section 4.1.

Difference calculation The low-resolution image is converted to grayscale and compared to a low-resolution grayscale version of the scene cache output at the previous frame. Subtracting these two yields a difference image.

Bounding box generation The difference image is thresholded where only pixels with a grayscale difference value above the threshold are considered to have changed. We explore tuning the threshold parameter in section 4.1. Minimum bounding boxes are drawn around the regions using the OpenCV findContours() and boundingRect() functions [3].

[1]On the first frame, the scene cache has no previous data in its low- and high-resolution caches, so it samples the full-resolution frame from the image signal processor, resulting in a one-time initialization cost.

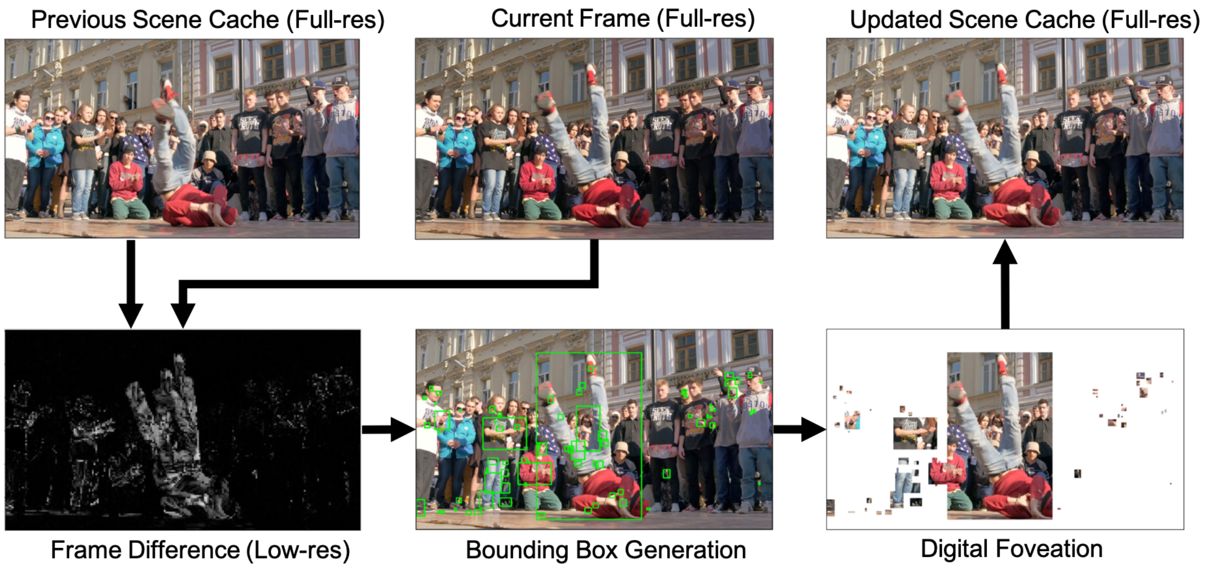

Previous Scene Cache (Full-res)
Current Frame (Full-res)
Updated Scene Cache (Full-res)
Frame Difference (Low-res)
Bounding Box Generation
Digital Foveation

Figure 2: Scene cache in action

The resulting bounding boxes tend to contain some overlap, so using them for digital foveation would result in duplicate transfers of pixel data. To overcome this, we developed an algorithm that removes overlap. We evaluated all the ways that two rectangles can overlap and determined how to cut them into one to three rectangles that cover the same area with no overlap. By performing this overlap removal on all pairs of bounding boxes (including those added by cutting two bounding boxes into three) we arrive at a new set of bounding boxes that covers the same area with no overlap.

High-resolution update The scene cache uses digital foveation primitives to sample full-resolution data from the ISP of the areas covered by the final bounding boxes. These high resolution boxes are pasted to the appropriate locations in the scene cache output. The low-resolution cache is updated for the next pass by downsampling the updated scene cache output.

3 Experiment Setup

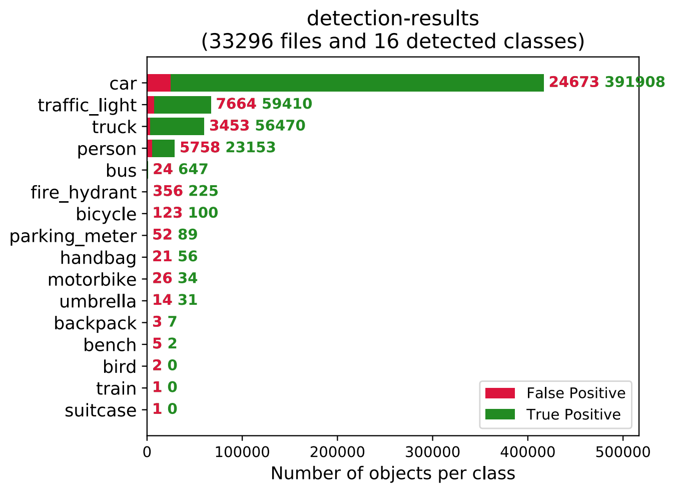
We evaluated our scene cache using a video streaming simulator built in Python. The simulator allowed us to input image sequences from standard video datasets to evaluate our scene cache. We can thus have repeatable testing and designated ground truth for our experiments.

To evaluate the performance of the scene cache, two sets of metrics were used:

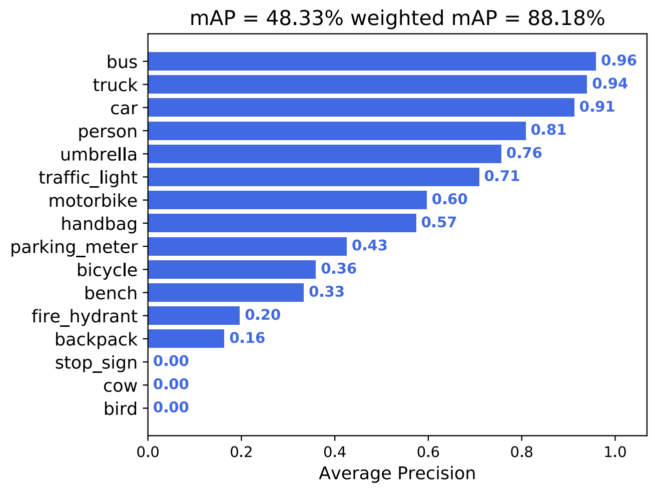
1. Video (per-frame) quality metric, i. e. how close is the output of the scene cache to the original. An output that faithfully replicates the original signal would suit a wide range of applications. There are no widely accepted video quality metrics specifically geared towards machine vision applications. We therefore present results of VIFP [4], SSIM [5], and PSNR as they are widely accepted video quality metrics. Although they are objective metrics, they are designed to reflect subjective human judgements of video quality.

2. Accuracy of specific tasks. To evaluate the effect of scene cache on vision-related tasks, we ran object classification with or without the scene cache, and used mean average precision (mAP) to compare the difference of the two [6]. However, mAP assigns equal weights to each class detected regardless of number of objects per class detected. Equal weight works fine for balanced datasets (a image dataset with similar number of objects per class), but poorly for a single sequence of frames from one video (Figure 3), because some objects are much more likely to appear than others (a video of an intersection would capture a lot more cars than cats). Due to this imbalance, we assigned weights to the AP of each class according to the number of objects detected in the original video for that class.

We used a test clip (breakdance) from DAVIS 2017 [7, 8] to validate our pipeline. After validating the pipeline, we chose clips from the baseline category of CDW-2012 dataset [9], as the dataset is specifically designed to test change detection.



(a)



(b)

Figure 3: Example of detection results (a) and average precision (b). Note the extreme imbalance of classes in (a).

To further stress our design, we used surveillance footage released by Digital Vision Security (DVS) [10]. We selected three clips, each capturing the same intersection at daytime, nighttime, and during rain.

In all test footage, the camera does not tilt, pan, zoom, or move. The background remains stationary.

We chose YOLOv3 [11] as an example of machine vision task. YOLOv3 is an object detection algorithm. Since most object detection databases use single still images rather than videos of objects, we ran video clips from our datasets through YOLOv3 to produce ground truth labels and bounding boxes for an object detection.

4 Experiments

In this section we evaluate the scene cache performance using our metrics. We first perform experiments to tune the scene cache downsample and threshold parameters. Once these are determined, we compare the scene cache performance with videos compressed using uniform compression.

4.1 Scene Cache Tuning

The two tunable scene cache parameters are the downsample rate and the pixel difference threshold whose functionality is mentioned in section 2.2. Since the downsample rate determines the number of pixels in the downsampled image used for calculating pixel-wise differences at every frame, it establishes a baseline amount of data transferred per frame. An increase in the downsample rate will decrease the amount of data transferred in proportion to the square of the downsample increase. Increasing it too much, though, could lead to missed motion detection on the pixels not sampled or large blocking artifacts thus reducing accuracy.

The threshold parameter determines how sensitive the scene cache is to change. There is some noise in pixel values between frames even for still scenes, so setting the threshold too low will cause it to capture large amounts of or possibly the whole image at each frame. If it is too high, then less motion will be captured, leading to blocking artifacts and reducing accuracy.

We ran a two dimensional parameter sweep of our scene cache and evaluated the performance of each instance on the CDW dataset. We swept the downsample rates in powers of two from $2\times$to $16\times$(i.e., keep only every 2nd, 4th, etc. pixel). Our threshold parameter ranged from 10 to 25 where the number is the grayscale difference between two pixels. Each pixel used 8-bit encoding and thus ranged from 0 to 255.

The results of our sweep are shown in Figures 4, 5, and 6. It can be seen that amount of data transferred decreases significantly as the downsample rate decreases and decreases slightly as the threshold increases. The mAP and weighted mAP scores decrease as the threshold and downsample rate increase but appear to be more affected by the threshold. We choose the set of parameters that gave weighted mAP scores above an arbitrarily chosen value of 90% while transferring the least amount of data. This yielded a downsample rate of $8\times$and a threshold of 10. We used these parameters for the scene cache in the rest of our experiments.

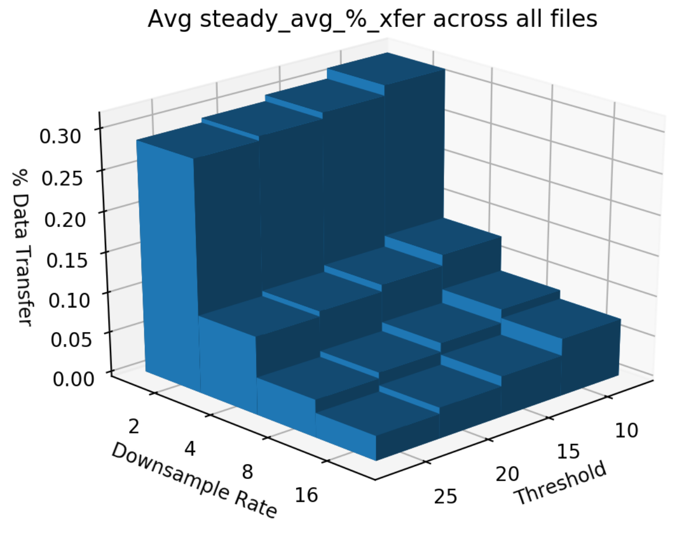


Figure 4: Data transfer parameter sweep results for CDW dataset. Data measured in ratio of data transferred by scene cache vs full resolution data transfer.

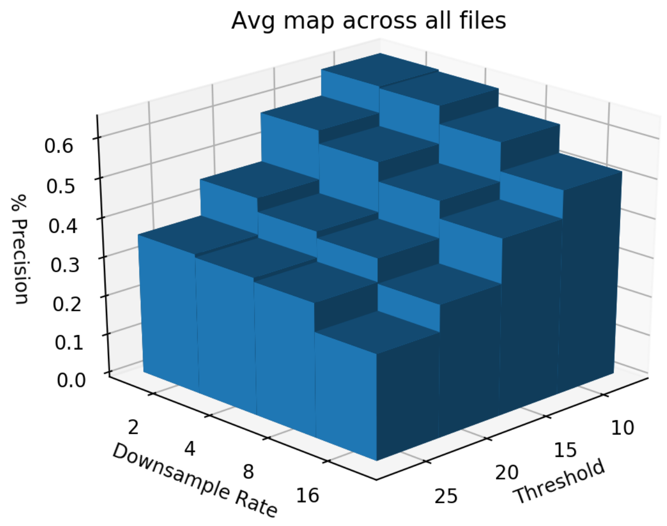


Figure 5: mAP parameter sweep results for CDW dataset. mAP measured in percent precision.

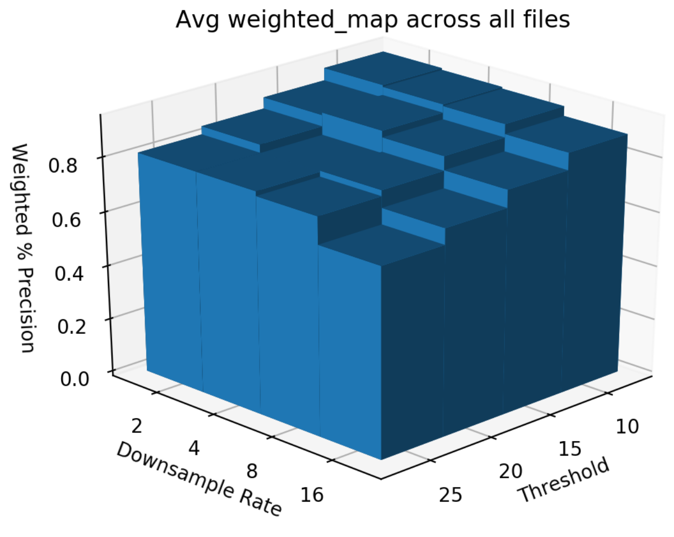


Figure 6: Weighted mAP parameter sweep results for CDW dataset. Weigthed mAP measured in weighted percent precision.

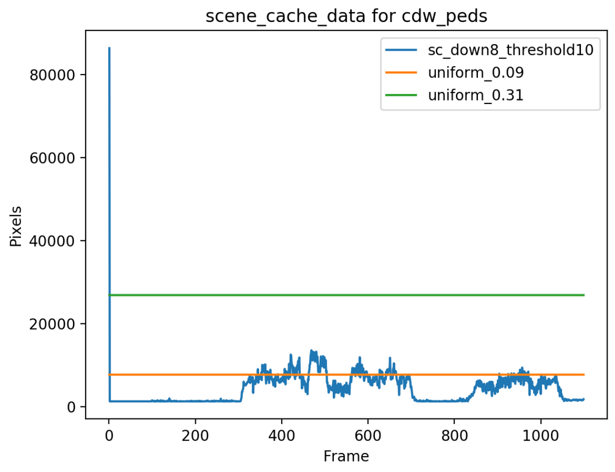


Figure 7: Scene cache data transfer for video "pedestrians" from CDW dataset versus uniform downsampling. Note the high outlier on the first frame.

4.2 Uniform Downsampling CDW Experiments

The scene cache affects both the data rate and classification accuracy of the machine learning algorithm using it. To evaluate its efficacy, we need to control for one of those variables. We decided to control for the data rate of the scene cache and compare the accuracy metrics.

We determined the data rate of the cache by looking at its steady state data transfer metrics. Steady state metrics refer to data transfer at all frames except the first one. For the first video frame, the scene cache has no prior information and requests the full image, making it an outlier. We calculate both the average and maximum steady state data rates for each file in the CDW dataset and take the average of both of those groups. The average and maximum steady state data rates were 9% and 31% respectively. Figure 7 shows an example of the scene cache data rate for one video in the CDW dataset compared with the average and maximum steady state data rates for the whole dataset.

We generated low data rate versions of our reference videos to compare with the scene cache. To do so we calculated the scaling factor necessary to reduce the number of pixels in the generated video to the percent data reduction of the maximum and average steady state scene cache rates. We downsampled the video by this scaling factor using bicubic interpolation. We then upsampled the video back to the original size using bicubic interpolation. This was done since our video quality metrics and YOLOv3 classification assume that the images being compared are the same size. The downsampling and upsampling effectively reduced the data rate by our desired amount, and the resulting videos appeared as blurry versions of the original.

Table 1 shows the accuracy results of the scene cache versus those of the downsampled videos. The scene cache has superior results for both mAP and weighted mAP.

Table 2 shows the video metric results of the scene cache versus those of the downsampled videos. Again, the scene cache has superior results in all three metrics.

Experiment	mAP	Weighted mAP
Scene Cache	0.591	0.920
Unif. Downsample 9%	0.338	0.707
Unif. Downsample 31%	0.543	0.859

Table 1: CDW Accuracy Metrics

Experiment	SSIM	VIFP	PSNR
Scene Cache	0.955	0.642	31.27
Unif. Downsample 9%	0.852	0.461	25.26
Unif. Downsample 31%	0.944	0.622	29.51

Table 2: CDW Video Metrics

4.3 DVS Experiments

We performed the same scene cache versus uniform downsampling experiments using the DVS dataset as we did for the CDW dataset. We kept the same scene cache parameters. We also used the same average and maximum steady state transfer rates (9% and 31%, respectively) as the CDW tests. This setup tests whether the parameters developed with CDW generalize well to another dataset.

Figure 8 shows the average steady data transfer for DVS. The scene cache outperforms both uniform subsampling metrics.

We were only able to obtain video quality metrics for one of the three videos in the DVS dataset due to issues with our processing script. Figure 9 shows the plot of SSIM for each frame for the one successful video "day." Note that the SSIM for the scene cache slowly decreases at the beginning before eventually stabilizing. A similar phenomenon appears for PSNR and VIFP in this and various videos CDW. We believe this is due to blocking artifacts accumulating over time (as seen in Figures 10 and 11). Despite this, the scene cache still outperforms the uniform downsampling in all three metrics as shown in Table 2 for CDW and in Table 4 for "day" in DVS.

Experiment	mAP	Weighted mAP
Scene Cache	0.384	0.847
Unif. Downsample 9%	0.233	0.578
Unif. Downsample 31%	0.478	0.824

Table 3: DVS Accuracy Metrics

Experiment	SSIM	VIFP	PSNR
Scene Cache	0.916	0.551	29.58
Unif. Downsample 9%	0.749	0.343	22.42
Unif. Downsample 31%	0.883	0.482	25.89

Table 4: Video Metrics for DVS "day"

Table 3 shows the accuracy results on the DVS dataset. The scene cache does not have the highest mAP, but still has the highest weighted mAP. Figure 10 shows the output from the scene cache and the two uniform downsamplings against the original video for DVS "day". The scene cache output is as sharp as the original video because all blocks are taken at the original resolution, but it contains more persistent artifacts, especially at places

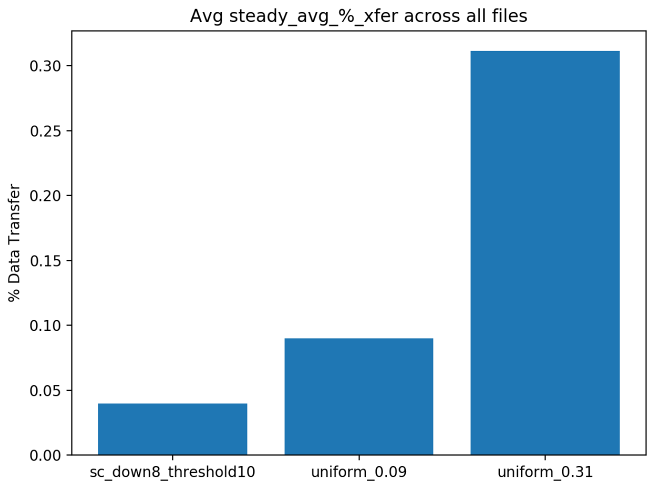


Figure 8: Average steady data transfer for DVS

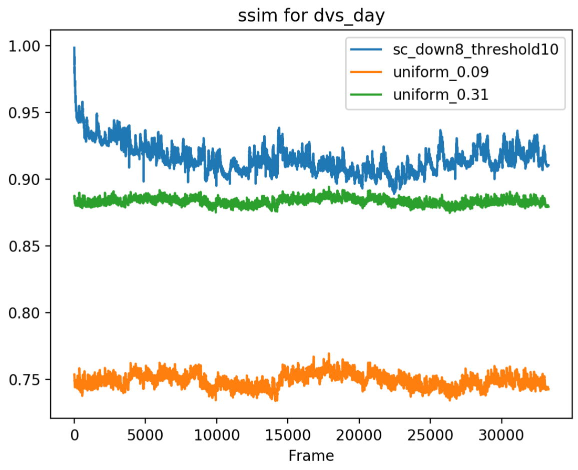


Figure 9: SSIM for "day" video in DVS for both scene cache and uniform downsampling

with small variations across frames (cloud and road). It is likely that changes from some blocks in the region reached the threshold while most blocks do not during most of the time, causing temporally inconsistent blocks to remain for a long time. Depending on the application, however, those artifacts may not affect the accuracy significantly. Movement of cars from the scene cache also appear choppy as blocks won't update until sufficient change is detected. The uniformly downsampled images look considerably blurrier, especially the 9% one.

Artifacting gets significantly worse in nighttime videos. Figure 11 shows the results for DVS "night". Output from the scene cache leaves a trail of red lights behind the car. Since the scene cache only looks at the difference in grayscale, it is effectively colorblind, only concerning itself with change in brightness. The taillight of the car has a bright spot that strongly contrasts with the road, but the afterglow is not bright enough to trigger a block update. Looking at the Euclidean distance in the RGB colorspace should alleviate such problem.

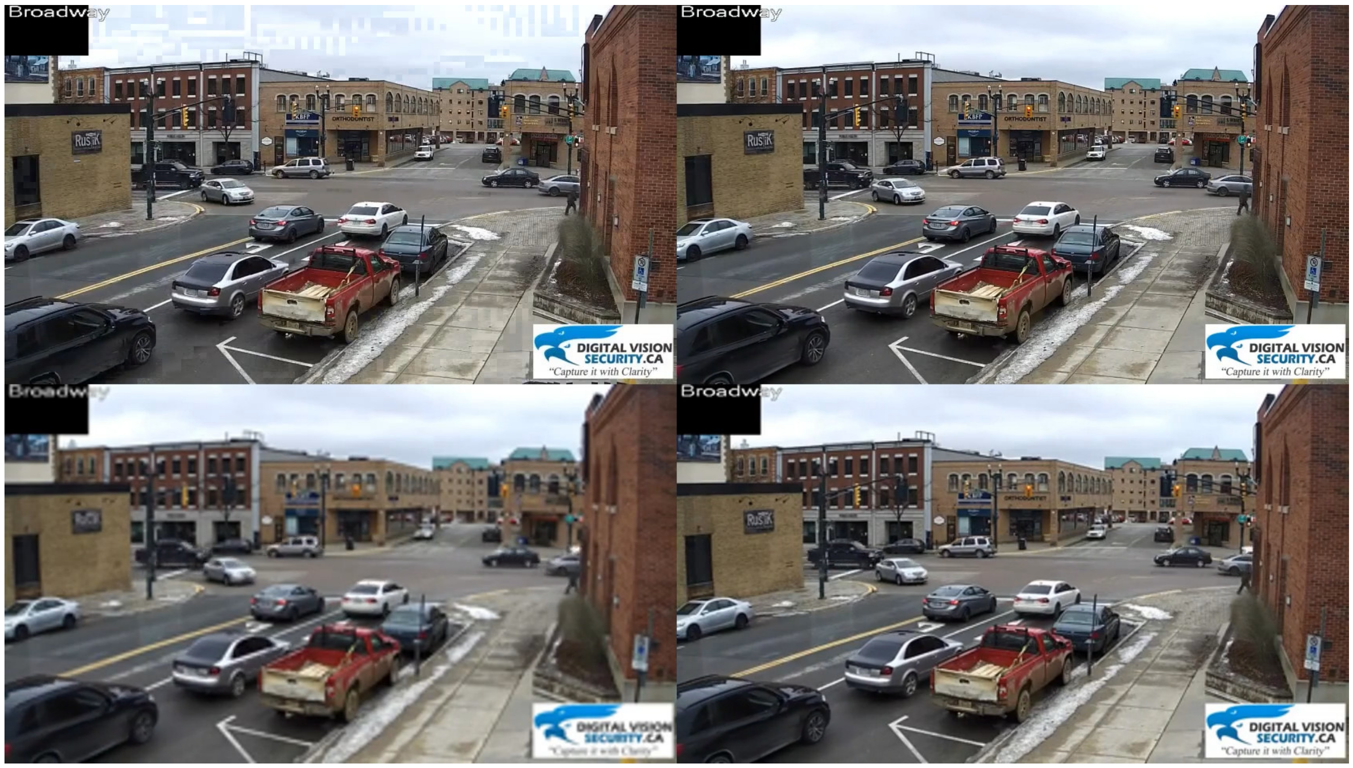


Figure 10: DVS "day" results, clock-wise, starting from top-left: scene cache with parameters determined in section 4.1, original video, uniform downsampling of 31% (bottom-right) and 9% (bottom-left)

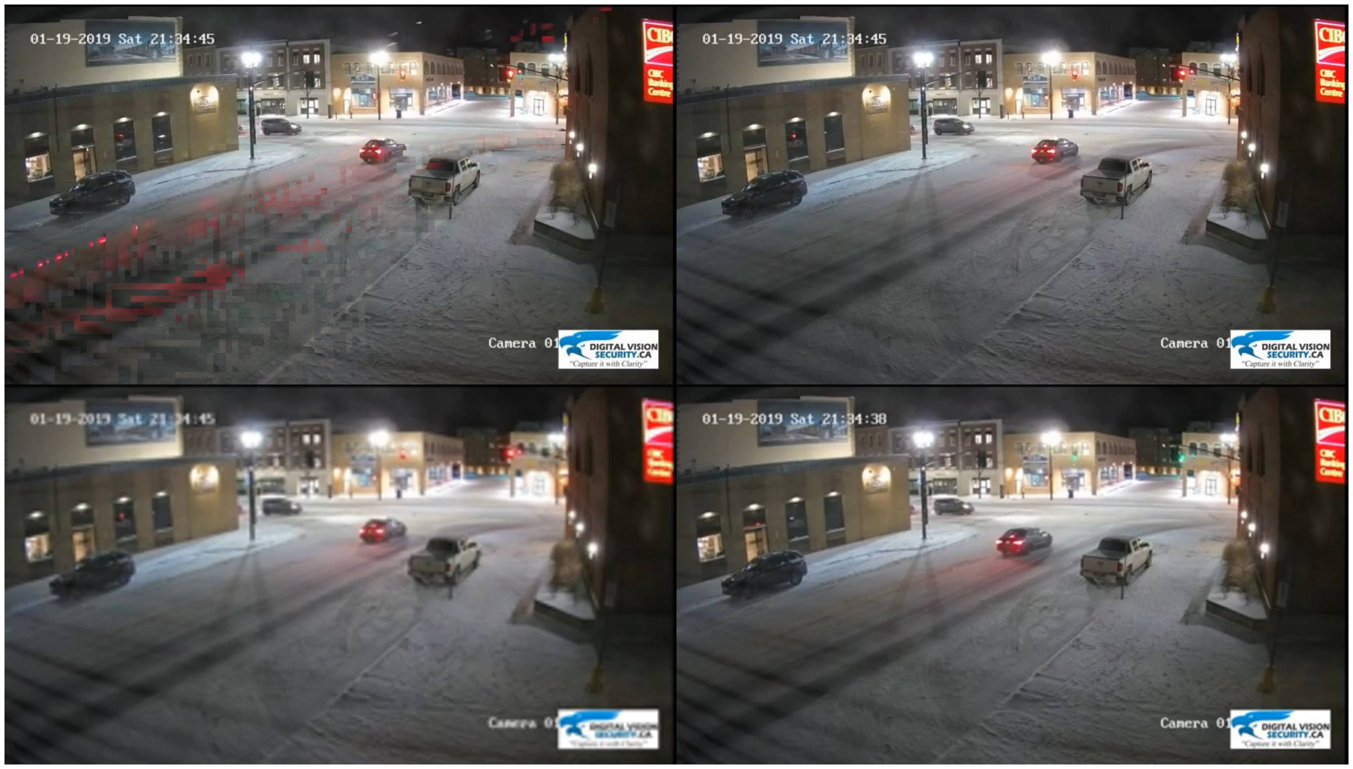


Figure 11: DVS "night" results, clock-wise, starting from top-left: scene cache with parameters determined in section 4.1, original video, uniform downsampling of 31% (bottom-right) and 9% (bottom-left)

5 Related Work

Euphrates is a hardware/software implementation for object detection and tracking aimed at reducing energy consumption [12]. It reduces the amount of frames where expensive object detection algorithms need to be run by interpolating the motion of known regions of interest (ROI) in the intermediate frames. It can achieve 66% energy savings with 1% or less loss in accuracy. This takes a similar approach to us in that it tracks ROI in a foveated manner using motion information. Unlike us, it is mainly concerned with the application layer, requiring detected ROI coordinates from the machine vision algorithm. It also relies on custom hardware modules. Our energy saving methods could likely be further improved by incorporating ROI information from the machine vision algorithm.

Our scene cache is somewhat similar to traditional video compression. Both try to reduce data transferred while outputting full video frames. However, video compression assumes full frames as input. It does not reduce energy spent of the ISP. Video compression also focuses on compression ratio instead of power consumption and speed.

Compressed sensing is another similar field. It attempts to reduce the amount of data sampled by a sensor by assuming that the domain of possible inputs is sparse. This allows for perfect signal reconstruction with sampling rates far lower than the Nyquist rate [13]. Our technique is similar in that it assumes sparse motion differences can be used to reproduce our input video. It is dissimilar in that its technique is implemented with ad-hoc heuristics rather than formal mathematical models.

6 Future Work

We did not test the case when the camera is not stationary. We plan to investigate camera motion estimation to better match consecutive frames, thus allowing us to better exploit spatial redundancy. Our scene cache currently relies solely on motion estimation for updates. We may investigate whether further data transfer savings can be achieved by using the classification algorithm to identify regions of interest. This would reduce the generalizability of the scene cache but could allow us to avoid sampling unnecessary information.

The scene cache attempts to reduce energy consumption by sampling less data from the camera. It also adds computational complexity and hence draws energy. It would be useful to analytically investigate the computational power consumption of different scene caches in order to optimize scene cache architecture based on total energy saved rather than solely on data transfer energy saved.

By this same reasoning, an ideal scene cache would also reduce the computational energy of the application processor, but we forego this for the sake of generality. Future work may reap further energy savings by adapting to a specific application.

The experiment setup needs to be improved as well. Current setup for evaluating video quality metrics is not fast or reliable.

7 Conclusion

Inspired by digital foveation, scene cache applies a multiround process that exploits both the spatial and temporal redundancy in image sequence. It can significantly reduce data transferred and thereby save power. It achieves 88% accuracy at 4% data transferred for videos with static perspectives.

References

[1] J. E. Niven and S. B. Laughlin, "Energy limitation as a selective pressure on the evolution of sensory systems," *Journal of Experimental Biology*, vol. 211, no. 11, pp. 1792–1804, 2008.

[2] E. S. Lubana and R. P. Dick, "Digital foveation: an energy-aware machine vision framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2371–2380, Nov 2018.

[3] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2018.

[4] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, Feb 2006.

[5] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[6] Cartucho, “mean average precision,” 2018. [Online]. Available: https://github.com/Cartucho/mAP

[7] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, “A benchmark dataset and evaluation methodology for video object segmentation,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, “The 2017 davis challenge on video object segmentation,” *arXiv:1704.00675*, 2017.

[9] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, “Changedetection.net: A new change detection benchmark dataset,” in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012, pp. 1–8.

[10] I. Digital Vision Security. (2014–2019) Digital vision security - video camera surveillance systems & solutions. [Online]. Available: https://www.youtube.com/channel/UCSeMkpNPR7UGFOlBfZ0bRNg

[11] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv e-prints*, p. arXiv:1804.02767, Apr 2018.

[12] Y. Zhu, A. Samajdar, M. Mattina, and P. N. Whatmough, “Euphrates: Algorithm-soc co-design for low-power mobile continuous vision,” in *45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los Angeles, CA, USA, June 1-6, 2018*, M. Annavaram, T. M. Pinkston, and B. Falsafi, Eds. IEEE Computer Society, 2018, pp. 547–560.

[13] M. Rani, S. B. Dhok, and R. B. Deshmukh, “A systematic review of compressive sensing: Concepts, implementations and applications,” *IEEE Access*, vol. 6, pp. 4875–4894, 2018.