2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)

ZenCam: Context-Driven Control of Continuous Vision Body Cameras

Shiwei Fang
UNC Chapel Hill
shiwei@cs.unc.edu

Ketan Mayer-Patel
UNC Chapel Hill
kmp@cs.unc.edu

Shahriar Nirjon
UNC Chapel Hill
nirjon@cs.unc.edu

***Abstract*—In this paper, we present — *ZenCam*, which is an *always-on* body camera that exploits readily available information in the encoded video stream from the on-chip firmware to classify the dynamics of the scene. This *scene-context* is further combined with simple inertial measurement unit (IMU)-based *activity level-context* of the wearer to optimally control the camera configuration at run-time to keep the device under the desired energy budget. We describe the design and implementation of ZenCam and thoroughly evaluate its performance in real-world scenarios. Our evaluation shows a 29.8-35% reduction in energy consumption and 48.1-49.5% reduction in storage usage when compared to a standard baseline setting of 1920x1080 at 30fps while maintaining a competitive or better video quality at the minimal computational overhead.**

***Index Terms*—Model Predictive Control, Encoded Video Analysis, Body Camera, System Optimization**

I. Introduction

In recent years, the use of body-worn cameras has increased exponentially—from law enforcement officers, to first responders, to the military. The need for having a body camera to carry out day-to-day jobs with a higher level of competency has brought many commercial products into the consumer market. Typically, body cameras are used as a recording device which stores multimedia content inside the device. These video feeds are later downloaded and analyzed off-line. Besides satisfying application-specific video-quality requirements, the two most desirable properties of body cameras are *extended battery life* and *efficient storage*. However, most body cameras of today have a short battery life which limits our ability to capture hours-long events. Storage space is also limited in these portable systems. Although the cost of storage has become cheap nowadays, efficiency is always desirable, and in some cases, it is necessary as there is monetary cost associated with archiving a large amount of data, and for this reason, often security videos are archived for a limited period.

Unfortunately, today's body cameras are developed somewhat in an ad hoc manner by gluing together different sensing, computational, and communication modules with limited or guarantee-less optimization in their designs. When such systems are deployed in the real world, they fail to provide a promised quality of service and an extended battery life. We argue that in order to develop an efficient body camera that provides an *extended battery-life*, *efficient storage*, and *satisfactory video quality*, instead of an ad hoc combination of independently developed modules, we need to apply context-aware control-theoretic principles and engineer a body-worn camera that is dependable, efficient, and robust.

Existing implementations of body cameras typically consist of an on/off switch that is controlled by its wearer. While this design suits the purpose in an ideal scenario, in many situations, the wearer (e.g., a law-enforcement officer) may not be able to predict the right moment to turn the camera on or may completely forget to do so in the midst of an action. There are some cameras which automatically turns on at an event (e.g., when a gun is pulled), but they miss the "back-story," i.e., how the situation had developed.

We advocate that body cameras should be *always on* so that they are able to continuously capture the scene for an extended period. However, cameras being one of the most power-hungry sensors, the lifetime of a continuous vision camera is limited to tens of minutes to few hours depending on the size of the battery. A commonsense approach to extend the battery-life of a continuous vision camera is to analyze the scene and record it at high resolution and/or at high frame rate only when the scene dynamics is high and vice versa. However, on-device scene dynamics analysis is extremely costly and the cost generally outweighs the benefit. Recent works [16, 22] therefore employ secondary low-power cameras (e.g., a thermal imaging sensor) and FPGA-based scene analysis to wake-up the primary camera when an interesting event is detected at a lower energy cost. These systems, however, have the same limitation of missing the back-story, and in general, they are bulkier than a single camera-based system due to the additional camera sensor.

In this paper, we present *ZenCam*, which is an *always on* and *continuously recording* body camera that ensures the *desired battery-life* and a *near-optimal*[1] *video quality*. The camera analyzes the *dynamics of the scene* as well as the *activity level* of the wearer in real-time, and controls the parameters of the body camera (e.g., frame rate and resolution) in order to satisfy the battery-life and the video quality requirements. The novel technical aspects of ZenCam are two-fold:

- First, we develop a light-weight video analytics algorithm that operates entirely on the encoded video stream obtained directly from the camera firmware for fast and low-power scene dynamics classification. This is different from existing algorithms that decompress and analyze video frames in the pixel domain.
- Second, a novel control-theoretic approach where we employ a *model predictive controller* [28] to dynamically control the frame rate and resolution of the camera sensor to achieve the desired battery life and a near-optimal video quality.

[1]Around 2% decrease in video quality compared to baseline model.

2325-2944/19/$31.00 ©2019 IEEE
DOI 10.1109/DCOSS.2019.00029

We develop a prototype of ZenCam using low-cost, off-the-shelf sensors, and lightweight, open source software modules. We identify the system parameters that affect the output power and the video quality, and empirically model their relationship. We implement a scene dynamics analysis algorithm which uses natively available motion vectors from the camera and implement a light-weight activity-level classifier that uses an on-board accelerometer to determine the activity-level of the wearer. The control algorithm uses both types of contextual information to dynamically adapt the camera parameters.

We deploy ZenCam in multiple real-world environments, such as an office building, a street, and inside a car, and evaluate its performance. We demonstrate that ZenCam achieves a 29.8-35% reduction in energy consumption and a 48.1-49.5% reduction in storage when compared to a fixed-configuration body camera, without losing the video quality. Compared to an Oracle system, ZenCam's computational overhead is 10-17% with unoptimized hardware and software implementation. To the best of our knowledge, ZenCam is the first of its kind to achieve such energy and storage savings via an on-device, single camera-based, light-weight, low-power scene dynamics analysis algorithm, and a minimal hardware addition (i.e., an IMU), without sacrificing the video quality as the scene and user dynamics change at run-time.

II. ZenCam System Design

ZenCam is an autonomous, always-on, continuous-vision body camera that dynamically adjusts its configurations based on both the degree of body movement of the wearer and the dynamics of the scene. As a result, ZenCam achieves a desired battery life and ensures an optimal video quality of the recorded scenes. The system architecture of ZenCam is depicted in Figure 1.

A. ZenCam Hardware Components

ZenCam's hardware consists primarily of a camera sensor, an onboard video processing unit (VPU), and an inertial measurement unit (IMU). Each of these components is independently adjustable to suit application-specific requirements. The video processing unit (VPU) is a chip that provides hardware-processed video frames along with the meta-data such as the motion vectors. ZenCam exploits these readily available meta-data to classify the scene dynamics efficiently at run-time. The inertial measurement unit (IMU) is a low-cost and low-power sensor that provides raw accelerometer data, which is used by ZenCam to estimate the activity level of the user in real-time.

B. ZenCam Software Components

ZenCam's software architecture consists of three major components: `Encoded Analysis`, `Activity Classifier`, and `Model Predictive Controller`. The `Activity Classifier` detects and classifies the user's body activity level into low, medium, and high levels. If the detected activity level is *high* or *medium*, which implies that the user is in motion, the result is directly used by the `Context Manager` of the `Model Predictive Controller` to determine the corresponding camera settings. In this case, the `Encoded Analysis` component remains dormant, which minimizes the energy and computational overhead of the system.

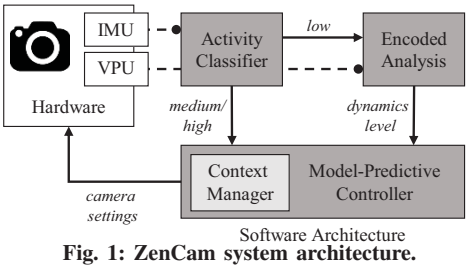


Fig. 1: ZenCam system architecture.

On the other hand, when the `Activity Classifier` determines that the user's activity level is *low*, it notifies the `Encoded Analysis` component to analyze the encoded video data along with the IMU data to determine the scene dynamics. The rationale behind this is that even if a user is inactive, the scene may have a different level of dynamics (e.g., a mostly static environment vs. an ongoing riot), which would require a different camera settings to capture the scene. With direct access to the motion vectors and residual values, ZenCam does not have to decode the video. The `Encoded Analysis` component works directly on the encoded video stream to determine the scene dynamics and passes the result to the `Context Manager`. Based on the context of the user as well as the scene, the `Model Predictive Controller` determines and sets the optimal values of the camera parameters.

C. The Novelty of ZenCam Design

We highlight novel aspects of ZenCam that differentiates it from existing continuous vision camera systems.

1) Uses Readily Available Information: Understanding the scene dynamics is important to control the camera parameters optimally. However, processing video frames in real-time is not practical in CPU and energy-constrained embedded systems. Hence, ZenCam relies on readily available information in the video meta-data, such as the motion vectors and residual data, to infer the scene dynamics. This information is available in every camera chip and comes at no extra cost of computation. As a result, unlike existing smart cameras, ZenCam does not have to decode, then process, and then again encode the video.

2) No Pixel-Domain Analysis: ZenCam does not analyze videos in the pixel domain, which is an extremely CPU and energy demanding task. Instead, it performs simple operations to filter unwanted motion vectors and to compute the entropy of a frame using these vectors – which accurately determines the dynamics of the scene at several order of magnitude less cost than pixel domain analysis of video frames.

3) Based on Control Theoretic Principles: In ZenCam, a model predictive controller is employed that dynamically adjusts the camera settings in order to achieve both reduced energy consumption and storage requirement while maintaining the desired video quality for a given user and scene

(a) Standing with minimal body movements **(b) Running in front of static camera** **(c) Camera movements in static scene** **(d) Running in front of moving camera**

Fig. 2: Motion vectors overlaid on video frames. The top frames are recorded at 10 fps while the bottom frames are at 30 fps. Black and white pictures are the magnitudes of the residual values. (a) also shows enlarged motion vectors near the subject's shoulder.

context. Unlike existing continuous vision cameras that rely on heuristics to balance battery life and image quality, ZenCam provides a provable near-optimal performance.

4) Lightweight Algorithms: ZenCam employs lightweight algorithms in all of its software components. For instance, the scene analysis algorithm uses simple arithmetic operations to obtain the scene dynamics in linear time. The activity classifier computes lightweight features from IMU data in $O(1)$ and uses a pre-trained classifier. The controller's computational complexity at runtime is also optimized to simple table look-ups and constant time operations. Hence, the overhead of the system is negligible and its benefit outweighs the cost.

III. Encoded Video Analytics

When a video is recorded, the encoder calculates *motion vectors* and *residual values* [27]. Motion vectors represent the translation of pair-wise most similar macro-blocks (e.g., 8x8 pixels) across frames. Residual value is calculated as the difference between the macro-block in the current frame and the previous frame's matching one. In the final video, the current frame is reconstructed from macro-blocks of the previous frame and the differences. ZenCam exploits these two freely available information as the foundation of its lightweight scene dynamics classification algorithm.

A. Preprocessing Motion Vectors

Motion vectors are computed by video coding algorithms which matches similar macro-blocks across different frames in order to minimize the residual value. Motion vectors produced in this manner sometimes do not represent the true motion of the objects in the scene as these greedy algorithms often match *similar* macro-blocks that minimize residuals. While such motion vectors, combined with residuals, are perfectly okay for video coding, they are problematic when determining the true movements of objects in a scene.

ZenCam filters out these erroneous vectors in a preprocessing step. From the latest frame in a time period T, we calculate the angle, $d_i = \arctan \frac{\hat{v}_y}{\hat{v}_x}$ of a motion vector $\hat{v} = (\hat{v}_x, \hat{v}_y)$. To identify erroneous cases, we create a histogram of $\{d_i\}$ and discard the motion vectors whose angles fall into bins that has a very small number of elements.

There are two intuitions behind this step. First, the number of erroneous motion vectors tends to be small within a certain angular range. Second, even if the discarded motion vectors correspond to correct motion of an object in the scene, we still should discard those as the object occupies a minimal area where a large macro-block is 16x16 pixels. This means that the object is either far away or tiny. In this situation, the motion of the object typically does not generate a fast motion across the frame. Thus, such scenes are not to be recorded at high frame rate.

After discarding erroneous motion vectors, we normalize the vectors to compensate for the effect of frame rates.

$$Scaled\ v_i = v_i \times \frac{current\ frame\ rate}{reference\ frame\ rate} \tag{1}$$

The scaling is necessary as different frame rates produce motion vectors of different magnitudes even when an object is moving at the same speed. This is shown in Figure 2(a). The motion of the human subject is identical in both frames as they are recorded simultaneously. The top one has a rate of 10 fps, and the bottom one is 30 fps. The zoomed-in view shows that the magnitude of the motion vector is larger at the lower frame rate. By scaling the motion vectors, we normalize the motion of an object across different frame rates. After scaling, we filter out motion vectors whose magnitude, $m_i = \sqrt{\hat{x}^2 + \hat{y}^2}$ is below a small threshold to eliminate random noise or movements corresponding to low scene dynamics.

Finally, we compensate the effect of camera movements on motion vectors. We calculate ZenCam's orientation using the gyroscope data and estimate the direction of the frame's movement by projecting the orientation on a plane parallel to the camera. We discard the motion vectors that are in the opposite direction of the camera's movement. This is because, when the camera moves, the frame tends to move in the opposite direction. As a result, we discard the motion vectors that are in the opposite direction of the camera's movement.

B. Scene Dynamics Estimation

A high residual value indicates that the macro-block does not match well across frames. In such cases, we consider that the probability of movement is high. Within a frame, we convert the residual values into a distribution, p_i by normalizing against the sum of residuals of a macro-block. We apply a *softplus* function to convert motion vectors to intensity:

$$I(\hat{v}_i) = \ln(1 + e^{m_i}) \tag{2}$$

We calculate the entropy of a frame as follows:

$$E = \sum_{i=0}^{N} p_i I(\hat{v}_i) \tag{3}$$

where, N is the total number of motion vectors after the preprocessing step. We divide E by N to get a score S for the current frame, which classifies the scene dynamics. We accumulate consecutive η classification results to perform a majority voting. The rationale behind this algorithm is that when there are no moving objects in a scene, motion vectors caused by small camera movements will be removed and the score will be low. The score of a scene having moving objects in it depends on the speed of motion and the percentage of the area occupied by the object in the frame. The larger the score, the more dynamic a scene is.

The proposed encoded-domain scene analysis technique uses readily available features and requires neither decoding nor re-encoding the frames. The algorithm is lightweight and parallelizable. The implication is that it can be implemented in hardware (e.g., an FPGA or even ASIC) to gain substantial energy savings and speedup.

IV. Controller Design

A. Camera Controller

We design the body camera control system as shown in Figure 3. The controller takes input from the sensor which measures the system's battery level $b(t)$. The context look-up table provides the weights $(w_1, w_2, \dots)$ and the reference values of resolution r^r and frame rate f^r for the current context. The controller computes the control input (r, f) based on these parameters and the reference value of the desired battery level $b^r(t)$.

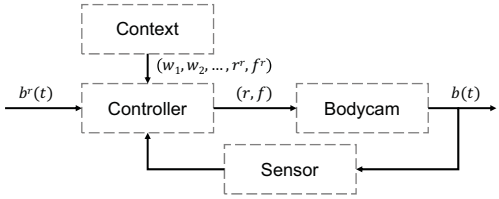
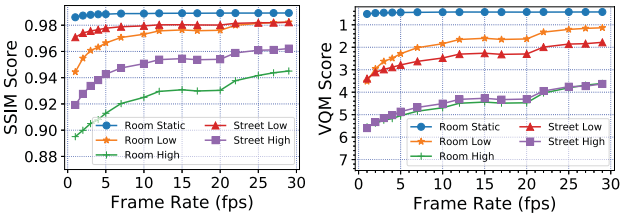


Fig. 3: Controller design for the body camera

We define the objective function for the controller to match the actual power consumption to a reference value (e.g., set by the application-specific policy) while maintaining a certain video quality. For instance, assuming a fixed camera setting and linear battery drain, if the target battery life is 8 hours, the system should have 50% battery remaining after 4 hours. In this case, the linearly dropping battery level is the reference at any point in time. The cost function is defined as:



(a) SSIM Score **(b) VQM Score**

Fig. 4: SSIM and VQM Scores at different frame rates.

$$J = (b - b^r)^2 + \sum_{i=0}^{N} w_i S(x_i - x_i^r) \tag{4}$$

$$\text{where, } S(x) = \frac{1}{1 + e^{-a \cdot x}} - 0.5 \tag{5}$$

In (4), b is the future battery level *after* applying the camera setting for the time step, b^r is the reference battery level, x_i is the control input to the system (i.e., (r, f) in Figure 3), and x_i^r is the reference values (i.e., (r^r, f^r) in Figure 3), w_i are the weights of these variables, S is the sigmoid function where a determines the range of configurations that we want to focus and apply aggressive control. Each context has its own reference values for control variables and weights, and the sigmoid terms reward or penalize certain behavior in each context. We use sigmoid functions because of their decreased rate of change when the actual value moves away from the reference and vice versa – which enables the system to take more aggressive action when necessary.

The model predictive controller optimizes its objective over a finite time horizon. Given the current context, we optimize (4) over the next time step considering the most likely sequence of contexts and their average duration. The sequence of contexts and their duration are statistically estimated over time. The power consumption for each camera configuration is updated to minimize the impact of inaccuracies in system modeling and changes in hardware performance such as battery degradation.

B. Determining Reference Values

There are studies that model user perception of video quality at varying frame rates and resolution [25, 30]. Although these data can not be directly used in ZenCam due to the differences in the system architecture and goals, we use the technique from [30] to calculate the quality scores for different camera settings. We use two metrics: *Structural Similarity Index Metric (SSIM)* [29] and *Video Quality Metric (VQM)* [26]. These two objective metrics correlates with human perceptions.

We record five types of videos at different indoor and outdoor scenarios at 1920x1080 resolution and 30 fps. Frame rates of these videos are altered by dropping frames to simulate different camera settings. Finally, all videos are converted to the same frame rate using the same encoder for comparison. We increase the frame rate of low-frame-rate videos by duplicating frames. We use MSU Video Quality

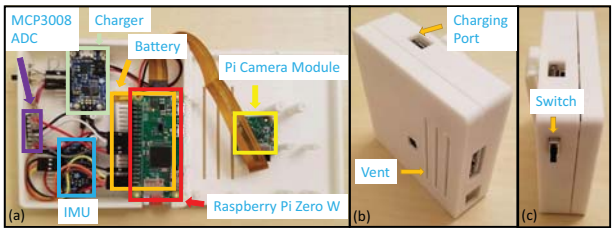


Fig. 5: ZenCam Prototype: (a) Internal components are labeled. (b) The right-front view. The camera is near the vent. (c) The left view, switch, charging port, and clip.

Measurement Tool [15] to measure the SSIM and VQM scores for each video. The SSIM scores (higher is better) are shown in Figure 4(a) and the VQM scores (lower is better) are in Figure 4(b).

From these measurements, we observe the tendency of decreasing quality with decreasing frame rate. The decrease in quality is less noticeable when the scene is less dynamic. In ZenCam, for each camera state, we set the reference video quality to 2% lower than the highest SSIM score. This reference value is a configurable parameter in ZenCam, and if needed, it can be set to a different value which is more suitable for specific applications.

V. System Implementation

A. Hardware Development

We implement a prototype of ZenCam which is shown in Figure 5. We use a Raspberry Pi Zero W and a Pi Camera Module V2 as the main components. We connect an LSM9DS0 IMU to the Raspberry Pi via I^2C. A Raspberry Pi Zero is used to demonstrate that our system works with off-the-shelf parts and runs on resource-constrained platforms. Pi Camera also provides us with an easy access to motion vectors and residual values without decoding the frames. This information is available in nearly every camera chip but requires a slight software modifications to expose it. For example, [20] made changes to the OS to expose the motion vectors on an Android device.

B. Software Development

We use the `picamera` Python library to control the camera configuration and to retrieve the encoded domain data. The `scikit-learn` Python library is used for SVM classification. We collect IMU data at 10Hz which is sufficient for user activity detection at a minimal CPU utilization. The battery voltage is queried every second and the percentage is calculated from the average voltage over 30s. Video scene analysis is performed every 250ms when the camera activity is low, which results in a high accuracy at a minimal overhead.

To classify the activity level of a user, we implement a lightweight algorithm similar to [12, 23]. From the raw accelerometer data, we calculate the magnitude of the acceleration: $\alpha = \sqrt{a_x^2 + a_y^2 + a_z^2}$. The standard deviation of $\{\alpha_i\}$ over 500ms is used to train a support vector machine (SVM) with a radial basis function (RBF) kernel. With a trained SVM classifier, we extract the data and perform classification on data points from past 500ms every 100ms (i.e., 10 Hz). We collect a series of classification results over 5s and apply majority voting to determine the most likely activity level of the user.

VI. Energy Overhead Measurement

We use Pi Camera sensor's mode 4 and select the resolution settings from a pool of three choices: {`1920x1080`, `1600x900`, and `1280x720`}, which are commonly used in real-world applications.

To understand the overhead of ZenCam algorithms, we measure its power consumption for the same camera setting with and without the algorithms running. The results are shown in Table I. The *baseline* refers to the same camera settings as ZenCam but without the algorithms running. We see that the overhead is between 3.7%–17.6%, and it's the smallest when the system uses the highest frame rate. The overhead of ZenCam when the camera is steady decreases as the scene dynamics decreases because there are fewer motion vectors to process when the camera records at a lower resolution.

IMU	Scene Level	Resolution	Frame Rate (fps)	Baseline Power (W)	ZenCam Power (W)
High	N/A	1600x900	40	1.88	1.95
Medium	N/A	1920x1080	26	1.53	1.62
Low	High	1920x1080	25	1.47	1.73
	Medium	1600x900	15	1.14	1.33
	Low	1280x720	4	0.86	0.99

TABLE I: Power consumption at different contexts and settings.

When ZenCam is in *high* or *medium* activity level, the energy overhead to drive the IMU sensor and its processing algorithm is about 80mW. This overhead can be reduced further to as little as 5mW by using an ultra-low-power microcontroller instead of a Raspberry Pi [22]. The energy overhead of scene dynamics analysis can be reduced to 50mW with an FPGA-based implementation [22].

VII. Real-World Deployment

A. Deployment Setup

Two body cameras having identical hardware are deployed in multiple real-world scenarios (indoors and outdoors) in two sessions for a total of 271 minutes. In each session, both cameras start and finish recording at the same time. The first camera runs ZenCam algorithms and the other one uses a fixed settings of 1600x900 resolution and 40fps to record high-quality ground-truth video of the scene. This second video footage is later used in our lab to produce two different solutions to compare with the ZenCam. We call these two: *Baseline* and *Oracle*, respectively.

The *Baseline* is obtain by re-recording the high-quality ground-truth video footage (1600x900 @40fps) obtained from the field experiments at a fixed settings of 1920x1080 @30fps – which is a common settings used in most video recorders. Re-recording is performed by playing the footage on a computer screen and then capturing the scene with the same camera used as ZenCam. We use this Baseline solution to compare

IMU	Low			Mid	High	Remaining Battery (%)			File Size (GB)	
Scene Level	Low	Mid	High	NA	NA	Baseline	ZenCam	Oracle	Baseline	ZenCam
Time (min) #1	30	33	34	22	11	29.9	50.8	55.3	5.4	2.8
Time (min) #2	48	43	19	20	14	36.6	58.8	64.8	9.1	4.6

TABLE II: Time duration for each classified activity in minutes, battery remaining as end of system, and size of video files generated.

ZenCam's performance against a typical fixed-configuration camera.

Likewise, the *Oracle* is also obtained by re-recording the high-quality footage as in Baseline, but this time the camera dynamically adapts to different camera settings based on its prior knowledge of the user's context and the scene dynamics obtained from ZenCam's classification results. We use this Oracle solution to quantify ZenCam's computational overhead as these two are identical in their hardware and software, except that the context detection algorithms do not run in the Oracle.

The duration of different activity levels and scene dynamics for both sessions are shown in Table II. The first session lasts for 130 minutes. It includes user's activity such as sitting in front of a desk, walking on corridors, and running outside. The second session lasts for 141 minutes which includes activities such as running and walking outdoors, driving, and sitting in front of a desk. These activity sequences, although may not be overly extensive, are chosen to mimic daily activities of a law-enforcement or a patrol officer, and captures all possible activity and scene types of ZenCam.

(a) Low **(b) Medium** **(c) High**

Fig. 6: Classification of scene dynamics.

We manually inspect the encoded video scene analysis results of ZenCam to verify that they match our expectation. Three sample frames are shown in Figure 6. Figure 6(a) is classified as low dynamics as the street is almost empty. Figure 6(b) is classified as medium dynamics where there is only one car. Figure 6(c) has multiple cars on the street which the system classifies as a scene with high dynamics.

B. Extended Battery Life

In Figure 7, we plot the remaining battery life of the three camera systems: ZenCam, Baseline (1920x1080 at 30 fps.), and the Oracle. At the end of the two sessions, energy savings with ZenCam is 29.8% and 35%, respectively, when compared to the baseline. ZenCam's energy overhead is 10% and 17%, respectively, when compared to the Oracle. The performance difference in the two sessions is expected as the first session contains more high and medium level activities that consume more energy but incurs less overhead due to lower amount of scene dynamics analyses. The second session contains more low level activities which results in a lower energy consumption but results in a higher overhead due to more frequent scene dynamics computation.

C. Storage Efficiency

To compare the storage efficiency, we examine the sizes of the generated video files. All recordings use the same quantization parameter to prevent the system from automatically changing compression ratio which affects the file size. During the first session, the baseline model produces a total of 5.4 GB video files while ZenCam produces 2.8 GB (48.1% reduction). The baseline model generates 9.1 GB file in the second session while ZenCam generates 4.6 GB (49.5% reduction). The difference is due to the second session having more outdoor activities where the spatial complexity is more than the first session. Further discussion on this is in Section VIII. This result demonstrates that our system reduces the storage requirement significantly, and thus reduces the cost of archiving data.

D. Video Quality Analysis

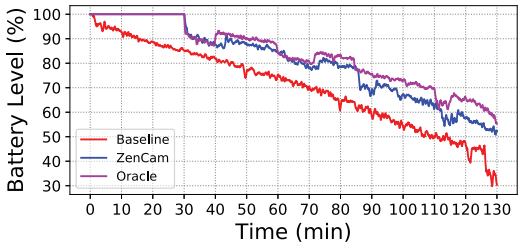
In this experiment, we investigate how well ZenCam preserves the video quality by using the method metrics used in Section IV. Figure 8(a) and Figure 8(b) compares video quality from ZenCam and the Baseline (1080p @30fps) in terms of SSIM and VQM, respectively. Recall that lower VQM scores implies better video quality. We observe that for low and medium scene dynamics, there is no significant difference in the video quality since the frame rate does not affect the quality of the video significantly in a mostly static scene. For high dynamic scenes and medium activity levels, ZenCam produces slightly lower quality videos than the baseline but the difference is small. During high activity level, ZenCam produces better quality videos as the frame rate goes higher than 30 fps. This is expected as high frame rates improve the video quality in a highly dynamic environment.

E. Summary

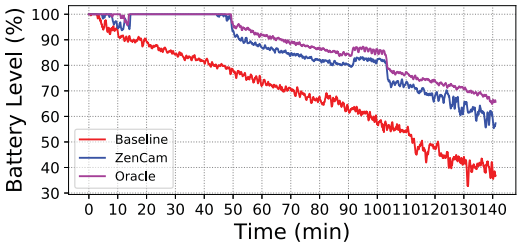
Through a series of empirical evaluation, we demonstrate that ZenCam significantly reduces energy consumption and storage space requirement while maintaining a competitive video quality and outperforms typical fixed configuration cameras. ZenCam reduces energy consumption by trading in minimal video quality. Our evaluation also shows that ZenCam has a lower overhead when the system is at the highest demand – which is a property that no state-of-the-art system has achieved till date. With a more efficient hardware implementation, ZenCam can achieve even higher efficiency and resource savings.

VIII. Discussion

Why the video features are free? The video features we use (i.e., motion vectors and residuals) are generated by the video codec during the video encoding process. Since these two are already computed inside the camera firmware, all we require

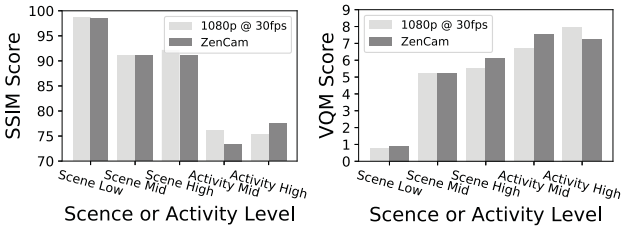


(a) Session #1



(b) Session #2

Fig. 7: Remaining battery life (%) over the duration of experiments.



(a) SSIM Score vs. Environment (b) VQM Score vs. Environment

Fig. 8: Video Quality Analysis for Different Scenes

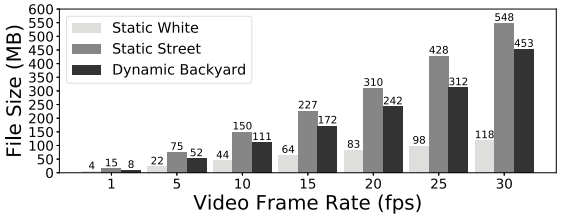


Fig. 9: File Size vs. Frame Rate

is to expose them to the application layer and use them at run-time in the proposed scene dynamics analyzer. Since there is no overhead of frame decoding, feature computation, and frame re-encoding, we are relieved from an expensive step of feature computation at zero cost.

Why not use machine vision techniques? We consider two aspects of machine learning and computer vision: energy consumption and accuracy. Modern computer vision techniques that are based on Deep Neural Networks (DNNs) require cloud and/or GPU machines as opposed to low-power embedded processors [17, 22]. Even specialized vision processing units consume more power than a Raspberry Pi (around 1W) [3]. In contrast, our unoptimized ZenCam implementation having no special hardware adds less than 400mW overhead. Furthermore, current commercial products with machine learning techniques for video capturing such as Google Clips [1] are not on par with average user's expectation [2].

Can existing image processing algorithms be used on video recorded by ZenCam? Since ZenCam generates standard video files, which we didn't modify the standard video compression algorithm been used, existing computer vision and image processing techniques are directly applicable to its output. Unlike [22], ZenCam preserves all information throughout its lifetime.

Does reducing frame rate help saving storage space? We record multiple 3-minute videos varying the frame rate for three different contents displayed on a computer monitor. The result is shown in Figure 9. `Static White` is a purely white picture, `Static Street` is a picture of a street, `Dynamic Backyard` is a short video clip of playing in the backyard. The file size of the `Static Street` is larger than the `Dynamic Backyard` as the street scene has higher spatial complexity[2].

[2]Video encoding happens in both spatial and temporal domain. Our system uses information from the temporal encoding only.

Overall, we see a decrease in the file size as the frame rate decreases. ZenCam saves a significant amount of storage space by reducing the frame rate significantly when the user is static facing a high complexity static scene.

IX. Related Work

A. Continuous Mobile Vision

Prior works [12] have leveraged low-power IMU sensors to control the camera parameters. However, for lack of scene dynamics analysis, these systems fails to record videos at a suitable frame rate when the scene is dynamic. [16, 22] utilize low-power secondary camera sensors such as a thermal imaging sensor and hardware implementation of algorithms using FPGAs to reduce power consumption. But these additional circuitry significantly increases the complexity of the system and lose important information due to misclassification of scene dynamics that is based on a thermal sensor. [22] employs an early discard algorithm to record about 10% frames during the operation. While this approach reduces a significant amount of energy and space, it is not truly a continuous vision system.

B. Video Processing

Video analysis in the compressed domain has been explored in different contexts such as moving object segmentation [21], human action recognition [8], and video classification [7]. These techniques are mostly used in fixed surveillance cameras and are not suitable for mobile continuous vision or body cameras. We refer our readers to [4] for a more detailed review.

The use of scene dynamics analysis is relatable to [14], however, it uses a large number of frames, does not consider the effect of different frame rates, and there is no IMUs involved to infer user context. Our algorithm takes advantage of the IMU sensors and does not rely on many frames – which enables real-time performance.

While it is possible to use Visual inertial odometry (VIO) [24] or simultaneous localization and mapping (SLAM) [13] techniques to construct the 3D environment and then perform scene analysis, this process is too computationally expensive to be implemented on a constrained embedded system and does not provide additional benefit over our framework. Furthermore, utilizing anomaly detectors [6, 19] or motion-triggered recording do not reduce energy consumption as the camera remains always on.

C. Context Recognition

Human Activity Recognition (HAR) has been studied extensively by many. *intra-class variability*, *inter-class similarity*, and *null-class problem* [10] are a few of many open problems in this domain. The methodology for data segmentation, which is required for activity recognition at a certain instant such as a sliding window, energy-based, and additional sensors [5, 10], is crucial to the classification accuracy. After data are processed to extract features, a classifier, e.g., Dynamic Time Warping (DTM) [9], Hidden Markov Models(HMMs) [11] or kNN [18], is used for training and classification. A more detailed review of HAR can be found in [10].

X. Conclusion

In this paper, we design, implement and evaluate ZenCam, an always on body camera that saves system resources via scene dynamics analysis in the encoded video domain and human activity classification using IMU-based sensing. ZenCam uses this information to control camera parameters for energy and storage consumption reduction. Our evaluation shows that ZenCam achieves 29.8-35% energy savings and 48.1-49.5% storage space reduction, while maintaining a competitive or better video quality, when compared to a baseline system having a fixed configuration of 1920x1080 at 30fps.

Acknowledgment

This paper was supported, in part, by NSF grants CNS-1816213 and CNS-1704469.

References

[1] Google clips. https://store.google.com/us/product/google_clips.

[2] Google clips review: A smart, but unpredictable camera. https://www.engadget.com/2018/02/27/google-clips-ai-camera-review/.

[3] Myriad 2 ma2x5x vision processor product brief. https://uploads.movidius.com/1463156689-2016-04-29_VPU_ProductBrief.pdf.

[4] R. V. Babu, M. Tom, and P. Wadekar. A survey on compressed domain video analysis techniques. *Multimedia Tools and Applications*, 75(2):1043–1078, Jan 2016.

[5] O. Banos, J.-M. Galvez, M. Damas, A. Guillen, L.-J. Herrera, H. Pomares, I. Rojas, C. Villalonga, C. S. Hong, and S. Lee. Multiwindow fusion for wearable activity recognition. In *International Work-Conference on Artificial Neural Networks*, pages 290–297. Springer, 2015.

[6] A. Bera, S. Kim, and D. Manocha. Realtime anomaly detection using trajectory-level crowd behavior learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 50–57, 2016.

[7] S. Biswas and R. V. Babu. H. 264 compressed video classification using histogram of oriented motion vectors (homv). In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2040–2044. IEEE, 2013.

[8] S. Biswas and R. V. Babu. Real time anomaly detection in h. 264 compressed videos. In *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013 Fourth National Conference on*, pages 1–4. IEEE, 2013.

[9] U. Blanke, R. Rehner, and B. Schiele. South by south-east or sitting at the desk: Can orientation be a place? In *Wearable Computers (ISWC), 2011 15th Annual International Symposium on*, pages 43–46. IEEE, 2011.

[10] A. Bulling, U. Blanke, and B. Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33, 2014.

[11] A. Bulling, J. A. Ward, H. Gellersen, and G. Tröster. Robust recognition of reading activity in transit using wearable electrooculography. In *International Conference on Pervasive Computing*, pages 19–37. Springer, 2008.

[12] S. Fang, K. Mayer-Patel, and S. Nirjon. Distributed adaptive model predictive control of a cluster of autonomous and context-sensitive body cameras. In *Proceedings of the 2017 Workshop on Wearable Systems and Applications*, pages 35–40. ACM, 2017.

[13] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015.

[14] W. Gillespie and T. Nguyen. *Classification of Video Sequences in MPEG Domain*, pages 71–86. Springer US, Boston, MA, 2005.

[15] M. Graphics and M. L. V. Group). MSU Video Quality Measurement Tool. http://www.compression.ru/video/quality_measure/video_measurement_tool.html, 2017. [Online; accessed 30-Nov-2017].

[16] S. Han, R. Nandakumar, M. Philipose, A. Krishnamurthy, and D. Wetherall. Glimpsedata: Towards continuous vision-based personal analytics. In *Proceedings of the 2014 workshop on physical analytics*, pages 31–36. ACM, 2014.

[17] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 123–136. ACM, 2016.

[18] K. Kunze, M. Barry, E. A. Heinz, P. Lukowicz, D. Majoe, and J. Gutknecht. Towards recognizing tai chi-an initial experiment using wearable sensors. In *Applied Wearable Computing (IFAWC), 2006 3rd International Forum on*, pages 1–6. VDE, 2006.

[19] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):18–32, 2014.

[20] S. Liu, M. Li, S. Zhu, and B. Zeng. Codingflow: Enable video coding for video stabilization. *IEEE Transactions on Image Processing*, 26(7):3291–3302, July 2017.

[21] C.-M. Mak and W.-K. Cham. Real-time video object segmentation in h. 264 compressed domain. *IET image Processing*, 3(5):272–285, 2009.

[22] S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan. Glimpse: A programmable early-discard camera architecture for continuous mobile vision. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 292–305. ACM, 2017.

[23] S. Nirjon, R. F. Dickerson, Q. Li, P. Asare, J. A. Stankovic, D. Hong, B. Zhang, X. Jiang, G. Shen, and F. Zhao. Musicalheart: A hearty way of listening to music. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 43–56. ACM, 2012.

[24] S. Omari, M. Bloesch, P. Gohl, and R. Siegwart. Dense visual-inertial navigation system for mobile robots. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2634–2640, May 2015.

[25] Y.-F. Ou, T. Liu, Z. Zhao, Z. Ma, and Y. Wang. Modeling the impact of frame rate on perceptual quality of video. In *2008 15th IEEE International Conference on Image Processing*, pages 689–692, Oct 2008.

[26] M. H. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, Sept 2004.

[27] I. E. Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[28] L. Wang. *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[29] 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*, 13(4):600–612, April 2004.

[30] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hossfeld. Impact of frame rate and resolution on objective qoe metrics. In *2010 Second International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 29–34, June 2010.