FINAL REPORT

Application Aware Approach to Compression and Transmission of H.264 Encoded Video for Automated and Centralized Transportation Surveillance

October 12, 2012

Zhaofu Chen

Eren Soyak

Sotirios A. Tsaftaris

Aggelos K. Katsaggelos

CCITT | 600 Foster Street | Evanston, IL 60208
Tel: 847-491-7287 | Fax: 847-491-3090
http://www.ccitt.northwestern.edu/

This work was funded by the Center for the Commercialization of Innovative Transportation Technology at Northwestern University, a University Transportation Center Program of the Research and Innovative Technology Administration of USDOT through support from the Safe, Accountable, Flexible, Efficient Transportation Equity Act (SAFETEA-LU).

Application Aware Approach to Compression and Transmission of H.264 Encoded Video for Automated and Centralized Transportation Surveillance

Abstract

In this report we present a transportation video coding and wireless transmission system specifically tailored to automated vehicle tracking applications. By taking into account the video characteristics and the lossy nature of the wireless channels, we propose video preprocessing and error control approaches to enhance tracking performance while conserving bandwidth resources and computational power at the transmitter. Compared with the current state-of-the-art H.264-based implementations our system is shown to yield over 80% bitrate savings for comparable tracking accuracy.

Index Terms

Transportation video, object tracking, surveillance centric coding, preprocessing, forward error correction (FEC), error concealment, H.264/AVC,

I. Introduction

Remote imaging sensors are commonly deployed for transportation monitoring and surveillance applications. Imaging sensors present a unique set of advantages over other conventional sensing modalities such as embedded inductor cables and radars. The captured videos can be used for applications such as tracking, as well as for real-time monitoring of traffic conditions [1]–[4].

To reduce infrastructure costs associated with the deployment of such sensors and increase urban coverage, most video imaging solutions require the transmission of video to a centralized location for viewing, (automated) analysis, and/or archiving. Remote nodes are deployed solely for the purpose of video capture and transmission, and hence should have low cost which implies limited computational power. Transmitting high-quality real-time video requires communication channels with bandwidth on the order of mega-bits per second per channel. The requirement for high bandwidth can only be fulfilled with wired links, which are both cost-ineffective and inflexible. Alternatively, video compression technologies can be applied at the remote sensors such that the compressed bitrate can be accommodated by the modern wireless communication channels.

Recently the state-of-the-art video compression technology H.264 has been proposed for transportation video related applications, and has significantly reduced the bandwidth requirement [5]. Most of the systems currently in use are not specifically optimized for transportation videos and employ generic video encoding procedures to remove redundant information from the source videos. Such information removal is based on numerical quality metrics such as peak signal to noise ratio (PSNR) or structural similarity (SSIM) [6].

Although several approaches have been proposed to address compression in surveillance and monitoring applications, these approaches are not application aware (i.e., they still optimize PSNR or similar criteria), may not be standard compliant, and do not take into account possible channel losses [7]–[9].

Recently, in [10], it was shown that it is possible to further reduce bitrate by focusing resources in an application aware context (automated tracking accuracy was used as the metric). While remaining standard compliant it was shown that computationally simple temporal and frequency filtering operations could maintain high tracking accuracy, with reduced bitrate.

Another challenge faced by the transportation video transmission system is the lossy nature of the wireless channels. The highly dependent H.264 bitstreams are sensitive to channel degradations, suffering error propagation due to the predictive coding structure. There is significant interest in resource-distortion optimization given channel losses and error concealment strategies [11], [12]; however, these works, as others, focus on maximizing PSNR and do not consider the accuracy of object tracking. While the human visual system has the capability to mask certain artifacts introduced by packet losses and their concealment, the effects of such artifacts on automated video analysis has not been explored. For example, error concealment that ignores motion information will confuse trackers. Thus it is important to consider approaches that use some of the bitrate to add resilience to errors in an application aware compression and transmission environment.

In this report, we extend the work in [10] and [13] and propose a transportation video transmission system that integrates components at both the transmitter and receiver sides to increase performance (tracking accuracy) while minimizing bitrate, given channel losses. The proposed system focuses on consolidating encoded bits on important video information and protecting it with error mitigation techniques. Specifically, at the transmitter, we propose a preprocessing component with the goal of removing the signal components unimportant for tracking from the video and thus decreasing bitrate, while being computationally efficient. To increase resilience to errors, we propose to use forward error correction (FEC) to minimize the overall loss probability of important video content (in our case moving objects) while conserving bandwidth resources. We consider both equal error protection (EEP) and unequal error protection (UEP) and provide the details on the implementation and their merits. At the receiver side, we discuss two classes of error concealment (ERC) strategies and how they affect tracking accuracy, and conclude that the temporal-domain motion copy (MC) algorithm is best suited for transportation video decoding.

The contributions of this report are centered on optimizing:

- bit allocation within video frames
- error protection schemes for video packets; and
- concealment strategies in case of losses, while
- maximizing tracking accuracy at the receiver's end, and
- minimizing computational load at the encoder's side.

Although other works discuss video quality-rate tradeoffs [11], [12], this is the first treatment of identifying beneficial error mitigation and concealment strategies from the viewpoint of the performance of automated video analysis.

The rest of this report is organized as follows. In Section II we provide a high-level overview of the proposed system. Section III presents the performance metrics used throughout this report. Section IV contains detailed descriptions of the proposed system components, as well as discussions of their relative merits and limitations. In Section V we present experimental results using real-life test videos and demonstrate the effectiveness of our proposed system. Our proposed systems yield significant performance improvement over the state-of-the-art implementations and is shown to allow for over 80% reduction in bitrate for comparable tracking accuracies. Finally the report is concluded in Section VI.

This report is based on work published in [14], [15].

II. System Overview

We propose an application-specific video transmission system that aims at maintaining tracking performance at the central office given the lossy nature of the wireless communication channels. The complete system design includes the processing at both the transmitter side and the receiver side. The system architecture is illustrated in Figure 1.

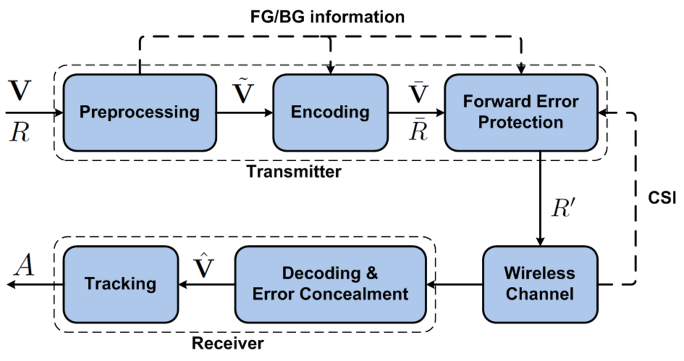


Fig. 1: System block diagram

At the transmitter side, the goal is to identify the video content that is most relevant to tracking and to consolidate the encoded bits on such information. This is achieved by preprocessing the input uncompressed video $\mathbf{V}$ to remove the signal components of low tracking interest. The preprocessor identifies motion of interest in the input video and classifies image regions associated with such motion as foreground (FG), and conversely other image regions as background (BG). The core operation of this process is to alter the signal composition in the BG such that a subsequent standard compliant generic video compressor can encode it with a minimum number of bits. For example, the preprocessed video $\tilde{\mathbf{V}}$ contains background regions exhibiting little motion, which can be efficiently encoded as the skipped macroblocks (MBs) using H.264. The goal is to obtain an encoded standard-compliant bitstream $\bar{\mathbf{V}}$ which is associated with bitrate $\bar{R}$ significantly smaller than the raw uncompressed bitrate R.

This preprocessing step also aids in the channel protection step that will follow in two ways. Indirectly, by intelligently allocating the encoded bits, it allows for the bitrate savings to be used for added redundancies to the encoded bitstreams such that the transmitted video information is more resilient to channel degradations. More directly, the FG/BG information from the preprocessing step serves as a useful guideline for unequal error protection, which is a channel protection option that assigns higher protection levels to the foreground than to the background.

For channel protection, we specifically consider forward error correction in this work. For transportation surveillance applications, the continuously captured video needs to be transmitted back to the central office in real time (due to application constraints or due to low buffering capabilities), which imposes stringent delay requirements. FEC is suitable for such a scenario because it has more controllable delay characteristics. FEC utilizes the channel state information (CSI) feedback from the channel to determine the appropriate protection schemes. Let FEC denote the channel protection parameters. The bitrate of the protected bitstream is consequently increased to $R'(\bar{R}, FEC)$, which is a function of both the base encoded bitrate

$\bar{R}$ and the protection scheme FEC. The preprocessor, encoder and FEC component comprise the transmitter module of the system. More details about the design will be given shortly and we will also show how they influence the system performance.

The received bitstream, with possible transmission errors is decoded with error concealment techniques to yield the reconstructed video sequence $\hat{\mathbf{V}}$, which is used as input for subsequent applications such as tracking [16]. Note that $\hat{\mathbf{V}}$ the reconstructed video at the receiver is different from $\mathbf{V}$ the captured raw video at the transmitter because (1) the video encoder employs transform coding and quantization, which alter the video signal composition; (2) the lossy wireless channels introduce degradations including packet losses and bit errors to the transmitted bitstream; and (3) the decoder employs error concealment algorithms to estimate the lost information. Without proper handling, the discrepancies between $\hat{\mathbf{V}}$ and $\mathbf{V}$ may severely affect the tracker performance . The design objective of the system presented herein is to achieve the optimal balance between tracking performance and bandwidth utilization, while always maintaining a low-computational profile at the encoder. In the following paragraphs, we will introduce our approach in detail and compare it with current state-of-the-art implementations.

III. System Performance Metrics

In our system of transportation video surveillance, the ultimate application is to track (automatically) the objects (e.g. vehicles) in the video and to perform subsequent operations based on the tracking results. Therefore, objective metrics are necessary to quantify and optimize the tracking performance of the overall system. To quantitatively analyze the impact of the various operations on the video quality, we compare the trajectories of the modified video with those of the input raw video $\mathbf{V}$, as is depicted in Figure 2. Here the modified video can be outputs at the different stages of the system, such as the preprocessed video $\tilde{\mathbf{V}}$, the encoder reconstruction $\bar{\mathbf{V}}$ and the decoded video after transmission $\hat{\mathbf{V}}$. Since both tracking and trajectory analysis are nonlinear operations, the relationship between the changes in tracking accuracy $\mathbf{A}$ and the video modifications is rather complicated and to the best of our knowledge there is no effective analytical estimator so far.

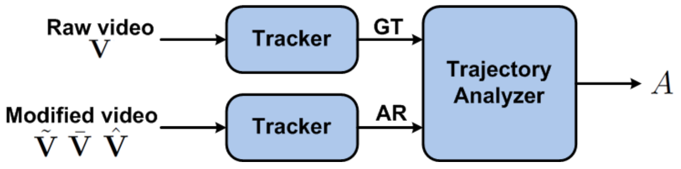

Raw video $\mathbf{V}$
Tracker
GT
Trajectory Analyzer
A
Modified video $\tilde{\mathbf{V}}$ $\bar{\mathbf{V}}$ $\hat{\mathbf{V}}$
Tracker
AR

Fig. 2: Computation of tracking accuracy

There exist a multitude of tracking performance metrics and the associated trajectory analysis approaches, with various solutions offering strength/weakness combinations suitable for different applications. In [17], a review of the state-of-the-art for video surveillance performance metrics is presented. We choose the Overlap, Precision and Sensitivity metrics presented in [10] and reintroduce them here for completeness due to their pertinence to the transportation tracking application. We denote the trajectory obtained using the input raw video $\mathbf{V}$ as the ground truth (GT) and the trajectories obtained using the modified video as the algorithm result (AR). The three accuracy components are defined as follows.

Overlap ($OLAP$) is defined in terms of the ratio between the intersection and union of the frame pixels covered by object segmentations in ground truth and algorithm result, averaged over all detected objects,

$$OLAP = \frac{1}{G}\sum_{g=1}^{G}\frac{GT_g \cap AR_g}{GT_g \cup AR_g}, \tag{1}$$

where GT_g and AR_g represent the g^{th} object tracked in the raw video and the modified video, respectively, $\cap$ and $\cup$ denote the intersection and union of the frame regions covered by the objects, respectively. The $OLAP$ metric is normalized with respect to the total number of objects detected in both the raw video and the modified video.

Define TP as the number of true positives and FP as the number of false positives, where a true positive corresponds to an object tracked in both the input video and the modified video while a false positive corresponds to an object tracked only in the modified video but not in the input video. Precision ($PREC$) defined as

$$PREC = \frac{TP}{TP + FP} \tag{2}$$

measures how credible the algorithm result is in terms of object identity.

Similarly, a false negative is an object tracked in the input raw video but not in the modified video. Let FN denote the number of false negatives. Sensitivity ($SENS$) is defined as

$$SENS = \frac{TP}{TP + FN}\ . \tag{3}$$

In order to jointly optimize for a combination of the above metrics, the tracking accuracy $\mathbf{A}$ is defined as a convex combination of $OLAP$, $PREC$ and $SENS$.

IV. Proposed System Components

A. Preprocessor

As is introduced in the previous section, the preprocessing step serves two purposes: (1) to filter the input video to consolidate encoded bits on video information that is most important for tracking; (2) and to provide a guideline for the channel protection component if requested. To fulfill such purposes we present a preprocessing module illustrated in Figure 3. This "two branch" design jointly serves the two aforementioned purposes. The lower branch analyzes the input video and identifies per-frame background regions, in which it suppresses the intensity variations with the goal of minimizing the encoded bitrate and maintaining tracking performance. The upper branch consists of an optional region-of-interest (ROI) extraction component that explicitly provides foreground/background separation information to the forward error correction component. This ROI extraction is optional since it is only used for explicit unequal error protection, which will be discussed in the following. For equal error protection this FG/BG discrimination is not necessary, i.e., the switch is left open. In the following we will discuss the details of each subsystem.

1) TDT Filtering: The temporal filtering process is performed using the temporal deviation thresholding (TDT) algorithm introduced in [10]. The purpose of this filtering process is to both suppress intensity variations irrelevant to tracking and hence reduce bitrate, and to improve tracking accuracy by removing signal components potentially distracting to tracking applications. At reasonable compression ratios the intensity variations due to capture noise or local illumination changes are

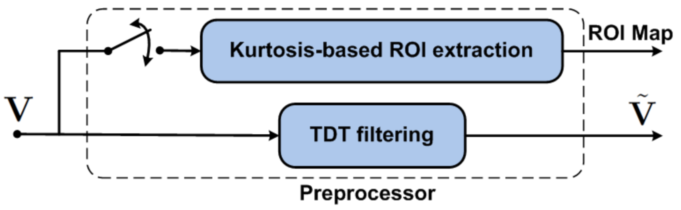


Fig. 3: Preprocessing block diagram

usually imperfectly represented in the compressed video, which, after decoding may contain artifacts that are misleading to the tracker. For example, such noise-like variations may be sampled infrequently relative to the frame rate, and can result in a seemingly random block update in the decoded video. These block updates may be interpreted as motion and hence may trigger the tracking mechanism (thus either decreasing accuracy or increasing the complexity of the tracker by requiring particular treatment of such spurious detections). The TDT algorithm seeks to suppress such intensity changes before encoding, thereby both reducing the bitrate and improving the tracking accuracy.

The TDT algorithm is based on the assumption that for transportation surveillance video obtained from a static camera, the majority of pixels undergo temporal variations due to noise. Therefore a video frame is predominated by a static background where intensity changes are mainly due to capture noise and local illumination changes rather than actual motion. In the TDT algorithm, noise-like intensity variations are modeled as a Gaussian process with zero mean and variance σ_t^2, where the subscript t is the time index. In our proposed system, we estimate the noise variance from both the temporal and spatial domain, by examining the temporal intensity changes of all pixels within the t^{th} frame and seeking the most probable one.

Specifically, let $\mathbf{F}_t$ denote the t^{th} frame in the uncompressed raw video $\mathbf{V}$. At any time instance, picture statistics computed from the T most recent frames are maintained. The per-pixel standard deviation is computed and a histogram is constructed from all these values. Since we assume most pixels undergo intensity variations due to noise or local illumination changes, the mode of the histogram can be used as an estimator of the noise standard deviation. The reader is advised that the way we estimate the noise is a suggestion and there are other (potentially more efficient and accurate) options that can replace this step. The estimated noise standard deviation $\hat{\sigma}_t$ is used as a threshold to classify the inter-frame intensity changes as either of tracking interest or not. Subsequently a frame mask $\mathbf{M}_t$ is obtained according to the classification which reveals the image regions associated with actual motion and covers the ones associated with noise or local illumination changes. Mathematically, the TDT algorithm operates iteratively as follows:

$$\begin{aligned}
&\hat{\sigma}_t = mode(std(\{\mathbf{F}_{t-T+1}, \ldots, \mathbf{F}_t\})) \\
&\boldsymbol{\Delta}_t = |\mathbf{F}_t - \mathbf{F}_{t-1}| \\
&\mathbf{M}_t = \boldsymbol{\Delta}_t > \tau\hat{\sigma}_t \\
&\tilde{\mathbf{F}}_t = \mathbf{M}_t \star \mathbf{F}_t + (1 - \mathbf{M}_t) \star \tilde{\mathbf{F}}_{t-1}
\end{aligned} \quad , \tag{4}$$

where $\star$ denotes the element-wise multiplication of two matrices with the same dimensions and $\tau > 0$ is a threshold multiplier that can be selected based on the tail probability of the Gaussian distribution. For example, letting $\Phi(\cdot)$ denote the cumulative distribution function (CDF) for a standard Gaussian random variable, we have

$$Prob(\Delta_t(n_1, n_2) < \tau\sigma_t) = 2\Phi(\tau) - 1\ . \tag{5}$$

If we assume 95% of the intensity variations are not due to actual motion, we can select τ to be 2. The output of the TDT filtering is in the original format as the raw video, which is then used as the input to the encoder. Since the irrelevant intensity variations in the background have been suppressed by the TDT algorithm, it was shown that the filtered video yields a bitrate reduction compared with unfiltered video [10]. Moreover, since the information removed from the raw video has little tracking interest, its removal is beneficial yielding a better rate-accuracy tradeoff.

2) Kurtosis-Based ROI Extraction for UEP: Although it will be presented in detail in a following paragraph, unequal error protection assigns different levels of protection to different encoded video packets. The goal is to assign more protection to video packets containing moving objects. As we will see later, we implement this using the flexible macroblock ordering (FMO) option in H.264, which essentially requires a relatively static and homogenous map of FG/BG that is valid for a large number of frames (ideally the whole sequence). Update to this map costs additional bits to be added to the bitstream. The requirement therefore is to estimate a map (we term it here ROI) automatically which remains valid for as many frames as possible (thus minimizes misclassification).

From the discussion above, TDT estimates a mask $\mathbf{M}_t$ for every frame, which would prohibit its direct use as an ROI. Multiple $\mathbf{M}_t$ masks must be combined in an intelligent fashion to minimize misclassification. Recall that TDT is a pixel level process that aims at reducing the effects of acquisition noise and thus may result in an increased number of false positives. Identifying and eliminating such errors would require several computationally intensive higher-order processing steps. Here rather than trying to combine masks from TDT we rely on another non-parametric algorithm that provides a homogeneous ROI map.

The classification of FG/BG for the ROI is based on a non-parametric model of the temporal distribution of pixel intensities that was introduced in [18] for the purpose of context aware compression. The goal there was to isolate the regions showing events of high tracking interest (e.g., vehicles moving in streets) from regions undergoing periodic changes (such as waving trees, water fountains, or light reflections) and focus bitrate in those regions. Note that TDT does not have the capability of separating such type of motion. Here we adopt this approach for the purpose of FEC and UEP and we reintroduce it for completeness with notation in agreement with the current manuscript. Specifically, let $F_t(n_1, n_2)$, denote intensity of a pixel located at (n_1, n_2) in the t^{th} frame. In order to detect the ROI, we use the excess kurtosis of intensities for each pixel position over time, defined as:

$$\kappa(n_1, n_2) = \frac{\frac{1}{T}\sum_{t=0}^{T-1}(F_t(n_1, n_2) - \bar{F}(n_1, n_2))^4}{(\frac{1}{T}\sum_{t=0}^{T-1}(F_t(n_1, n_2) - \bar{F}(n_1, n_2))^2)^2} - 3, \tag{6}$$

where $\bar{F}(n_1, n_2)$ is the mean value of the intensities over the training length T. Note that in order not to clutter the notation we use T to denote both the buffer length in the TDT algorithm and the summation extent in the ROI extraction module. Also note that T should be less than the length of the entire sequence. In practice, if the scene is relatively fixed (e.g., the camera is mounted on a pole), a single ROI can be used for a relatively long time, until significant scene change occurs. Therefore, the switch before the "ROI Extraction" block in Figure 3 is closed only during the initialization (and re-initialization if necessary) phase(s) of operation. We should emphasize that the ROI is not updated on a frame-by-frame basis, and hence has a relatively low amortized computational complexity.

By definition, a Gaussian distribution has an excess kurtosis of 0. Furthermore, the additive property of kurtosis implies that a mixture of Gaussians also has an excess kurtosis of 0. Since the capture noise is modeled as additive Gaussian, and the periodic movements of objects (such as trees) can be modeled as a mixture of Gaussians, they both can be characterized as having an excess kurtosis of 0 [18]. The desired type of motion due to events such as moving vehicles can be modeled as a Poisson process, which has an excess kurtosis of 6. Based on the above discussion, we can build the ROI map of pixels by thresholding their excess kurtosis values. For computational reasons, the threshold is set to a fixed value of 3, the middle point between the excess kurtosis values of the two models. Once the pixels are classified, the classification of macroblocks can be done based on a majority vote rule, i.e., the MB class label (FG or BG) is determined by the majority class of the pixels within that MB.

Since the kurtosis-based ROI extraction can utilize higher-order statistics to associate the intensity variation with different types of motion, it can potentially generate an ROI that highlights the most important image regions. This ROI map can be used as a guideline for explicit unequal error protection, which treats the foreground and background regions differently by assigning higher protection levels to the more important foreground. Furthermore, this UEP treatment requires the grouping of MBs according to their class labels, which can be fulfilled by using the FMO feature of the H.264 encoder. Therefore, the ROI map generated herein is fed into both the encoder and the FEC module, as is illustrated in Figure 1.

3) Discussion on preprocessing: As is introduced in the previous section, the preprocessing step serves the purpose of filtering the input videos for bitrate consolidation and providing when needed a guideline for channel protection in an UEP scheme. In order to achieve these goals, we propose TDT for video filtering and a kurtosis-based approach for ROI extraction. As we have seen from the previous sections, both TDT and kurtosis-based ROI extraction target at identifying video components of high tracking interest. However, each algorithm implements this at a different scale.

TDT is essentially a pixel-level noise filtering process, and considers any pixel that is above a certain threshold as of interest. This process is repeated at each frame and the goal is not to spend bits on encoding noise-like intensity variations. Thanks to its fast adaptivity, TDT is less sensitive to misclassifications as it has the opportunity to correct misclassified pixels on a frame basis. In this way TDT ensures the encoded bits are spent on the most important pixels. However, TDT cannot distinguish between pixels undergoing periodic motion (e.g., swaying trees) versus a passing vehicle. In contrast the kurtosis-based ROI extraction is a non-parametric area of interest extraction algorithm that aims at identifying a region (ideally the road infrastructure) where aperiodic motion is mostly likely to occur. This ROI is not adapted at every frame and is estimated during an initialization phase that is longer than the number of frames needed in TDT. In fact, a static ROI map is desirable from an implementation viewpoint. Due to limitations of the H.264 standard and how flexible macroblock ordering is designed which was alluded previously and will become apparent in a later paragraph, the ROI map cannot be updated at every frame and must remain static for as long as possible, since updating it incurs bitrate cost. Thus, the choice of how up to date an ROI map is constitutes a choice of bitrate savings: one could use the mask $\mathbf{M}_t$ of TDT as an ROI map for UEP, but this map would need to be updated frequently which would cost bits, or could use a single map from several TDT masks (e.g., an OR combination of multiple $\mathbf{M}_t$ for an extended duration) to minimize the need for an ROI update but risk degrading accuracy. Therefore in our system we propose to use a separate kurtosis-based non-parametric approach to solve this problem. In the results section we will discuss the above points further with some examples.

B. Encoder

In this system we use the state-of-the-art video compression technology H.264 as the source encoder [5]. The input to the encoder is the TDT-filtered video in its original format (YUV in our experiments), and the output of the encoder is a standard-compliant bitstream. The video frames are divided into slices, each consisting of a set of MBs. Each slice is independently decodable and can be transmitted as an individual unit.

As mentioned above, if the ROI map is present and UEP is desired, the FMO option is enabled in the H.264 encoder. Specifically, we divide the video frame into two slice groups, with one for the foreground regions and the other for the background regions. The slice groups cover potentially disconnected image regions, and with FMO we can assign non-consecutive macroblocks to the same slice (within the same slice group). This feature adds flexibility in the encoded bitstream and facilitates the application of unequal error protection. Since the SG mapping information is conveyed in the picture parameter set (PPS) network abstraction layer (NAL) unit as specified by H.264 and one PPS is referenced by a set of frames (possibly all frames within a video sequence), the information contained in PPS is relatively stationary and does not allow fast per-frame changes. Any necessary update in the PPS due to a new ROI will cost bits. Thus, how up to date the FG/BG information for the purpose of UEP is limited by the H.264 standard and by bitrate requirements.

C. Forward Error Correction

The output of the H.264 encoder is a set of NAL units, where each NAL unit contains a slice of encoded macroblocks. Encoded slices are encapsulated into individual packets. Therefore, we will use “packets” and “slices” interchangeably.

The forward error correction module improves error resilience of the transmitted packets by adding redundancies in the encoded bitstreams. There exist various approaches for adding redundancies, including both intra-packet FEC and inter-packet FEC [16], [19]. Considering the limited computational resources at the remote node, we propose a simple yet effective channel protection methodology using redundant slices (RS) to minimize the overall packet loss probability in the wireless transmission.

Since the multiple copies of an encoded slice are transmitted independently, a slice can be decoded without error as long as at least one copy is received intact. Let C denote the number of copies for a packet and $\{e_c\}_{c=1}^{C}$ the loss probabilities for the copies. Therefore, the overall loss probability e of the protected packet is the product of loss probabilities of all the copies, given as follows:

$$e = \prod_{c=1}^{C} e_c \, . \tag{7}$$

If the channel is characterized by an independently and identically distributed (IID) fading model, then we have

$$e_1 = e_2 = \cdots = e_C, \tag{8}$$

and

$$e = e_1^C \; . \tag{9}$$

Hence for channels characterized by an IID fading model, the RS strategy linearly increases bandwidth usage, yet exponentially decreases the overall packet loss probability.

With RS as the FEC scheme, the question at hand is how to determine the protection level C. As is mentioned in the previous paragraphs, based on the choice of C, the protection schemes can be broadly categorized as either equal error protection or unequal error protection. EEP assigns a uniform protection level to all the encoded packets, while UEP assigns possibly different protection levels to different packets.

The apparent advantage of EEP is its inherent simplicity. In many cases since the TDT filtering step has suppressed the video signal of low tracking interest, our system can be considered to have an “indirect” UEP. Although we assign the uniform protection level to all the encoded packets in this case, significantly higher bitrate is devoted to the important video signal than to the intensity variations due to noise and local illumination changes. In practical systems, given the available bandwidth and the channel conditions, joint source-channel coding can be employed to determine the encoded bitrate and the protection level C. For example, if the channel condition is good, more bits should be spent on source encoding to achieve better video quality. Otherwise, if the channel is very lossy, using low bitrate and high protection level can potentially achieve better result.

In the other cases when the input frames exhibit complex motion of mixed types including actual object movement, background light reflection, and periodically changing objects, UEP and the proposed kurtosis-based ROI extraction (Section IV-A2), have the potential to distinguish the motion types and make better utilization of the bandwidth resources. The assignment of protection levels (in terms of the number of copies transmitted for a packet) constitutes a trade-off between the effective bitrate and the overall loss probability.

Given the estimated ROI map (Section IV-A2), we divide a video frame into slices in such a way that all the MBs in a single slice belong to the same group. For foreground slices, we assign the protection level $H = 1, 2, \cdots$, and for background slices, we assign the protection level $L \leq H$. The values of H and L can be selected based on the maximum supported bitrate $\hat{R}$ and a target overall loss probability P_{target}. Specifically, let $\mathcal{I}_H$ denote the set of foreground slices, and let $\mathcal{I}_L = \{0, 1, \cdots, I-1\} \backslash \mathcal{I}_H$, where I is the total number of slices and “\” is the set difference operator. Let R_i denote the size of the i^{th} slice. The pseudocode shown in Algorithm 1 can be carried out to determine H and L.

Algorithm 1 Determining UEP Error Protection Levels

```
1: $c_{\text{target}} = \left\lceil \log_{P_{\text{unprot}}}(P_{\text{target}}) \right\rceil$ ; /* $P_{\text{unprot}}$ is the unprotected packet loss probability */
2: **if** $c_{\text{target}} \sum_{i \in \mathcal{I}_H} R_i + \sum_{i \in \mathcal{I}_L} R_i > \hat{R}$ **then**
3:     $H = \left\lfloor (\hat{R} - \sum_{i \in \mathcal{I}_L} R_i) / \sum_{i \in \mathcal{I}_H} R_i \right\rfloor$ ;
4:     $L = 1$;
5: **else**
6:     $H = c_{\text{target}}$;
7:     $L = \left\lfloor (\hat{R} - c_{\text{target}} \sum_{i \in \mathcal{I}_H} R_i) / \sum_{i \in \mathcal{I}_L} R_i \right\rfloor$ ;
8: **end if**
```

The underlying assumption here is that $\hat{R}$ can support at least one copy of each packet (in either group). If this assumption is violated, then we can prioritize important packets and drop the less important packets first as discussed in [20].

D. Decoder and Error Concealment for Tracking

In our case by design of FEC a lost packet will not be retransmitted; consequently the information contained in the lost packet must be estimated in the decoding process. The estimation of the lost video content using the reconstructed video content available in the decoded picture buffer (DPB) is known as error concealment (ERC). Although ERC strategies have been developed in the context of video transmission their effects on tracking performance (and thus their use in application-aware environments) have not been studied. Here we present a few of the most commonly used strategies and discuss their relative merits in our application.

In general, ERC works by utilizing the spatial or temporal correlation between the lost information and its neighbors [21]. A typical example of ERC scheme utilizing spatial correlation is the boundary matching algorithm (BMA) [22], which is implemented in the JM H.264 reference model. The algorithm works by interpolating the video content from reliably reconstructed spatial neighbors into the region with information loss. The interpolation option is selected to minimize the discrepancies of the boundaries surrounding the lost region. Pseudocode for the BMA strategy is shown in Algorithm 2.

Algorithm 2 Boundary-Matching Error Concealment Algorithm

```
1: $predMBStatus = CollectPredMBStatus(currMBLocation, MBLossMap, frameWidth, frameHeight)$;
2: for all $block$ in the current MB do
3:     $minDist = BIGNUMBER$;
4:     $bestNeighbor = 0$;
5:     for all $neighbor$ of $block$ do
6:         $motionInfo[block] = motionInfo[neighbor]$;
7:         $predRec[block] = BuildPredRec(DPB, motionInfo[block])$;
8:         $currDist = CalcEdgeDist(predRec[block], distMetric)$;
9:         if $currDist \leq minDist$ then
10:            $minDist = currDist$;
11:            $bestNeighbor = neighbor$;
12:        end if
13:    end for
14:    $reconMB[block] = BuildPredRec(DPB, motionInfo[bestNeighbor])$;
15: end for
```

Besides spatial types of ERC (e.g., the BMA), there are ERC schemes making explicit use of temporal correlation. A straightforward but intuitive example in this category is the motion-copy (MC) algorithm [23]. The MC algorithm copies the motion information from co-located MBs in the previously decoded frames and then reconstructs the pixel values using such estimated motion information. In a simple case, the MC algorithm copies the reference picture index from the previously decoded frame, and then scales the motion vectors accordingly. Pseudocode for the MC strategy is shown in Algorithm 3.

Any mismatch between the reconstructed frame and its encoded variant prior to errors (known as encoder/decoder mismatch) creates severe artifacts (e.g., flashing/unsmooth macroblocks, errors in motion consistency, intensity variations). In transportation videos, it is reasonable to assume that objects of tracking interest exhibit smooth and consistent translational motion throughout consecutive frames. In such scenarios, encoder/decoder mismatch creates trailing artifacts that are particularly distracting to trackers and lead to decreased accuracy. Thus, it is desirable to identify ERC strategies that minimize such type of errors.

Algorithm 3 Motion-Copy Error Concealment Algorithm

```
1: for all $block$ in the current MB do
2:     $motionVector[block] = ScaleMotionVec(motionVector[colocatedBlockInPrevFrame]);$
3:     $refPicIdx[block] = refPicIdx[colocatedBlockInPrevFrame];$
4:     $motionInfo[block] = \{motionVector[block], refPicIdx[block]\};$
5:     $reconMB[block] = BuildPredRec(DPB, motionInfo[block]);$
6: end for
```

The MC algorithm is potentially able to accurately estimate the motion information due to its use of the motion vector (line 2) in the pseudocode list. The MC algorithm is particularly suitable for the preprocessed video, because the pixel variations not due to translational object motion have been suppressed by TDT. With the MC algorithm, the scaled motion vector and extrapolated reference frame index provide reliable reconstruction of the lost video information using its temporal predecessors. In the numerical examples below, we will show several examples of such artifacts and demonstrate that the MC algorithm indeed outperforms the spatial BMA scheme, in terms of tracking accuracy improvement and show that use of BMA leads to an increase in false positives.

V. Experiments

A. General Framework

To verify the gains made possible by our proposed schemes, we conduct experiments using multiple sequences with different characteristics such as viewing angles, quality and type of observed vehicle traffic. Details of the sample implementation and experimental procedure with the test results are presented below.

The following video sequences are used for the experiments. The “Camera6” sequence (resolution 640×480) [24] used under the NGSIM license courtesy of the US FHWA shows an intersection with light traffic, with trees swaying in the wind and buildings casting reflections of passing cars as part of the scene. The “dt_passat” sequence (resolution 768×576) [25] by courtesy of KOGS/IAKS Universität Karlsruhe shows a busy intersection with steady traffic interrupted by a traffic signal and a light urban rail crossing. Both sequences contain significant capture noises. Both sequences consist of 600 raw video frames in YUV420 format. For videos captured at real-time (e.g. 25 Hz), this amounts to uncompressed bitrates on the order of hundreds of Mbits/sec, which far exceeds the capacity of current wireless channels. Therefore videos must be heavily compressed prior to transmission.

To implement the proposed system the open-source JM 16.2 [26] encoder is used. When UEP is employed and the ROI map is present, FMO is enabled and the ROI map is used to allow for packetization of the two different slice groups. Otherwise, the packets are to be uniformly protected and hence no FMO is needed. Unless otherwise noted, in the experiments we employ the packetization strategy of maximum slice size (in terms of encoded bytes). At the receiver side the JM decoder is modified to enable the MC strategies. The JM decoder has a built-in BMA for error concealment, and it is used for performance evaluation. The open-source OpenCV 2.3 [27] “blobtrack” module is used as the object tracker which relies on the mean shift object tracking algorithm [28].

In the following experiments, we consider an IID channel model, which results in independent losses of the transmitted packets. Although it is an ideal model, it provides a reasonable approximation to real channel conditions under certain assumptions. One such assumption is the channel conditions vary much faster than the transmission data rate. In that case the effect of channel variation is smoothed over the transmission of packets. In addition, if packet interleaving is possible then the multiple copies of an encoded slice can be transmitted at disperse time instances. If the temporal distance is sufficiently large compared with the channel memory, the channel conditions can be reasonably approximated as being IID. As an example, consider the transportation videos captured and transmitted at constant frame rate of 25 frames per second (FPS). If our system allows maximum latency of 5 frame intervals, and we interleave the transmission of different copies of a packet by this amount of time, then the temporal distance between two copies of the same packet is 200 ms. Since the coherence time of modern wireless communication channels is typically on the order of a few milliseconds [29], [30], IID channel model is a reasonable approximation.

B. Effect of Preprocessing

To illustrate the effect of the proposed preprocessing filter in terms of bitrate consolidation and accuracy improvement, we take the raw video encoded using H.264 as the benchmark. In addition, we consider an independent FG/BG segmentation algorithm initially introduced in [31] and improved in [32]. Despite its popularity as a FG/BG segmentation approach, this algorithm has not been used for video filtering applications similar to what is considered herein. We adopt the principles from the original algorithm and explain how to adapt it for video filtering in the following paragraphs. This adapted segmentation approach is used as a reference for our proposed preprocessing module, and we will discuss the relative merits of the two designs.

Briefly explained, in this reference approach, the temporal pixel intensity variation is considered as a "pixel process" [31], where at each particular time instance, the pixel intensity value is a random variable drawn from a mixture of Gaussians (MoG) distribution. The Gaussian components are ranked by their pertinence (inversely related to variance) and prior probabilities. The pertinent Gaussian components with high prior probabilities are considered to be associated with the background. For pixels in a new frame, they are matched to one of the Gaussian components based on the normalized distance between the pixel value and the component mean. A pixel is classified as background if the Gaussian component it matches to is considered to be associated with the background, and it is classified as foreground otherwise. After the matching process, the parameters (means and variances) and the prior probabilities of the Gaussian components are updated, incorporating the information carried by the new pixel. For details about the MoG FG/BG segmentation algorithm, the reader is referred to [31] and [32].

The MoG-based FG/BG classification is performed for all pixels in the new frame. After the FG/BG separation process is completed, a mask $\mathbf{M}_t$ is generated such that

$$M_t(n_1,n_2) = \begin{cases} 1, & \text{if } F_t(n_1,n_2) \in \text{foreground} \\ 0, & \text{otherwise} \end{cases}, \tag{10}$$

where $M_t(n_1,n_2)$ is the value of the mask at location (n_1,n_2) and time instance t. Adopting notations similar to what are used in the discussion of TDT, we have

$$\tilde{\mathbf{F}}_t = \mathbf{M}_t \star \mathbf{F}_t + (1-\mathbf{M}_t) \star \mathbf{F}_{\text{static}}, \tag{11}$$

where $\mathbf{F}_{\text{static}}$ is a static image taken from the video to represent the background. In a typical background subtraction application the background frame is usually set to black but for visual purposes a static frame from the video is used to obtain the processed video. Although any frame taken from the video can be used as the static image $\mathbf{F}_{\text{static}}$, it is necessary that this frame is not updated because otherwise it will confuse the tracker .

Although TDT can be considered a simplification of MoG using a single Gaussian component, the different classification decisions between the two imply different treatment as to what is considered to be background. For TDT algorithm, the background regions are replaced by the collocated image signal from the previous filtered frame; thus a pixel in background is considered unchanged between frames t and $t-1$. For the MoG algorithm, the FG/BG decisions are made by comparing the input pixel values with the statistical models and therefore the background pixels may not exhibit resemblance to their predecessors in the previous frame. This implies that TDT will have a more visually appealing background that is constantly updated and adaptable to changes. However, this adaptability leads to bitrate costs since as expected fewer macroblocks are skipped. MoG on the other hand uses a static background that may be visually less appealing but results in more skipped macroblocks (thus bitrate savings). Letting MoG mimic TDT in BG treatment and use the previous filtered output to replace the background will result in trailing artifacts due to how MoG treats the background.

An illustrative example will demonstrate the different treatment of background between TDT and MoG. For explanatory purposes we assume the MoG has two components, with one representing the FG (high intensity) and the other representing the BG (low intensity). The MoG algorithm classifies a pixel based on its normalized distance to the means of these two components instead of based on its difference between its predecessor. Consider a pixel on the inner boundary of a moving vehicle (FG object) in frame $t-1$. In this frame $(t-1)$, this pixel is classified as FG and hence the pixel in the filtered frame $\tilde{\mathbf{F}}_{t-1}$ has high intensity value. In frame t due to the motion of the vehicle, this pixel is no longer located within the vehicle. This is where the difference between TDT and MoG starts to appear. For TDT, because this pixel experiences a significant change in intensity value (from high value in the foreground to low value in the background), it is classified as FG and hence the output at time t will be the same as the input, which is low intensity value. However, in MoG the pixel will be classified as BG because of its low intensity value matches the BG Gaussian component. Since MoG replaces this BG pixel by the colocated pixel in the static image, the output pixel assumes a low intensity value. However, if we let MoG mimic TDT in BG treatment and use the previous filtered output to replace the background, this pixel (classified as BG) would assume the value from the filtered frame at time $t-1$, which was FG and had high intensity value. Therefore as the video proceeds, there would be severe trailing artifacts following the moving vehicle if the MoG algorithm had used a background treatment similar to that employed in TDT. In summary, for TDT, whatever decided to be BG should exhibit little difference from its predecessor, and hence using its predecessor to replace it makes sense. For MoG, a pixel classified as BG does not necessarily mean it is similar to its predecessor, and therefore we may not directly use its predecessor to replace it.

An additional benefit of TDT is its lower computational complexity. The MoG algorithm maintains a multi-component model for each pixel while the TDT algorithm maintains a single-component model for all pixels within a frame. Hence TDT requires less computation to update its statistical model and uses less space to store the model than MoG. This difference is important in light of the application considered in this work, where the remote imaging sensors are typically computational power constrained and memory limited.

We will see shortly that TDT, despite its relative simplicity, has satisfactory performance in terms of bitrate consolidation and accuracy improvement compared to the MoG algorithm, which is used as a reference.

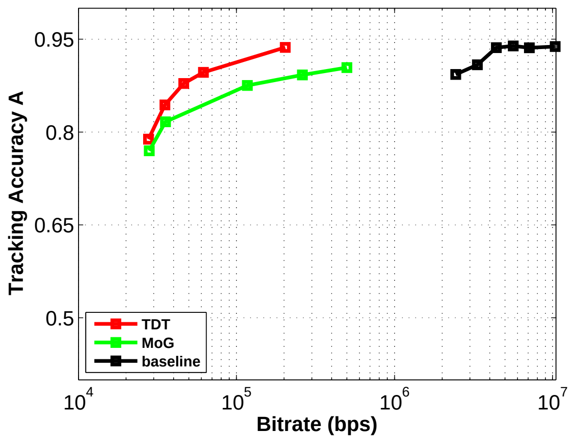
For the remainder of our experiments for TDT filtering, we set the threshold τ to be 2 (assuming 95% intensity variations are due to noise or illumination changes) and buffer size T to be 7; for MoG filtering 5 Gaussian components are used. Unless otherwise noted, in the experiments we use the first frame in a video sequence as the static image for MoG. Note that in a practical deployment the statistical models can be updated in an auto-regressive fashion instead of maintaining a full buffer of image frames.

In Figures 5 and 6, we present sample outputs from the preprocessing steps from typical frames that exhibit moving objects of tracking interests (e.g. vehicles and pedestrians) as well as noise and irrelevant motion. Figure 5(a) shows the 422^{th} frame in the "Camera6" sequence, which includes a group of people walking on the sidewalk and streams of vehicles passing through the scene. Figure 5(b) and 5(d) show the masks generated by the MoG and TDT filters, respectively. As can be seen from the figures, both algorithms generate relatively accurate foreground masks, uncovering the moving objects of tracking interest. TDT filtering misclassifies some solitary pixels in the background as foreground, leading to false positives. Although these may not hurt bitrate performance noticeably (because of the small proportion of such pixels per frame) as we will see next, they are evidence as to the difficulty of fusing multiple masks to create a homogeneous region of interest for UEP. To the contrary, Figure 5(c) shows the ROI map extracted by the kurtosis-based algorithm. After analysis of the video statistics (excess kurtosis), the ROI extraction identifies the regions in the video where motion of tracking interest is likely to occur. The generated ROI map clearly highlights the streets and sidewalk and is thus well suited for UEP. However, if it were to be used for compression some bits would be spent on encoding static information (e.g., road surface). Thus our proposed design takes advantage of the complementary behavior of the two algorithms achieving higher accuracy with less computational cost.

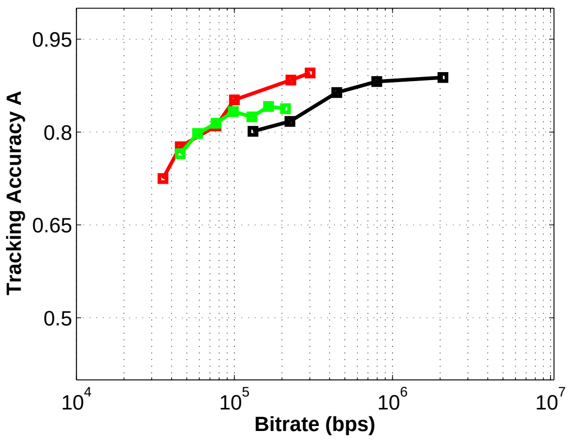
We examine quantitatively the effectiveness of the TDT and MoG algorithms in bitrate consolidation and compare their performance with unpreprocessed video encoded using H.264 (the baseline). All videos are encoded and decoded using the standard JM codecs and tracking accuracies are recorded. It is evident from Figure 4 that the proposed TDT filtering effectively consolidates bitrate on the video content most relevant to tracking. Compared to the baseline, TDT significantly reduces the encoded bitrates while maintaining satisfactory tracking accuracy. Compared to its MoG counterpart, TDT demonstrates comparable or even improved performance despite its computational simplicity.

C. Effects of Error Concealment

As alluded in Section IV-D error concealment algorithms vary in terms of their effectiveness in recovering the lost information and maintaining tracking accuracy. To investigate these variations, the unpreprocessed videos are encoded with various quantization parameters (QPs) and for each QP eight random channel realizations are obtained. In order to make a fair comparison, for each realization, the same lossy bitstream is decoded with the ERC strategies being considered, and the final tracking accuracy results of each realization are averaged. Besides, in order to clearly examine the effects of ERC, we employ a popular packetization strategy in which each slice consists of a row of MBs.

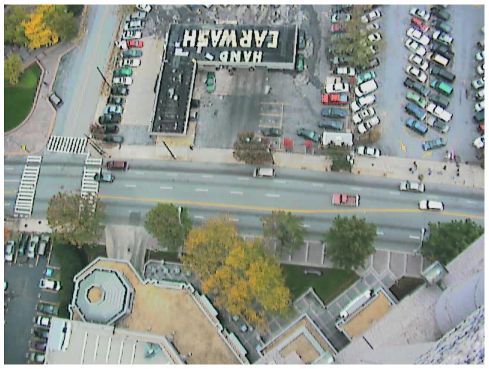


(a) “Camera6” sequence

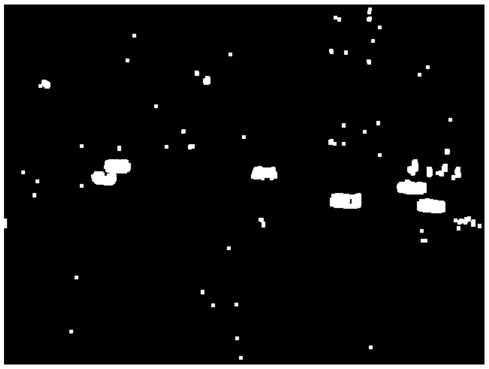


(b) “dt_passat” sequence

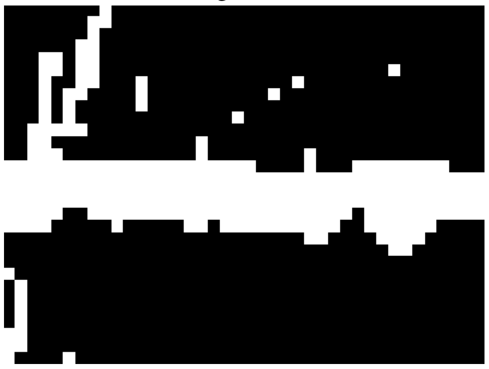
Fig. 4: Comparison between filtering approaches.



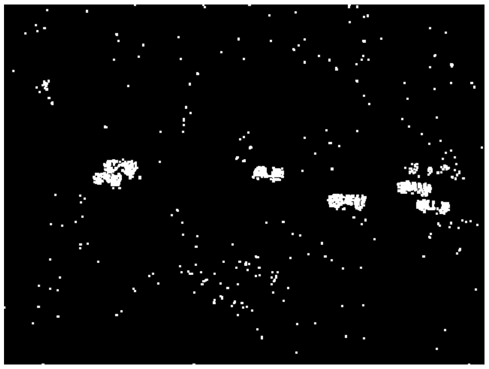
(a) original frame



(b) MoG mask



(c) kurtosis-based ROI



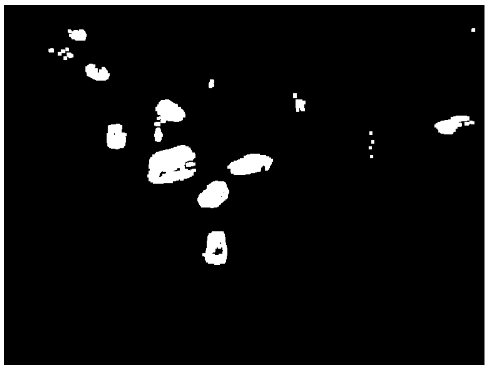
(d) TDT mask

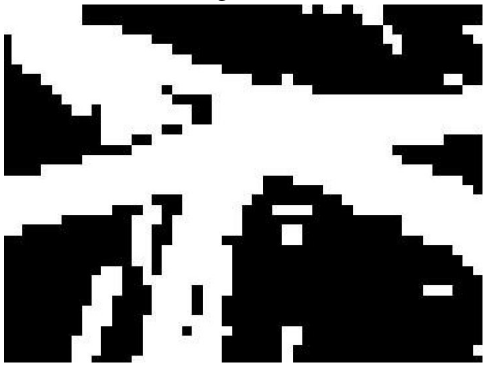
Fig. 5: Preprocessing for “Camera6” sequence

Figures 7 and 8 show zoom-in views of the concealed image regions for the “Camera6” and “dt_passat” sequences, respectively. As discussed in Section IV-D, the “trailing” artifacts seen in the figures are examples of artifacts due to encoder/decoder mismatch that occurs when the lost information is imperfectly estimated at the receiver. It is expected that such artifacts are quite misleading to many trackers, which rely on local image features to determine the position and shape of the objects.

Since the spatial-domain BMA algorithm only considers local image consistency it uses metrics based on the difference between pixel intensities (such as sum-of-squared-differences and sum-of-absolute-differences). Minimizing cost functions

(a) original frame


(b) MoG mask


(c) kurtosis-based ROI

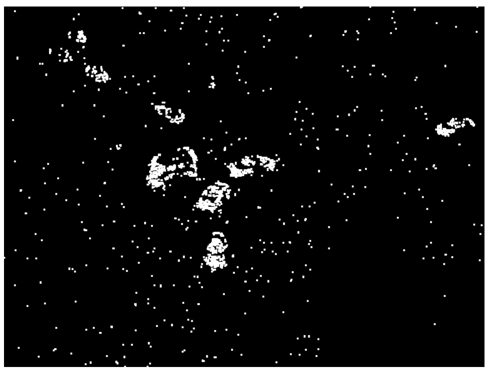
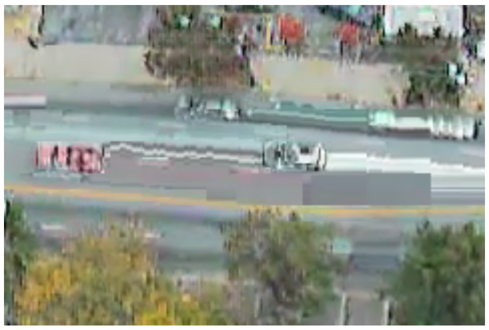

(d) TDT mask

Fig. 6: Preprocessing for "dt_passat" sequence


(a) BMA concealed frame

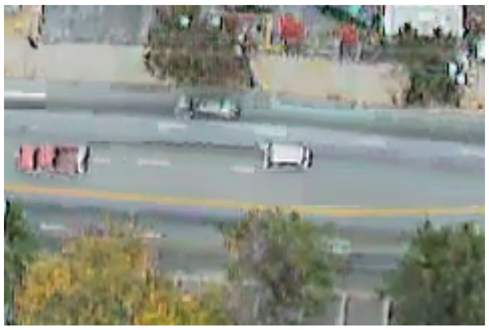
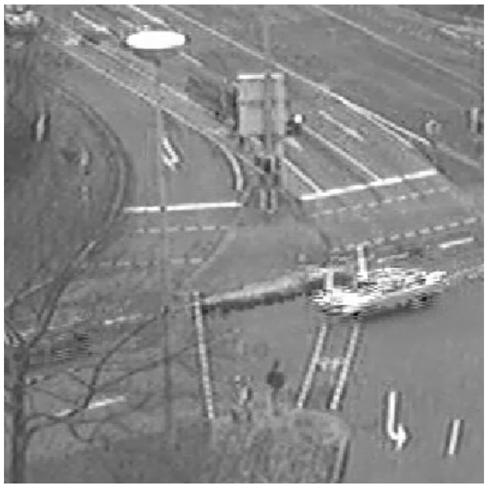

(b) MC concealed frame

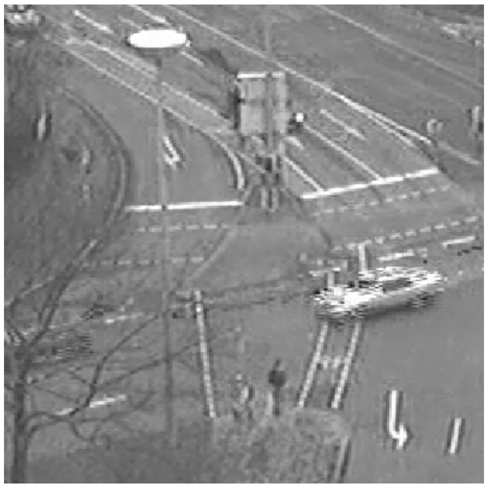
Fig. 7: Sample concealed frame for "Camera6" sequence

based on such metrics does not necessarily lead to optimal concealment for the purpose of tracking. On the other hand, the temporal-domain error concealment strategy MC, takes into account the importance of consistent motion. It makes better utilization of the motion information embedded in the previously decoded frames and gives results with less severe "trailing" artifacts. This visual observation is confirmed also from our quantitative analysis.

In Figure 9 we plot the tracking accuracies for the concealed video sequences at various bitrates. Average packet loss probability is maintained at 0.05 for both sequences. As can be seen from the figures, videos concealed with MC in general preserve more useful information for tracking and therefore yield higher tracking accuracies. The comparison of tracking accuracy and its components (Section III) between videos concealed with MC and BMA is given in Table I. The average improvements in tracking performance (overlap, precision, sensitivity, and overall accuracy) for videos concealed using MC

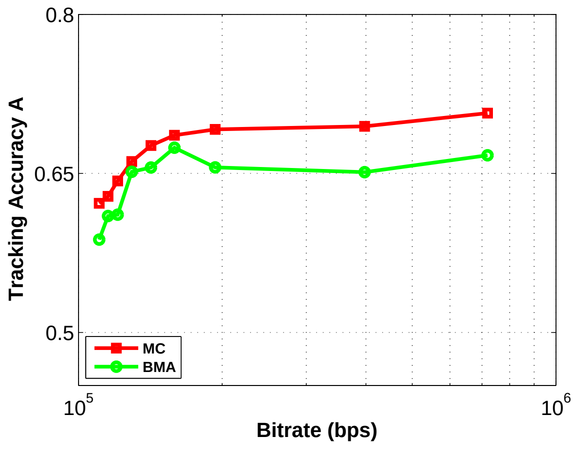


(a) BMA concealed frame

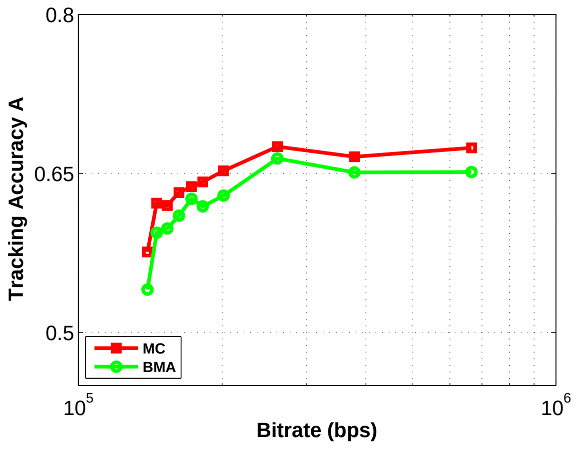


(b) MC concealed frame

Fig. 8: Sample concealed frame for "dt_passat" sequence



(a) the "Camera6" sequence



(b) the "dt_passat" sequence

Fig. 9: Comparison between error concealment for tracking performance

over BMA are given and the standard deviation of such improvements are computed. The experimental results verify our theoretical reasoning from the previous section. In fact when examining the individual components of tracking accuracy, we see that when BMA is used a 12% reduction in precision is observed across all bitrates compared to MC, while for overlap and sensitivity the differences are relatively small. By the definition of precision and sensitivity (Section III) this implies that BMA results in an increase of false positives, which is directly attributed to trailing artifacts due to encoder/decoder mismatches.

We should note that channel impairments (even under moderate channel conditions) do affect tracking accuracy negatively and in particular the precision metric due to an increase in false positives. This explains the overall low tracking accuracies for both sequences. Thus either trackers must be designed that are capable of dealing with such inputs or as we will see in the next paragraph bitrate consolidation and channel protection must be introduced to improve overall system performance.

TABLE I: Tracking Performance Improvement with MC over BMA

Sequence	Overlap		Precision		Sensitivity		Accuracy	
	Mean	Std_dev	Mean	Std_dev	Mean	Std_dev	Mean	Std_dev
Camera6	0.0203	0.0099	0.0581	0.0207	0.0053	0.0170	0.0279	0.0129
dt_passat	0.0194	0.0118	0.0330	0.0133	0.0117	0.0122	0.0214	0.0074

TABLE II: System component settings

System ID	Input Filtering Method	Kurtosis-based ROI Extraction Enabled (Section IV-A2)	FMO Enabled	FEC Type (Section IV-C)	Protection Level	ERC Type
Proposed-1	TDT (Section IV-A1)	No	No	EEP	$C = 3$	MC
Proposed-2	TDT (Section IV-A1)	Yes	Yes	UEP	$H = 3$ $L = 2$	MC
Reference	MoG (Section V-B)	No	No	EEP	$C = 3$	MC
Baseline-1	None	No	No	None	$C = 1$	MC
Baseline-2	None	No	No	EEP	$C = 2$	MC
Baseline-3	None	No	No	EEP	$C = 3$	MC

D. Overall System Performance

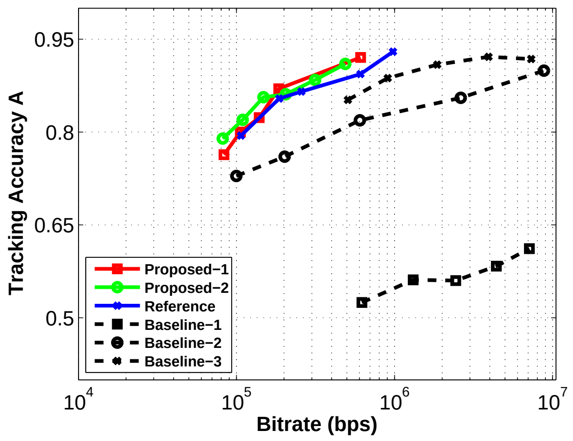
In the previous paragraphs we show that some preprocessing (TDT or MoG) can consolidate bitrate and that different ERC strategies have different effects on tracking accuracy. In this section, we combine the proposed TDT preprocessing, FEC (EEP and UEP) and ERC modules together into a complete system, and compare the performance with a reference system based on MoG as well as variations of baseline H.264 implementations. Depending on the selection of preprocessing configuration and FEC strategy, there exist multiple settings for the proposed system. From all possible combinations we present those settings (Table II) that help illustrate the combined gains of the proposed approach.

We refer to the different system settings by their respective IDs listed in the first column in Table II. Note that both Proposed-1 and Proposed-2 use TDT. However Proposed-1 employs EEP and hence does not require FMO-enabled encoding or explicit ROI extraction. Proposed-2 on the other hand uses UEP for channel protection, which requires FMO-enabled encoding for macroblock to slice group mapping and requires the ROI extraction step. The reference system employs the MoG algorithm for video filtering and EEP as the channel protection option. MoG filtering replaces the background regions with a static image, and thus minimum bits are spent on encoding such regions. Therefore a UEP scheme leads to minimal gains and is not shown for clarity.

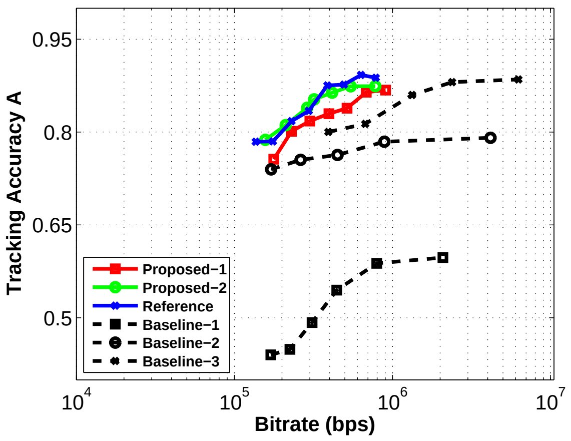
For the baseline implementations we use the H.264 encoder with the unpreprocessed video. Since channel errors have a major impact on the decoded video quality, unprotected bitstreams (Baseline-1) yield exceptionally low tracking accuracy. In order to make the comparison of results more meaningful, we apply EEP with two levels (C=2 and C=3) to the baseline encoded bitstreams and record their respective tracking accuracy after transmission and decoding.

The performance comparisons are shown in Figures 10. The proposed and reference systems are plotted in solid curves while the baseline implementations are plotted in dashed curves.

The figures make it evident that the baseline implementations, despite their great efficiency in generic video compression, are



(a) the “Camera6” sequence



(b) the “dt_passat” sequence

Fig. 10: Comparison of system performance for IID channel

less effective in handling the tracking-specific video compression and transmission. The proposed TDT filtering step removes video content unimportant for tracking from the raw video and saves bitrate for protection purposes. The FEC module utilizes the bitrate savings to add robustness to the encoded bitstream, providing protection to the video information most relevant for tracking. At the decoder side, tracking-aware error concealment strategy is employed to recover the lost motion information and therefore further improves the overall system performance. Quantitatively the proposed system yields approximately 80% reduction in bitrate for the same tracking accuracy compared with the protected baseline implementation.

Comparing the proposed systems with the reference MoG-based system, we can see that they have similar performance despite the simplicity of TDT. TDT being computationally less expensive than MoG, provides a significant advantage in computational power constrained environments such as the remote nodes considered here.

Comparing between Proposed-1 and Proposed-2, we can see that under certain circumstances gains are possible by using UEP. In situations when complex motion types are present UEP has the potential to outperform its EEP counterpart because the ROI map it relies on is derived using higher-order statistics.

From this numerical analysis and comparison, we see that the proposed system is effective in (1) consolidating bitrate onto the most important video content for tracking; (2) protecting the encoded bitstreams against channel degradations; and (3) recovering lost information via tracking optimized error concealment strategies, while always maintaining a low computational cost profile at the encoder side. The proposed system yields significant performance improvement over current state-of-the-art video coding implementations that are not customized for tracking applications. Its performance is also validated with an independent design adopted for the transportation video transmission and surveillance system. While achieving comparable (and in some cases better) performance, our proposed system is computationally more efficient and provides a more feasible solution in a resource-constrained environment.

VI. Conclusions

In this report we presented a video coding and transmission system specifically tailored to automated centralized vehicle surveillance and monitoring. The characteristics of transportation video and the lossy nature of the wireless channels were considered when designing the system. Video signal components were analyzed and information less important to tracking was removed effectively by preprocessing steps. To mitigate the negative effects of channel losses to automated tracking of vehicles, we combined forward error correction at the transmitter and an error concealment module at the receiver. The effectiveness of the proposed system was tested using real-life video sequences and the performance improvement over the current state-of-the-art system in maximizing tracking accuracy shown.

The following papers have resulted from the project:

1) Z. Chen, E. Soyak, S. A. Tsaftaris, and A. K. Katsaggelos, "Tracking-optimal error control schemes for H.264 compressed video for vehicle surveillance", in 20th European Signal Processing Conference (EUSIPCO), 2012, pp. 1900-1904
2) Z. Chen, E. Soyak, S. A. Tsaftaris, and A. K. Katsaggelos, "Application-aware approach to compression and transmission of H.264 encoded video for automated and centralized transportation surveillance", to appear in IEEE Transactions on Intelligent Transportation Systems, 2013

Acknowledgment

This work is funded in part by the Commercialization of Innovative Transportation Technology (CCITT), Northwestern University.

References

[1] M. Kyte, A. Khan, and K. Kagolanu, "Using machine vision (video imaging) technology to collect transportation data," *Transportation Research Record, Innovations in Travel Survey Methods*, no. 1412, pp. 23–32, 1993.

[2] N. Anjum and A. Cavallaro, "Multifeature object trajectory clustering for video analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, no. 11, pp. 1555–1564, 2008.

[3] F. Jiang, Y. Wu, and A. Katsaggelos, "A dynamic hierarchical clustering method for trajectory-based unusual video event detection," *IEEE Transactions on Image Processing*, vol. 18, no. 4, pp. 907–913, 2009.

[4] F. Jiang, J. Yuan, S. Tsaftaris, and A. Katsaggelos, "Anomalous video event detection using spatiotemporal context," *Computer Vision and Image Understanding, Special Issue on Feature-Oriented Image and Video Computing for Extracting Contexts and Semantics*, vol. 115, no. 3, pp. 323–333, 2010.

[5] (2011, June) H.264 : Advanced video coding for generic audiovisual services. [Online]. Available: http://www.itu.int/rec/T-REC-H.264/

[6] A. Kannur, "Power-aware content-adaptive h.264 video encoding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009.

[7] C. Erdem, "Video object tracking with feedback of performance measures," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 4, pp. 310–324, 2003.

[8] W. Ho, "Content-based scalable H.263 video coding for road traffic monitoring," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 615–623, 2005.

[9] R. Sutter, K. DeWolf, S. Lerouge, and R. de Walle, "Lightweight object tracking in compressed video streams demonstrated in region-of-interest coding," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, p. 59, 2007.

[10] E. Soyak, S. Tsaftaris, and A. Katsaggelos, "Low-complexity video compression for automated transportation surveillance," *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Video Analysis on Resource-Limited Systems*, vol. 21, no. 10, pp. 1378–1389, 2011.

[11] A. Katsaggelos, Y. Eisenberg, F. Zhai, R. Berry, and T. Pappas, "Advances in efficient resource allocation for packet-based real-time video transmission," *IEEE Proceedings*, vol. 93, pp. 135–147, January 2005.

[12] P. Baccichet, S. Rane, A. Chimienti, and B. Girod, "Robust low-delay video transmission using H.264/AVC redundant slices and flexible macroblock ordering," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 4, July 2007, pp. 93–96.

[13] E. Soyak, S. Tsaftaris, and A. Katsaggelos, “Channel protection for H.264 compression in transportation surveillance applications,” in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, 2011, pp. 2285–2288.

[14] Z. Chen, E. Soyak, S. Tsaftaris, and A. Katsaggelos, “Tracking-optimal error control schemes for H.264 compressed video for vehicle surveillance,” in *20th European Signal Processing Conference (EUSIPCO)*, 2012.

[15] ——, “Application-aware approach to compression and transmission of H.264 encoded video for automated and centralized transportation surveillance,” *IEEE Transactions on Intelligent Transportation Systems (to appear)*, 2013.

[16] Y. Wang and Q. Zhu, “Error control and concealment for video communication: A review,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 86, no. 5, pp. 974–997, 1998.

[17] A. Baumann, M. Boltz, J. Ebling, M. Koenig, H. Loos, M. Merkel, W. Niem, J. Warzelhan, and J. Yu, “A review and comparison of measures for automatic video surveillance systems,” *EURASIP Journal on Image and Video Processing*, vol. 2008, 2008.

[18] E. Soyak, S. Tsaftaris, and A. Katsaggelos, “Content-aware H.264 encoding for traffic video tracking applications,” in *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, March 2010, pp. 730–733.

[19] F. Zhai, Y. Eisenberg, T. Pappas, R. Berry, and A. Katsaggelos, “Joint source-channel coding and power adaptation for energy efficient wireless video communications,” *Signal Processing: Image Communications*, vol. 20, no. 4, pp. 371–387, March 2005.

[20] P. Pahalawatta, R. Berry, T. Pappas, and A. Katsaggelos, “Content-aware resource allocation and packet scheduling for video transmission over wireless networks,” *IEEE Journal on Selected Areas in Communications, Special Issue on Cross-Layer Optimization for Wireless Multimedia Communications*, vol. 25, pp. 749–759, 2007.

[21] Y. Wang, S. Wenger, J. Wen, and A. Katsaggelos, “Error resilient video coding techniques,” *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, July 2000.

[22] W. Lam, A. Reibman, and B. Liu, “Recovery of lost or erroneously received motion vectors,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, April 1993, pp. 418–420.

[23] S. Bandyopadhyay, Z. Wu, P. Pandit, and J. Boyce, “An error concealment scheme for entire frame losses for H.264/AVC,” in *IEEE Sarnoff Symposium*, March 2006, pp. 1–4.

[24] (2012, Feburary) Next Generation Simulation (NGSIM) community. [Online]. Available: http://www.ngsim-community.org/

[25] (2012, Feburary) KOGS/IAKS Karlsruhe image sequence server. [Online]. Available: http://i21www.ira.uka.de/imagesequences/

[26] (2012, Feburary) The open-source H.264/AVC verification model. [Online]. Available: http://iphome.hhi.de/suehring/tml/

[27] (2010, December) The OpenCV real-time computer vision library. [Online]. Available: http://opencv.willowgarage.com

[28] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Hilton Head, SC, USA, 2000, pp. 142–149.

[29] P. Shankar, *Introduction to Wireless Systems*. Wiley, 2001.

[30] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2002.

[31] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR)*, 1999.

[32] P. KaewTraKulPong and R. Bowden, “An improved adaptive background mixture model for real-time tracking with shadow detection,” in *Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems (AVBS)*, 2001.