Low-Complexity Tracking-Aware H.264 Video Compression for Transportation Surveillance

Eren Soyak, *Student Member, IEEE,* Sotirios A. Tsaftaris, *Member, IEEE,* and Aggelos K. Katsaggelos, *Fellow, IEEE*

***Abstract*—In centralized transportation surveillance systems, video is captured and compressed at low processing power remote nodes and transmitted to a central location for processing. Such compression can reduce the accuracy of centrally run automated object tracking algorithms. In typical systems, the majority of communications bandwidth is spent on encoding temporal pixel variations such as acquisition noise or local changes to lighting. We propose a tracking-aware, H.264-compliant compression algorithm that removes temporal components of low tracking interest and optimizes the quantization of frequency coefficients, particularly those that most influence trackers, significantly reducing bitrate while maintaining comparable tracking accuracy. We utilize tracking accuracy as our compression criterion in lieu of mean squared error metrics. Our proposed system is designed with low processing power and memory requirements in mind, and as such can be deployed on remote nodes. Using H.264/AVC video coding and a commonly used state-of-the-art tracker we show that our algorithm allows for over 90% bitrate savings while maintaining comparable tracking accuracy.**

***Index Terms*—Quantization, urban transportation video, video compression, video object tracking, video processing.**

I. Introduction

VIDEO imaging sensors are commonly used in transportation monitoring and surveillance. Such sensors are a cost effective solution that yields information on a large field of view, allowing for real time monitoring of video feeds and video archiving for forensic, surveillance and traffic analysis applications. Other vehicular monitoring solutions, such as embedded inductor cables or radars, can only identify and count vehicles and measure instantaneous speed without providing any further information. Video imaging is the only existing modality that observes a vehicle's complete trajectory, opening the door to a completely different set of applications [1]. Possible applications include the remote surveillance of transportation hubs, automatic detection of anomalies, and study of transportation phenomena such as driver behavior and its possible effects on safety and congestion [2]–[4].

In order to limit infrastructure costs associated with their deployment, most video imaging solutions require the transmission of the 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; such nodes require no hardware other than cameras, low-cost consumer-grade embedded systems and a network connection.

This centralized approach mandates the compression of video before it is transmitted. Transmitting high-quality video requires expensive wired communication links, while relying on lower cost wireless links requires heavy compression and results in reduced video quality. Most video compression systems use block-based motion compensation [5], where temporal redundancy is eliminated via the use of block motion vectors and frequency-transformed residuals. For typical transportation surveillance the camera is stationary, and the majority of bitrate is spent representing temporal changes to the scene due primarily to acquisition noise or small changes to lighting.

There exist several specialized encoders addressing surveillance applications [6]–[10]. However, there has been an increasing interest in identifying video compression solutions that can further reduce the required bitrate without violating standard compliance or increasing encoder complexity.

Within the scope of remaining standard-compliant, reducing bitrate and increasing video quality, a number of approaches have been suggested to reduce noise as much possible [11] or to take into account the fact that the camera is stationary. Alternatively, methods such as suggested in [12] focus on consolidating processing power on critical visual elements by their motion type. In [13], an approach was proposed that adds higher level elements such as motion field correction filtering in the context of H.263. In [14], a method of using automatic resizing of ROIs detected by video encoder motion estimation in conjunction with object tracking is presented; for this algorithm an effective ROI estimate requires encoder motion estimation capturing true motion. In [15], a method of using ROIs to focus limited processing power on highest gain encoder components in the context of H.264/AVC is presented. These methods are all low in complexity, but rely on information generated by the encoder (such as motion vectors

Manuscript received December 13, 2010; revised May 23, 2011 and July 4, 2011; accepted July 7, 2011. Date of publication August 4, 2011; date of current version October 5, 2011. This work was supported in part by the Center for the Commercialization of Innovative Transportation Technology. This paper was recommended by Associate Editor H. Gharavi.

E. Soyak is with AirTies Wireless Networks, Istanbul 34394, Turkey (e-mail: e-soyak@northwestern.edu).

S. A. Tsaftaris is with the Area of Computer Science and Applications, IMT Institutions Markets Technologies Institute of Advanced Studies, Lucca, LU 55100, Italy (e-mail: sotirios.tsaftaris@imtlucca.it).

A. K. Katsaggelos is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: aggk@eecs.northwestern.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2011.2163448

or macroblock types) to limit computation. Results for such algorithms rely on assumed encoder behavior, and for example will suffer in cases where a "best block match" motion field, which reduces block residual energy without actually reflecting true motion, is used.

The above algorithms do not perform application-specific compression, i.e., they do not take into account the possible data analysis that will be applied to the compressed video. As a result of such compression the tracking accuracy and efficiency can be severely affected, necessitating the development of new methodologies taking compression distortion into account (one example of such an approach is presented in [16]).

The design of a standard-compliant low-complexity encoder tailored to tracking applications, and to this end using tracking accuracy rather than peak signal-to-noise ratio (PSNR) as a compression performance metric, had not been considered until [17]. Therein, it was proposed that subjective tracking accuracy is better suited to judge system performance and (assuming a static background) an automated low-complexity ROI detection algorithm for bitrate consolidation was offered. Indeed, as Fig. 1 illustrates, one type of compression distortion may be significantly more misleading for a tracker compared to a second one. Note that with respect to Fig. 1(a), both the smoothing distortion shown in Fig. 1(b) and the blocking artifacts shown in Fig. 1(c) have the same PSNR.

In this paper, we present a combined set of algorithms to allow compression resources to be focused on video elements of tracking interest. The algorithms presented herein are designed to be low in complexity and to be readily deployable as a simple modular add-on to low processing power remote nodes of centralized transportation video systems. They make no assumptions about the operation of the video encoder (such as its motion estimation or rate control methods) and are thus suitable for use in a variety of systems. The resulting bitstreams are standard-compliant, thereby guaranteeing interoperability with other systems. Early versions of some of the algorithms proposed herein are covered in [18] and [19].

The rest of this paper is organized as follows. In Section II, we outline our proposed system. In Section III, we present our framework for measuring automated tracker efficiency. In Section IV, we propose a method for bitrate concentration on critical temporal changes to video. In Section V, we propose a method of quantization table optimization tailored to tracking applications. In Section VI, we discuss details pertaining to the joint implementation of the two algorithms, and in Section VII we show experimental results. Finally, in Section VIII we present concluding remarks.

II. Proposed System

We propose a system using application-specific video compression to minimize the bandwidth requirement for links connecting central and remote nodes. This is done by minimizing bits spent coding components of low tracking interest, specifically: 1) temporal pixel variations such as local changes to illumination which are not useful to trackers in general, and 2) frequency components that are less valuable to the specific tracker being used. While 1) is achieved in real-time

Fig. 1. Example of distortions with same PSNR. (a) Original. (b) Smoothing distortion. (c) Blocking artifacts. With respect to (a), both (b) and (c) have a PSNR of 34.7 dB.

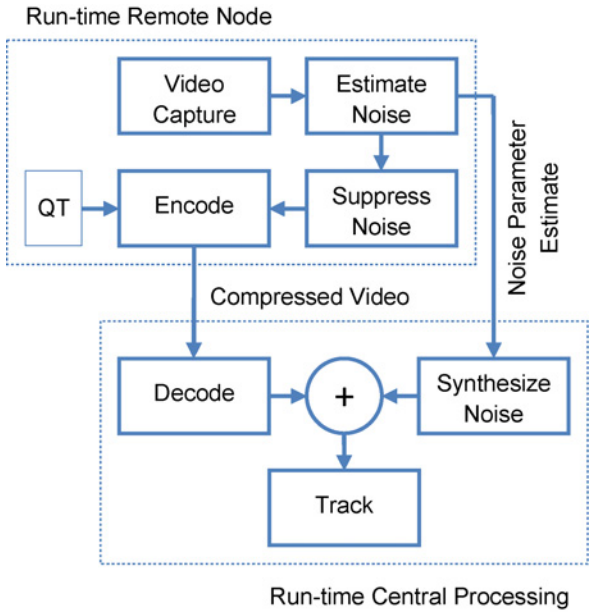

Fig. 2. Proposed runtime system.

by estimating pixel-level statistics in input video and filtering out small pixel-level fluctuations (details presented in Section IV), 2) is achieved by optimizing a quantization table specific to the automated tracker used (details presented in Section V). This allows us to affect implicitly rate-distortion level decisions without requiring the presence of a tracker at the encoder.

Fig. 2 shows the proposed runtime system. At the remote (camera) node, video components considered to be noise are estimated and suppressed prior to compression, where an optimized quantization table (QT) may be used to further focus bit allocation on elements of tracking interest. The compressed video and estimated noise parameters are transmitted to the central node, where the video is decoded and noise synthesized as per the remote node estimated parameters is inserted prior to tracking. This system can be deployed in a multitude of scenarios (e.g., using low-cost wireless/cellular links), that vary according to the available infrastructure. Section VII-C discusses these scenarios in detail.

Hardware costs associated with processing requirements on numerous remote nodes are minimized by keeping algorithmic complexity low enough to run real-time on low-cost consumer-grade processors. We assume that some processing power exists at a central location equipped with automated trackers. This processing power can be either utilized in an initialization

phase or not at all. As such multiple combinations of the two algorithms 1) and 2) exist, which are detailed in Section VI.

III. Measuring Automated Tracking Efficiency

The field of video object tracking is quite active, with various solutions offering strength/weakness combinations suitable for different applications. For urban transportation video tracking, most applications involve a background subtraction component for target acquisition such as the one developed in [20], and an inter-frame object association component such as those developed in [21] and [22]. Most such tracking algorithms account only for the native statistics of video objects, and as a result distortion of these statistics by sources such as compression may severely degrade their accuracy.

In order to optimize tracking quality a metric to measure tracking accuracy is required. In [23], a review of the state-of-the-art for video surveillance performance metrics is presented. While more complex metrics such as the ones presented in [24] may be used, due to their pertinence to transportation surveillance, we choose the overlap, precision, and sensitivity metrics presented in [23], with the ground truth defined as tracking results generated using uncompressed video.

Overlap ($OLAP$) is defined in terms of the ratio of the intersection and union of frame pixels covered by ground truth (GT) and algorithm result (AR) object segmentations

$$OLAP = 1/N \sum_{i=1}^{N} (GT_i \cap AR_i)/(GT_i \cup AR_i) \tag{1}$$

where GT_i represents the ith object tracked in uncompressed video, AR_i the ith object tracked in compressed video, $\cap$ the intersection of the two regions, and $\cup$ their union. Note here that $OLAP$ refers to an average over all N objects in a given video sequence. *Precision* ($PREC$) is defined in terms of the number of true positives (TPs) and false positives (FPs) in the video sequence as

$$PREC = TP/(TP + FP). \tag{2}$$

A true positive results from an object being present in both the GT and AR (i.e., the tracked regions in the GT and AR overlap). A false positive results from an object being present in the AR but not in the GT, or if an object detected in the AR does not overlap an equivalent object in the GT. *Sensitivity* ($SENS$) is defined in terms of the number of TPs and false negatives (FNs) in the video sequence as follows:

$$SENS = TP/(TP + FN). \tag{3}$$

A false negative results from an object being present in the GT but not in the AR, or if an object detected in the GT does not overlap an equivalent object in the AR.

In order to jointly optimize for a combination of the above metrics we define the aggregate tracking accuracy A as

$$A = (\alpha * OLAP) + (\beta * PREC) + (\gamma * SENS) \tag{4}$$

where α, β, and γ are weighting coefficients. Given that $OLAP$, $SENS$, and $PREC$ are all in the range [0, 1], no normalization of A is necessary as long as $\alpha + \beta + \gamma = 1$.

IV. Tracking Aware Video Processing

At reasonable compression ratios frame-to-frame pixel intensity variations, e.g., due to acquisition noise or local changes in illumination, are usually imperfectly represented in compressed video due to lossy coding. For example, such variations may be sampled sparsely over time rather than at every frame, leading to seemingly random block updates in the decoded video. Such changes may be interpreted as significant motion by an automated tracking system and thus be highly misleading. The proposed algorithm seeks to suppress such changes, thereby reducing both the required bitrate and post-compression tracking inaccuracies.

Our algorithm, termed temporal deviation thresholding (TDT), operates in two distinct parts: 1) we model, detect, and remove temporal pixel variations of low tracking interest as a preprocess to compression, and 2) after decoding the video at the receiver we use the estimated noise parameters from part 1) to synthesize and re-insert noise prior to tracking. Part 1), referred to as TDT$^-$, aims at minimizing the bitrate requirement, while part 2), referred to as TDT$^+$, aims at improving post-compression tracking results. Details for the algorithm are presented below.

For the remainder of this paper, scalars are defined italicized, e.g., C, while vectors and matrices (2-D, 3-D) are denoted by bold characters, e.g., $\mathbf{M}_t$. We define temporal variations of pixel intensity of low tracking interest as *noise*. This noise is modeled for each frame $\mathbf{V}_t$ as a Gaussian process with zero mean and standard deviation σ_t. We assume that for transportation surveillance video obtained from a static camera, the majority of pixels undergo temporal variation due to noise. Thus, the majority of the frame is comprised of a static background, whose variation in the video will be predominantly due to effects such as acquisition noise (assumed to be Gaussian) and illumination changes rather than to actual motion.

We denote by $\mathbf{D}_t$ the per-pixel standard deviation at time t over the past B buffered frames. We expect the majority of the non-zero values of $\mathbf{D}_t$ to be due to noise. Therefore similarly to [25], we estimate the noise standard deviation σ_t by finding the mode of the distribution of the $\mathbf{D}_t$ values. The goal is to generate a mask $\mathbf{M}_t$, a bitmap of pixels in frame $\mathbf{V}_t$ whose variation is of tracking interest. The filtered frame $\hat{\mathbf{V}}_t$ is iteratively computed as follows:

$$\mathbf{D}_t = std([\mathbf{V}_{t-B}, ..., \mathbf{V}_t]) \tag{5}$$

$$\sigma_t = mode(\, vec(\mathbf{D}_t)\,) \tag{6}$$

$$\Delta_t = |\mathbf{V}_t - \mathbf{V}_{t-1}| \tag{7}$$

$$\mathbf{M}_t = \Delta_t > C * \sigma_t \tag{8}$$

$$\hat{\mathbf{V}}_t = \mathbf{M}_t * \mathbf{V}_t + \overline{\mathbf{M}_t} * \hat{\mathbf{V}}_{t-1} \quad \forall\ t \geq B\ . \tag{9}$$

In (5), *std* denotes the per-pixel standard deviation operator. In (6), *mode* and *vec* denote, respectively, the histogram mode and matrix vectorization operations. In (7), || denotes the per

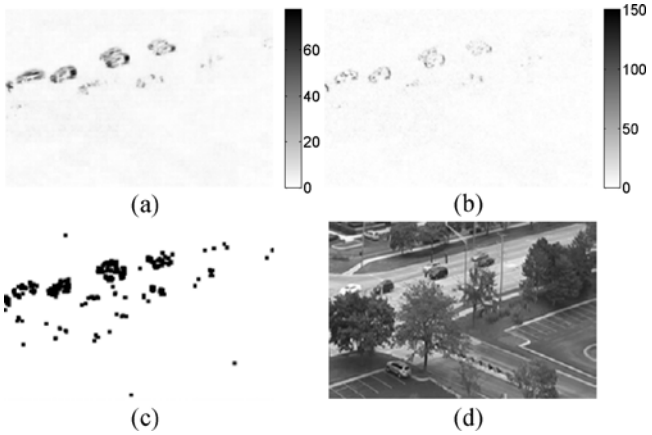

Fig. 3. Sample estimation process for σ_t at t = 18 of the *Golf* sequence. Here shown are (a) $\mathbf{D}_t$, (b) Δ_t, (c) $\mathbf{M}_t$, and (d) $\hat{\mathbf{V}}_t$.

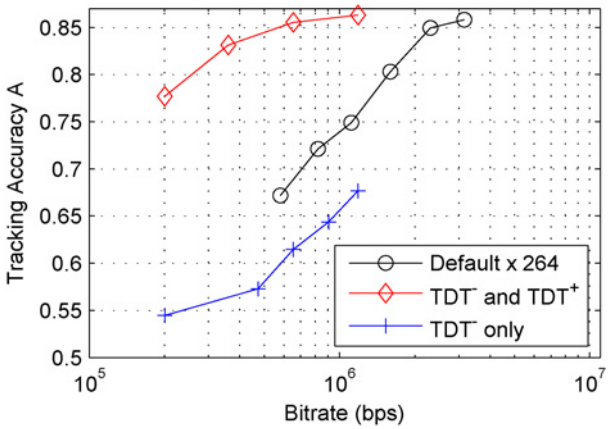

Fig. 4. Performance gain achieved by adding noise at the receiver (TDT$^+$) before tracking versus using TDT$^-$ video for tracking.

matrix entry absolute value operation. In (8), $\mathbf{M}_t$ is a logical bitmap and $\overline{\mathbf{M}_t}$ its logical inverse. Equation (9) describes the iterative operation, where only pixels in $\mathbf{V}_t$ corresponding to values of 1 in the bitmap $\mathbf{M}_t$ are updated in $\hat{\mathbf{V}}_t$.

The process is initialized by setting $\hat{\mathbf{V}}_t = \mathbf{V}_t, \forall\ t < B$, i.e., no frames are processed until B frames have been buffered. Refer to Fig. 3 for a sample iteration of the process described in (5)–(9). Note that in the figure the video encoder receives only the filtered input $\hat{\mathbf{V}}_t$.

The derivation of the coefficient C is done based on the desired confidence interval CI. Since we model noise as Gaussian, its distribution is subject to the cumulative Gaussian distribution function Φ. Refer to [26] for further discussion on the derivation and use of Φ in this context. Using Φ, we express the probability of any given pixel variation being due to noise as the probability that it belongs to a zero-mean Gaussian distribution with standard deviation σ_t

$$CI = Prob(|\Delta_t| < C * \sigma_t) = \Phi(C) - \Phi(-C). \qquad (10)$$

As described above, for TDT$^+$ synthetic noise is reinserted into the decompressed video as part of our algorithm. For each frame $\mathbf{V}_t$, the noise parameter σ_t estimated at the remote node is transmitted along with the compressed video frame. At the receiver randomly generated Gaussian noise with zero mean and standard deviation σ_t is added to the frame, clipping any resulting overflows due to 8-bit pixel precision. The goal of this process is to restore the Gaussian temporal noise characteristics the video possessed prior to compression. Given that many tracking algorithms rely on noise modeling for background subtraction, this step is critical to allow such trackers to be able to distinguish actual foreground (such as vehicles) from pixel variations introduced by compression. In addition, without adding noise (TDT$^+$), any noise that was not suppressed during TDT$^-$ is very likely to be misinterpreted as true motion by a tracker. Observe in Fig. 4 that due to these reasons, the tracking performance of using only TDT$^-$ is relatively low for all bitrates, while using full TDT (both TDT$-$ and TDT$^+$) yields very significant gain.

For video with multiple color components, TDT is applied independently to each component. While joint application to multiple components may improve the robustness of the algorithm, special care should be taken in this case for low-contrast scenes (such as in low-light or fog) presenting a disadvantageous tradeoff between sensitivity and precision.

TDT adds no latency to the system, and the resulting bitstream is fully standard-compliant. Note that the core algorithm (7)–(9) requires only an estimate of the noise standard deviation, which we estimate using (5) and (6), although other possible spatial estimations of noise variance are possible. The parameters for the algorithm are the buffer size B, which should be set as high as possible given memory and processing constraints, and the confidence interval CI, which should be set based on specific application requirements in terms of tracking accuracy.

V. Iterative Quantization Table Search

In this section, we propose an iterative greedy search algorithm which automatically identifies and concentrates bit allocation to frequencies useful to tracking. During each iteration, the encoder quantization scheme of each individual frequency is modified, and tracking accuracy is measured for a sample clip of the video. From these results, only those frequencies which provide the highest tradeoff of bits for tracking accuracy are kept, and subsequent iterations proceed cumulatively. This algorithm aims to make encoder quantization decisions based on tracking accuracy as opposed to the traditional rate-distortion method.

The quantization scheme is varied by the algorithm via the QT specified as part of the sequence and picture parameter set structures in the H.264/AVC video compression standard [27]. Each entry of the QT partially defines quantization of a coefficient resulting from the 4×4 spatial transform—the goal is to spend the fewest bits on coefficients containing the least useful information pertaining to features utilized by the tracker. This allows us to implicitly affect rate-distortion level decisions without requiring a tracker on the encoder. The end result of this algorithm is a QT lookup table (QT-LUT), which is formed of an array of bitrates and corresponding optimized quantization parameter (QP) and QT pairs for each bitrate. Details for the algorithm are presented below.

The quantization of the jth transform coefficient in terms of the QP q and QT is described as

$$\mathbf{QT} = [\phi_0, \phi_1, \phi_2, ..., \phi_{15}] \quad (11)$$

$$quant_j = q * \phi_j / 16 \quad (12)$$

where each 8-bit ϕ_j corresponds to a rasterized 4×4 H.264/AVC residual transform coefficient (e.g., ϕ_0 operates on position [1, 1] and ϕ_{15} on position [4, 4]). Therefore $2^{8 \cdot 16}$ distinct QTs are possible. We define τ, a simplified binary QT where each ϕ_j can only have values of either 16 or 255, thereby limiting our optimization search space to 2^{16} values. The two options (16 or 255) lead to, respectively, using q directly or quantizing coarsely enough to effectively suppress the coefficient. We define τ as follows:

$$\tau = \sum_{j=0}^{15} 2^j * P_j \quad \text{where } P_j = \begin{cases} 1, \text{ if } \phi_j = 16 \\ 0, \text{ if } \phi_j = 255. \end{cases} \quad (13)$$

For example, $\tau = 2^{16} - 1$ (all bit positions equal to 1) would indicate the use of a "flat" quantization table, whereas $\tau = 1$ (all bit positions but the zeroth equal to 0) would indicate a DC-only QT. We define the mapping operation of compression and tracking f_{enc} as follows:

$$\{R, A\} \overset{f_{enc}}{\leftarrow} \{\mathbf{V}, q, \tau\} \quad (14)$$

where R is the resulting compressed bitrate and A the tracking accuracy for the video sequence $\mathbf{V}$ compressed using q and τ. We define a "data point" as associated to $\{R, A, q, \tau\}$, and denote as $\{\mathbf{R}, \mathbf{A}, \mathbf{q}, \boldsymbol{\tau}\}$ a collection (in vectors) of concatenated data points. We define the function f_{opt} isolating the iteration-optimal data points $\{\mathbf{R}^*, \mathbf{A}^*, \mathbf{q}^*, \boldsymbol{\tau}^*\}$ as follows:

$$\{\mathbf{R}^*, \mathbf{A}^*, \mathbf{q}^*, \boldsymbol{\tau}^*\} = f_{opt}(\mathbf{R}, \mathbf{A}, \mathbf{q}, \boldsymbol{\tau}). \quad (15)$$

f_{opt} operates by starting with the lowest bitrate R data point available, and adding all other data points of monotonically increasing tracking accuracy A. Finally, we define the function f_{branch}, which generates the QT modifications necessary for the search iterations, as

$$\boldsymbol{\tau}_n = f_{\text{branch}}(\boldsymbol{\tau}^*_{n-1}) \quad (16)$$

where for every element τ in the input vector $\boldsymbol{\tau}^*_{n-1}$ (size $L \times 1$) an output vector $\boldsymbol{\tau}_n$, which is formed of the concatenation of all 32 possible bit permutations (i.e., of size $32 \cdot L \times 1$), is generated. We initialize our QT search as

$$\mathbf{q}_o = [QP_1, QP_2, ..., QP_M] \quad (17a)$$

$$\tau_{\text{flat}} = 2^{16} - 1 \quad (17b)$$

$$\boldsymbol{\tau}_o = [\tau_{\text{flat}}, \tau_{\text{flat}}, ..., \tau_{\text{flat}}] \quad (17c)$$

$$\{\mathbf{R}_o, \mathbf{A}_o\} \overset{f_{\text{enc}}}{\leftarrow} \{\mathbf{V}, \mathbf{q}_o, \boldsymbol{\tau}_o\} \quad (17d)$$

$$\{\mathbf{R}^*_o, \mathbf{A}^*_o, \mathbf{q}^*_o, \boldsymbol{\tau}^*_o\} = f_{\text{opt}}(\mathbf{R}_o, \mathbf{A}_o, \mathbf{q}_o, \boldsymbol{\tau}_o). \quad (17e)$$

A range of M QPs, each with a corresponding flat QT τ_{flat} (i.e., where τ_o is of size $M \times 1$), are evaluated. We define search iterations $n > 0$ as follows:

$$\mathbf{q}_n = \mathbf{q}^*_{n-1} \quad (18a)$$

$$\boldsymbol{\tau}_n = f_{\text{branch}}(\boldsymbol{\tau}^*_{n-1}) \quad (18b)$$

$$\{\mathbf{R}_n, \mathbf{A}_n\} \overset{f_{\text{enc}}}{\leftarrow} \{\mathbf{V}, \mathbf{q}_n, \boldsymbol{\tau}_n\} \quad (18c)$$

$$\{\mathbf{R}^*_n, \mathbf{A}^*_n, \mathbf{q}^*_n, \boldsymbol{\tau}^*_n\} = f_{\text{opt}}(\mathbf{R}_n, \mathbf{A}_n, \mathbf{q}_n, \boldsymbol{\tau}_n). \quad (18d)$$

The algorithm converges when $\{\mathbf{R}^*_n, \mathbf{A}^*_n\} = \{\mathbf{R}^*_{n-1}, \mathbf{A}^*_{n-1}\}$.

Briefly explained, the core algorithm operates as follows: we initialize by encoding the sample video using a range of QPs and a flat QT (all entries = 16) as shown in (17). The iteration-optimal points are identified as the data point with the lowest bitrate R and those points with tracking accuracy A monotonically increasing from this point. To generate the data points for each iteration, each of the coefficients for every QT from the previous set of iteration-optimal data points is then set to 16 and 255 as per (16). We then proceed as per (18) by evaluating each data point (i.e., compressing the sample video using the QT being evaluated and tracking using the decompressed output) and afterwards finding those that are iteration-optimal among them.

Observe that given the operation of f_{opt} this algorithm uses a greedy search—only iterative results showing improvement (i.e., higher A for the same or lower R) are evaluated in subsequent iterations. Note also that the QT search is performed simultaneously across a range of q and τs. This is because tracking is a nonlinear process subject to different sources of distortion at different quantization levels—the cost and benefit of each QT coefficient is dependent on the QP being used. Also note that a single QT is used to code the entire video, i.e., the QT is not a macroblock or frame-level variable.

The above process can be performed in multiple ways based on the video input (compressed or uncompressed), its length, and the available time. In a real-time implementation this process will use only a few hundred input frames as discussed in Sections VI-A and VI-B. On the other hand, it can be completed offline over a range of input videos of greater length as discussed in Section VI-C, yielding a global QT that does not require any initialization delay to compute.

VI. Combined Implementation

The algorithms presented in Sections IV and V can each be independently deployed. In order to combine the possible bitrate reduction from using the two algorithms, we propose a combined system using them in series. While the basic runtime system (i.e., the part which is operational when the system is online) is straightforward, multiple variations of the system design can be obtained based on how the system is initialized (i.e., which QT is used and how it is derived).

Fig. 2 shows the combined runtime system. Raw digital video frames that are captured by the remote node (camera) are first passed through TDT$^-$, where the noise component is estimated and suppressed. This filtered video is then passed to the encoder, which can use a default "flat" QT or the QT specified during initialization. The compressed video and estimated noise parameter for each frame are transmitted to central processing, where the video is decoded and noise

synthesized per the estimation parameters is added back before automated tracking proceeds (i.e., TDT^{+} is performed). Overall, there exist multiple options for the quantization scheme used by the system, each having unique characteristics and accommodating a multitude of limited resource scenarios and applications such as follows:

1) default "flat" H.264/AVC quantization;
2) QT-LUT derived over a long initialization time, where high bitrate video is transmitted to central processing as described in Section VI-A;
3) QT-LUT derived over a short initialization time, where low bitrate TDT^{-} processed video is transmitted to central processing as described in Section VI-B;
4) global QT-LUT derived offline, as described in Section VI-C.

We consider TDT to be a key component of the system because it reduces noise which affects tracking accuracy. As such all QT-enabled scenarios assume the presence of TDT. A QT search performed on unprocessed video will result in an attempt to remove temporal noise in the frequency domain, potentially attenuating high frequencies, introducing smoothing and reducing the tracking accuracy of the final QTs.

A. Long Initialization Time System

The most aggressive method of QT initialization is possible via the long initialization time (LIT) system, shown in Fig. 5, which can initially require a long initialization delay and high bandwidth. As described in the figure, during initialization the LIT system encodes the captured sample video at a high bitrate and transmits this bitstream at the channel rate. At the receiver, after TDT processing is applied the reconstructed video is used as a source video estimate, used for automated tracking to generate a "ground truth" tracking baseline. This ground truth is used to calculate the tracking accuracy A of each QT search iteration. At the conclusion of initialization the final QT-LUT resulting from the iterative search is sent via the uplink to the remote node, which uses it for encoding during runtime until the next initialization phase.

B. Short Initialization Time System

A less aggressive alternative is possible via the short initialization time (SIT) system. Here, TDT^{-} is applied to the sample video from the remote node prior to encoding, thereby significantly reducing the required bitrate. The low bitrate video and noise estimate are transmitted to central processing, where noise synthesized per the estimate is added to the decoded video. The iterative QT search is performed using this noise-added video. The difference between SIT and LIT is the video that is used to generate the "ground truth" tracking baseline for the iterative QT search. For LIT, the high quality compressed video is used, whereas for SIT the compressed TDT video is used. The premise of SIT is that transferring TDT video to central processing will be significantly faster than transferring the original.

C. Global QT System

The least aggressive and easiest to deploy and operate method of QT utilization is possible via the global QT

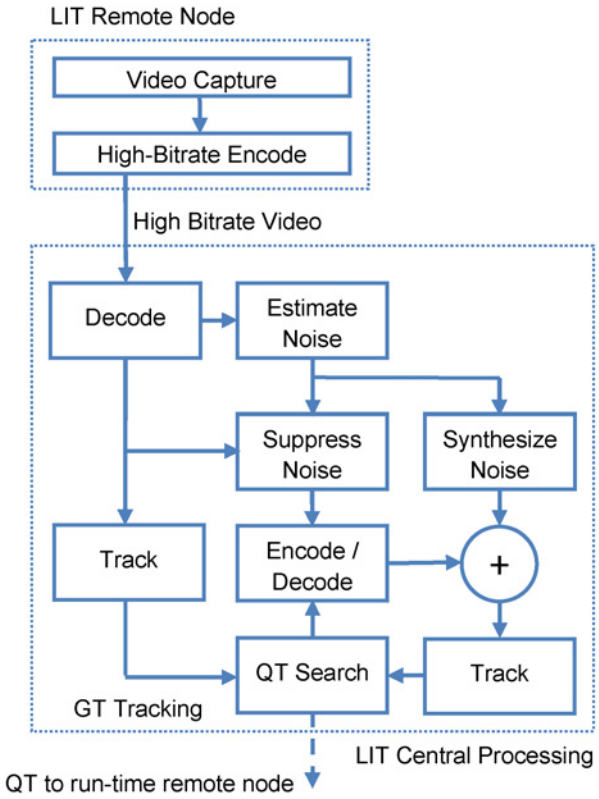

Fig. 5. LIT system initialization process.

(gQT) system. In the gQT scenario, a "tracker-focused scene-agnostic" QT-LUT is generated offline using video from many different scenes. This eliminates any need for a feedback loop between the remote nodes and central processing and imposes no system initialization delay. Note that for the gQT system no additional complexity regarding QT searching is introduced into the overall system—the gQT is computed offline and built into the remote nodes during deployment. Here, the QT search mechanism is similar to the central processing section of the LIT system shown in Fig. 5. However, instead of a single high bitrate video source captured from a remote node, raw content previously acquired from various cameras with different visibility conditions and viewing angles is used. In order to implement gQT we modify the QT search described in Section V by replacing (14) by

$$\{\overline{R}, \overline{A}\} \overset{f_{\text{enc}}}{\leftarrow} \{[\mathbf{V}^1, \mathbf{V}^2, ..., \mathbf{V}^K], q, \tau\} \tag{19}$$

where instead of a single video sequence $\mathbf{V}$ a range of K video sequences $[\mathbf{V}^1, \mathbf{V}^2, ..., \mathbf{V}^K]$ are encoded and tracked, and the average bitrate $\overline{R}$ and tracking accuracy $\overline{A}$ for all sequences is reported. The rest of the algorithm as described in (17) and (18) proceeds as normal using f_{enc}, resulting in a Global QT well suited for the variety of video sequences being considered.

D. Complexity Analysis

Real-world implementations of our algorithms would need to run real-time on low-cost remote nodes, such as consumer grade embedded systems. The premise of our design is that

the majority of additional resources required by our algorithms reside on the central processing location, while any additions to remote nodes are kept to a minimum.

Certain operations in our design, such as the *mode* operation used in (6), can be considered processor-intensive for some embedded systems. For our remote node system we considered low-cost consumer grade hardware such as the Texas Instruments (TI) OMAP3 System-on-Chip platform referred to in [28], which features a 520 MHz TI C64x DSP and ARM Cortex-A8 video processing engine. We consider 25% processor loading on such a platform to be reasonable in terms of additional resource requirements for our algorithms. Only an additional single frame buffer in which to store the previous processed frame output $\hat{\mathbf{V}}_{t-1}$ is required for TDT. Since the buffered B frames of $\mathbf{V}$ are not modified they can be shared with the video system—note that here we are referring to the buffer of original frames necessary for real-world system implementation, not the reference frame buffer of the video compression encoder. The gQT, SIT and LIT systems require no additional memory or processing power on remote nodes beyond what TDT requires.

In terms of added complexity introduced to central processing, only the processing power required for TDT need be considered. The gQT system requires no other operations. While the QT search required by the LIT and SIT systems involves tracking and analysis capability at the receiving end, this functionality is assumed to be present in the central processing station of a surveillance system capable of automated tracking. Given that the QT search is performed only during initialization, we can exclude it from the run-time processing load of the system. The noise estimate σ_t required for TDT can easily be encapsulated in the user data Network Abstraction Layer Unit of the H.264 stream being sent to central processing. In LIT and SIT systems the QT-LUT will be sent to remote nodes over the link as a standalone message.

One limitation of our QT search algorithm is that it is not exhaustive and is suboptimal, considering only individual modifications to $\{\mathbf{R}^*_{n-1}, \mathbf{A}^*_{n-1}\}$ when populating candidates for $\{\mathbf{R}^*_n, \mathbf{A}^*_n\}$. This was necessary to allow the algorithm to be able to operate within realistic time constrains considering real-world processing power and memory availability.

VII. Results

A. Experimental Framework

To demonstrate the gains possible with our algorithm a sample implementation of the system described in Sec. VI was tested using multiple sequences with different characteristics such as viewing angle, video quality and type of vehicle traffic observed. Details for the implementation and experimental procedure in addition to test results are presented below.

The video compression for the experiments presented herein was performed using the open-source H.264 encoder x264 [29] and the JM 16.0 H.264 reference decoder [30]. The open-source OpenCV [31] "blobtrack" module was used as the object tracker, which relies on the mean shift object tracking algorithm [21]. For some experiments, we also used the connected component (CC) tracker [22]. As part of the tracking module, tracker activity was limited to areas of transportation surveillance interest such as roads or sidewalks. This was done to better simulate real world transportation surveillance applications, which would concentrate on tracking vehicles and pedestrians as opposed to other objects such as trees or clouds.

TABLE I
Experimental Parameters

Encoder	Decoder	TDT$^+$ Noise Realizations	B (5)	C (8)	α,β,γ (4)
JM 16.0	x264	10	7	2	1/3

We used the parameters in Table I for our experiments. We used fixed QP rate control, with any variations in bitrate generated via varying the QP for the entire sequence, or in QT search experiments by varying both the QP and QT. No frame or macroblock level rate control was exercised. The search for the gQT system was done jointly over a library of sequences used in this paper and at various spatial resolutions.

For comparison we chose the LMMSE filtering algorithm presented in [11]. This algorithm is similar to those presented herein in that it is a low complexity, encoder-embedded video processing module aimed at removing noise from video. For a given bitrate LMMSE aims to maximize reconstructed video PSNR, while our algorithm specifically aims to preserve automated tracking accuracy. The LMMSE implementation used here was based on the JM 8.2 available at [30].

The following video sequences were used for testing.

1) The *Golf* sequence (resolution 720×480), shot by the authors on DV tape (a relatively high fidelity source), showing a local road intersection with normal traffic flow. Also visible are trees and parking lots for office buildings and a strip mall. There is little change to illumination during the sample. The video is interlaced, with little acquisition noise.
2) The *Camera6* sequence (resolution 640×480), available at [32], showing an intersection with light traffic, with trees swaying in the wind and buildings casting reflections of passing cars as part of the scene. Passing clouds also vary global illumination and create reflections on car surfaces. This "noisy" content had significant acquisition noise and was MPEG4 intra-only compressed during acquisition.
3) The *dt_passat* sequence (resolution 768×576), available at [33], showing a busy intersection with steady traffic interrupted by a traffic signal, and a light urban rail crossing. During the sample video global changes to illumination occur. This content is uncompressed luminance-only with significant acquisition noise as well as global illumination changes.

B. Experimental Results

Shown in Fig. 6 is an illustration of the joint operation of the TDT and QT systems. While there is a significant difference in the bitrates required for Fig. 6(c)–(e), the variance characteristics of regions to be tracked (i.e., the trajectories of cars and pedestrians) among the three, remain

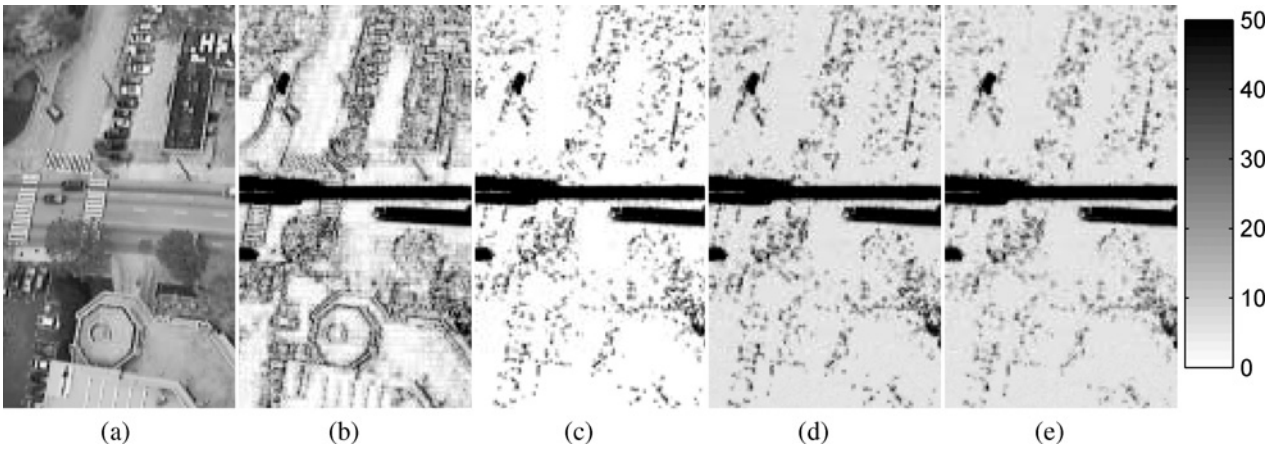

Fig. 6. Pixel intensity variances over 30 frames of the *Camera6* sequence. Here shown is (a) frame sample of the range being processed and, variances for the (b) unprocessed (original) and (c) TDT^{-} samples. (d) Sample from (c) is shown after compression using default "flat" quantization and TDT^{+}, while in (e) it is shown after compression using a QT from a LIT system and TDT^{+}.

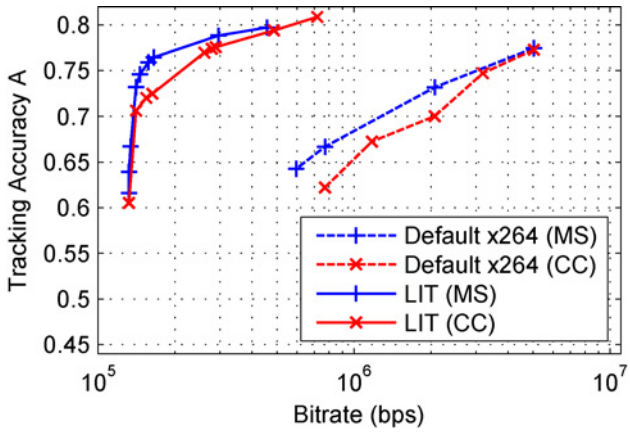

Fig. 7. Performance comparison of the MS and the CC trackers used as part of the LIT system for the textitdt_passat sequence.

largely unchanged. Note the major reduction in variance of the displaced frame difference (DFD), and therefore encoded bitrate, from Fig. 6(b) to (c). Observe that this can be attributed to the suppression of DFD in static regions of the frame such as buildings and roads. These variations would clearly have been due to noise, and therefore their suppression does not impact tracking accuracy while reducing bitrate by 80%. Likewise, note that Fig. 6(d) and (e) appear identical, especially along trajectories of interest (i.e., cars and pedestrians). This similarity is because in Fig. 6(e) frequency coefficients of low tracking impact (which are still costly to encode) have been suppressed, providing additional bitrate gains. Note that Fig. 6(d) and (e) have a global gray offset representing the variance of the added synthetic noise.

The QT search uses the tracker as part of optimization and as such our system can be considered tracker-agnostic. To illustrate that similar bitrate gains can be achieved using another tracker, in Fig. 7 we present a comparison of the performance of the LIT system using the CC tracker [22], where the system uses the CC tracker instead of the mean shift (MS) tracker [21] for all operations. Note in the figure that the bitrate reduction from using LIT over default coding

TABLE II
GLOBAL QT-LUT FOUND AFTER THREE ITERATIONS, AND THE CORRESPONDING SAMPLE TRACKING ACCURACIES

R (kb/s)	QP	QT	A_{mean}
145	32	[○ ● ● ● ○ ○ ● ● ○ ● ● ● ● ● ● ●]	0.652
156	32	[● ○ ○ ● ● ○ ● ● ○ ● ● ● ● ● ● ●]	0.743
185	32	[● ● ● ● ● ● ● ● ○ ● ● ● ● ● ○ ●]	0.757
308	28	[● ○ ○ ● ○ ○ ● ● ● ● ● ● ● ● ● ●]	0.772
322	28	[● ○ ○ ● ● ○ ● ○ ● ● ● ● ● ● ● ●]	0.794
368	28	[● ● ● ○ ● ● ○ ○ ● ● ○ ● ● ● ● ●]	0.810
702	24	[● ● ○ ○ ● ● ○ ● ● ● ○ ● ● ● ● ●]	0.823
760	24	[● ● ● ● ● ○ ○ ● ● ○ ● ● ● ● ● ●]	0.836

● = 16
○ = 255

is not significantly affected by whether the MS or CC tracker is used. In the following, all results are presented using the MS tracker.

The global QT-LUT shown in Table II was used to generate the gQT experimental results presented herein. In the table the bitrate ranges are presented in the first column, the optimized QP and QTs, respectively, in the second and third columns, and sample A values from our experiments in the last column. The QTs in the table are presented in the format described in (11) and (12). This QT-LUT was found using the MS tracker. Such a global QT-LUT can be built into any gQT system where it is known that the MS tracker is going to be used. The remote node would simply choose the appropriate QP/QT pair for compression based on the bitrate indicated by the available channel bandwidth. If it is known that another tracker will be used, that tracker should be used to search for the gQT. [Note that if desired it is possible to incorporate multiple trackers by averaging their respective tracking accuracies for (19).]

Experimental results from our test framework are presented in Fig. 8. Note that the TDT operating points form the seed values for the QT search, and therefore are the basis from which LIT, SIT, and gQT experiments are performed. As seen in Fig. 8, up to a 90% reduction in bitrate is possible using TDT alone. A further reduction of up to 50% of the TDT bitrate is possible using the QT search algorithms.

Note that only for the "dt_passat" experiments shown in Fig. 8(e), the SIT results at high bitrates actually underper-

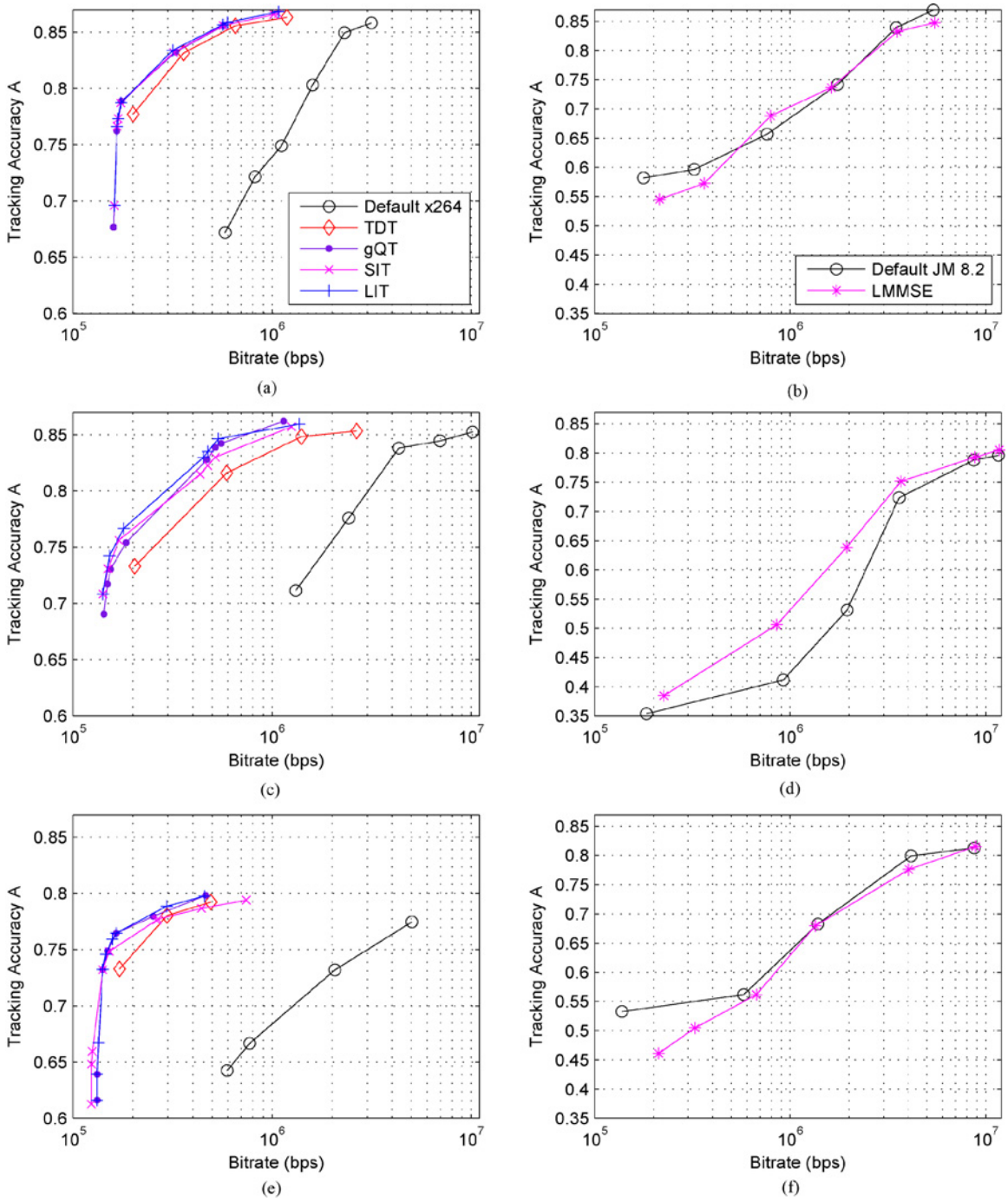

Fig. 8. Tracking accuracy A [(4), $\alpha = \beta = \gamma = 1/3$] across bitrates. Results on the left column (showing default, TDT, gQT, SIT, and LIT systems) from the proposed system. Results on the right column showing default JM 8.2 and LMMSE encoded video. Sequences used are (a), (b) *Golf*, (c), (d) *Camera6*, and (e), (f) *dt_passat*. The range of y-axis varies among the graphs to increase readability.

form their base TDT results. This is because the QT search is performed using a filtered and compressed estimate as ground truth for tracking (as opposed to one that is only compressed). As such some decisions are made that hurt rather than help tracking accuracy. Overall the gQT system performs on par with the LIT variant, despite using the global QT. The performance of the gQT system depends on the sequences used to train the gQTs. For our experiments we used variations of the three sequences discussed herein. Adding more sequences to this training set can decrease the performance gap even further. In Fig. 9, we show results for experiments performed using the *Rheinhafen* sequence (688×560) [33]. This sequence shows a busy intersection viewed by a low vantage point camera, which results in vehicles appearing much larger when closer to the camera. Note in the figure that even though a sequence significantly different in viewing angle than those in the gQT training set was used, the gQT system still allows for up to an additional 30% reduction in bitrate over the gains possible from using TDT alone.

TABLE III
SUMMARY OF ALGORITHMS AND RESULTS

System	Bitrate Gain %	Init. Time	Additional Resources	Comments
LMMSE	-11.3 ± 51.6	None	Encoder complexity	PSNR-optimized filter [11]
TDT	84.6 ± 3.8	Minimal	Encoder/decoder complexity	No uplink or no QT capability available
gQT	87.6 ± 4.4	Minimal	Encoder/decoder complexity	No uplink available or no initialization delay tolerable
SIT	86.4 ± 4.6	Short	Encoder/decoder complexity, uplink	Uplink available, only short initialization delay tolerable
LIT	88.1 ± 4.2	Long	Encoder/decoder complexity, uplink	Uplink available, long initialization delay tolerable

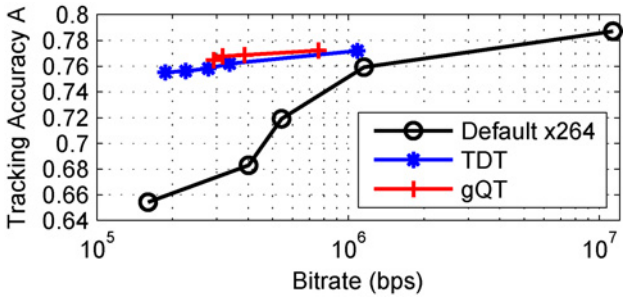

Fig. 9. Tracking accuracy across bitrates using the *Rheinhafen* sequence (showing default, TDT, and gQT systems). *Rheinhafen* is significantly different in terms of viewing angle when compared to the set of videos used to train the gQT system shown here.

Experiments for the LMMSE filtering system were compared to a baseline unmodified JM 8.2 encoder; refer to the right column in Fig. 8 for results. Note that the results shown in the figure indicate that both gain and loss in bitrate is possible from the use of the algorithm, that varies among the three sequences. This variability can be attributed to the fact that the algorithm uses an objective function based on maximizing reconstructed PSNR, which does not always translate to improved automated tracking accuracy. This observation motivating the use of tracking accuracy rather than PSNR as the metric driving rate decisions at the encoder is critical to the work presented herein.

Note from the results presented in Fig. 8 that the difference in performance between the LIT, SIT, and gQT systems is very small in most cases. This indicates that the decision as to which system to use should be made primarily based on specific system constraints.

C. Summary of Findings and Comparison of Systems

Table III summarizes the various aspects of the algorithms presented in this paper. In the table, bitrate gain ranges are reported as the mean $\pm$ standard deviation of the interpolated gains between algorithm and default compression results.

As indicated in Table III, PSNR-optimized algorithms such as the LMMSE filter [11] do not necessarily allow better automated tracking performance. The LIT system involves a potentially lengthy initialization time depending on the available channel rate and the chosen "high" bitrate for the ground truth estimation video. For example, 1 min of video compressed at a rate of 15 Mb/s and sent over a 512 kb/s channel will take 30 min to transmit. Where such long initialization times and initial bandwidth requirements are acceptable, LIT is ideal since it consolidates much of the system complexity in the central processing unit and provides the greatest reduction in required channel bandwidth for run-time operation. The SIT system offers similar advantages as the LIT system without requiring high bandwidth and long initialization times, at a moderate bitrate/accuracy performance penalty. In the example where LIT requires 30 min for initialization, assuming 90% TDT bitrate savings SIT would require only 3 min. Such a reduced initialization time would allow SIT to re-initialize the QT for a greater number of scenarios as compared to LIT.

Drastic changes such as the onset of heavy fog or rain require re-initialization in LIT and SIT systems. If such changes are frequent, the system downtime required for initialization can be impractical. On the other hand, the gQT system does not require any such downtime and can thus be considered as a more practical alternative.

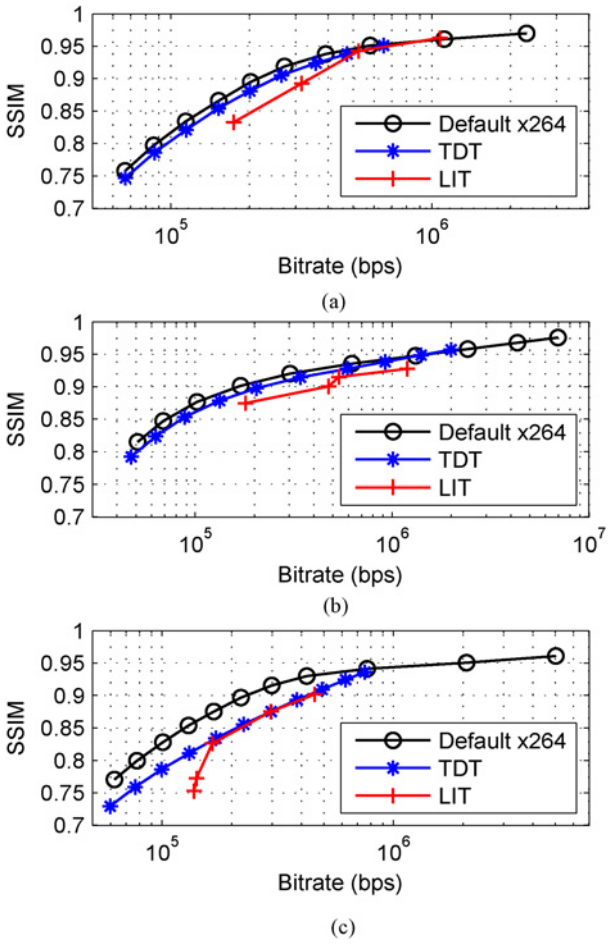

Fig. 10. Visual quality (measured by SSIM [34]) for TDT and LIT systems and default x264 for three sequences. (a) *Golf* sequence. (b) *Camera6* sequence. (c) *dt_passat* sequence.

Given that the gQT system does not optimize for specific scenes but only for a given tracker(s), it is expected that it may provide less reduction in bitrate compared to LIT or SIT systems for any given scene. It is suitable for applications where no initialization time is acceptable, or where the full-duplex communication between the remote node and central processing required for LIT and SIT systems is not available. Where QT support is not available (e.g., no H.264/AVC high profile support) TDT-only systems should be deployed.

To demonstrate that despite optimizing for tracking accuracy, we do not affect visual quality, we used the structural similarity (SSIM) metric [34], to estimate visual quality. PSNR cannot be used as a measure of video quality in our work due to its inability to account for denoising (achieved by TDT). SSIM (similar to PSNR) compares every frame of the encoded video for a given bitrate with the corresponding frame in the raw uncompressed video. We set SSIM parameters as in [34] and report the average SSIM over all frames. Fig. 10 shows that our algorithm does not significantly reduce visual quality while providing the dramatic bitrate gains shown in Fig. 8.

VIII. Conclusion

In this paper, we proposed a combined video processing and iterative quantization table search algorithm that removes elements of low tracking interest as part of the video compression system. We proposed three alternatives for system initialization, each appropriate for systems with different requirements. Using H.264/AVC video coding and a commonly used tracker, we showed that while maintaining comparable tracking accuracy our system allows for over 90% bitrate savings on the video link from remote nodes in centralized transportation surveillance systems. While in this paper we focused on transportation surveillance applications, the algorithms presented herein can readily be extended to other surveillance scenarios (such as maritime, rail, commuter terminals), and generalized applications whenever similar requirements (limited resources) and assumptions (static camera, motion of large objects, tracking awareness) are in place.

Acknowledgment

The authors would like to thank Dr. L. Guo and Prof. O. C. Au for providing the LMMSE implementation and the reviewers for their comments.

References

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

[2] F. Jiang, J. Yuan, S. A. Tsaftaris, and A. K. Katsaggelos, "Anomalous video event detection using spatiotemporal context," *Comput. Vis. Image Understanding*, vol. 115, no. 3, pp. 323–333, Mar. 2011.

[3] N. Anjum and A. Cavallaro, "Multifeature object trajectory clustering for video analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 11, pp. 1555–1564, Nov. 2008.

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

[5] K. Sayood, *Introduction to Data Compression*, 3rd ed. San Fransisco, CA: Morgan Kaufmann, 2005.

[6] M. Loomans, C. Koeleman, and P. de With, "Real-time scalable video codec implementation for surveillance," in *Proc. ICME*, Jun. 2009, pp. 1130–1133.

[7] D. Venkatraman and A. Makur, "A compressive sensing approach to object-based surveillance video coding," in *Proc. ICASSP*, Apr. 2009, pp. 3513–3516.

[8] V. Cevher, A. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *Proc. ECCV*, vol. 5303. 2008, pp. 155–168.

[9] T. Zgaljic, N. Ramzan, M. Akram, E. Izquierdo, R. Caballero, A. Finn, H. Wang, and Z. Xiong, "A compressive sensing approach to object-based surveillance video coding," in *Proc. VIE*, Jul. 2008, pp. 835–839.

[10] B.-F. Wu, Y.-L. Chen, C.-J. Chen, C.-C. Chiu, and C.-Y. Su, "A real-time wavelet-based video compression approach to intelligent video surveillance systems," *Int. J. Comput. Applicat. Technol.*, vol. 25, no. 1, pp. 50–64, Jan. 2006.

[11] L. Guo, O. C. Au, M. Ma, and P. H. W. Wong, "Integration of recursive temporal LMMSE denoising filter into video codec," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 2, pp. 236–249, Feb. 2010.

[12] M. Yang and W. Wang, "Fast macroblock mode selection based on motion content classification in H.264/AVC," in *Proc. ICIP*, vol. 2. Oct. 2004, pp. 741–744.

[13] W. K. Ho, W. Cheuk, and D. P. Lun, "Content-based scalable H.263 video coding for road traffic monitoring," *IEEE Trans. Multimedia*, vol. 7, no. 4, pp. 324–327, Aug. 2005.

[14] R. D. Sutter, K. D. Wolf, S. Lerouge, and R. V. de Walle, "Lightweight object tracking in compressed video streams demonstrated in region-of-interest coding," *EURASIP J. Adv. Signal Process.*, vol. 2007, no. 97845, p. 16, Jan. 2007.

[15] A. K. Kannur and B. Li, "Power-aware content-adaptive H.264 video encoding," in *Proc. ICASSP*, Apr. 2009, pp. 925–928.

[16] D. Schonfeld and D. Lelescu, "VORTEX: Video retrieval and tracking from compressed multimedia databases-visual search engine," in *Proc. Hawaii Int. Conf. Syst. Sci.*, vol. 3. Jan. 1999, p. 10.

[17] E. Soyak, S. A. Tsaftaris, and A. K. Katsaggelos, "Content-aware H.264 encoding for traffic video tracking applications," in *Proc. ICASSP*, Mar. 2010, pp. 730–733.

[18] E. Soyak, S. A. Tsaftaris, and A. K. Katsaggelos, "Quantization optimized H.264 encoding for traffic video tracking applications," in *Proc. ICIP*, Sep. 2010, pp. 1241–1244.

[19] E. Soyak, S. A. Tsaftaris, and A. K. Katsaggelos, "Tracking-optimal pre and post-processing for H.264 compression in traffic video surveillance applications," in *Proc. ICECS*, Dec. 2010, pp. 375–378.

[20] S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proc. VCIP*, vol. 5308, no. 1. 2009, pp. 881–892.

[21] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of nonrigid objects using mean shift," in *Proc. CVPR*, vol. 2. 2000, pp. 142–149.

[22] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," in *Proc. 2nd IEEE Workshop PETS*, Dec. 2001, pp. 1233–1243.

[23] M. B. A. Baumann, J. Ebling, M. Koenig, H. S. Loos, W. N. M. Merkel, J. K. Warzelhan, and J. Yu, "A review and comparison of measures for automatic video surveillance systems," *EURASIP J. Image Video Process.*, vol. 2008, no. 824726, p. 30, 2008.

[24] W. Hao, A. C. Sankaranarayanan, and R. Chellappa, "Online empirical evaluation of tracking algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1443–1458, Aug. 2010.

[25] J. S. Lee, "Digital image enhancement and noise filtering by use of local statistics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 2, pp. 165–168, Mar. 1980.

[26] A. H. Haddad, *Probabilistic Systems and Random Signals*. Englewood Cliffs, NJ: Prentice-Hall, 2006, p. 77.

[27] *Advanced Video Coding for Generic Audiovisual Services*, Recommendation H.264 and ISO/IEC 14 496-10 (MPEG-4) AVC, Int. Telecommun. Union-Telecommun. (ITU-T) and Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC) JTC 1, 2003.

[28] Texas Instruments. (2010, Dec.). *Texas Instruments OMAP3 Processors* [Online]. Available: http://focus.ti.com/docs/prod/folders/print/omap3530.html

[29] The VideoLAN Organization. (2010, Dec.). *The Open-Source x264 Video Codec* [Online]. Available: http://www.videolan.org/developers/x264.html

[30] K. Sühring. (2010, Dec.). *The Open-Source H.264/AVC Verification Model* [Online]. Available: http://iphome.hhi.de/suehring/tml
[31] G. Bradski. (2010, Dec.). *The Open-Source OpenCV Real-Time Computer Vision Library* [Online]. Available: http://opencv.willowgarage.com
[32] J. Halkias and J. Kolyar. (2010, Dec.). *Next Generation Simulation (NGSIM) Community* [Online]. Available: http://www.ngsim-community.org
[33] H. H. Nagel. (2010, Dec.). *KOGS/IAKS Karlsruhe Image Sequence Server* [Online]. Available: http://i21www.ira.uka.de/image_sequences
[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

Eren Soyak (S'10) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Northwestern University, Evanston, IL, in 2003, 2006, and 2011, respectively.

In 2003, he joined Ingenient Technologies, Rolling Meadows, IL, where until 2009 he developed embedded real-time video communication systems for a variety of applications including cellular telephony, handheld media, television broadcasting, surveillance, law enforcement, and defense. He is currently with AirTies Wireless Networks, Istanbul, Turkey. His current research interests include context-specific video processing and compression for real-time systems.

Sotirios A. Tsaftaris (S'02–M'06) received the M.Sc. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, in 2003 and 2006, respectively, and the Diploma degree in electrical and computer engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2000.

He is currently an Assistant Professor of Computer Science and Applications with Institutions Markets Technologies, Lucca, Italy, where he is a Founding Member of the Pattern Recognition and Image Analysis Unit. He is also an Adjunct Assistant Professor with the Departments of Electrical Engineering and Computer Science and Radiology, Feinberg School of Medicine, Northwestern University. Previously, he was a Research Assistant Professor with these departments from 2006 to 2011. He has published extensively, particularly in interdisciplinary fields (18 journal papers, 55 conference papers, and 4 book chapters). His current research interests include medical imaging analysis (MRI in particular), imaging for the life sciences, biomolecular computing applications, bioinformatics, data mining, digital copyright management, and image analysis applied to cultural heritage preservation.

Dr. Tsaftaris is a Murphy Fellow and a Fellow of the Alexander S. Onassis Public Benefit Foundation.

Aggelos K. Katsaggelos (S'80–M'85–SM'92–F'98) received the Diploma degree in electrical and mechanical engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1979, and the M.S. and Ph.D. degrees in electrical engineering from the Georgia Institute of Technology, Atlanta, in 1981 and 1985, respectively.

In 1985, he joined the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, where he was the Ameritech Chair of Information Technology and is currently a Professor holder of the AT&T Chair, the Director of the Motorola Center for Seamless Communications, a member of the Academic Affiliate Staff, NorthShore University Health System, an Affiliated Faculty Member with the Department of Linguistics, and he has an appointment with the Argonne National Laboratory. He is the author or co-author of more than five books, 170 journal papers, 420 conference papers, and 40 book chapters. He holds 17 patents.

Prof. Katsaggelos became a fellow of the International Society for Optical Engineers in 2009. He was a recipient of the IEEE Third Millennium Medal in 2000, the IEEE Signal Processing Society Meritorious Service Award in 2001, the IEEE Signal Processing Society Best Paper Award in 2001, the IEEE International Conference on Multimedia and Expo Paper Award in 2006, the IEEE International Conference on Image Processing Paper Award in 2007, an ISPA Paper Award in 2009, and the IEEE Signal Processing Society Technical Achievement Award in 2010. From 2007 to 2008, he was a Distinguished Lecturer of the IEEE Signal Processing Society.