TRACKING-OPTIMIZED QUANTIZATION FOR H.264 COMPRESSION IN TRANSPORTATION VIDEO SURVEILLANCE APPLICATIONS

E. Soyak [a], S. A. Tsaftaris [a,b] and A. K. Katsaggelos [a]

[a] Dept. of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA
[b] Dept. of Radiology, Feinberg School of Medicine, Northwestern University, Chicago, IL, USA
email: {e-soyak, s-tsaftaris}@northwestern.edu, aggk@eecs.northwestern.edu

ABSTRACT

We propose a tracking-aware system that removes video components of low tracking interest and optimizes the quantization during compression 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. The process of optimizing quantization tables suitable for automated tracking can be executed online or offline. The online implementation initializes the encoding procedure for a specific scene, but introduces delay. On the other hand, the offline procedure produces globally optimum quantization tables where the optimization occurs for a collection of video sequences. 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 while maintaining comparable tracking accuracy our system allows for over 50% bitrate savings on top of existing savings from previous work.

Index Terms— Urban traffic video tracking, transportation, video compression, quantization, preprocessing, postprocessing

1. INTRODUCTION

Video imaging sensors are commonly used in transportation monitoring and surveillance. In order to limit infrastructure costs associated with their deployment, most transportation video imaging solutions require the transmission of the video to a centralized location for viewing, (automated) analysis, and/or archiving. This centralized approach mandates the compression of video before it is transmitted. 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.

The subject of standard-compliant video compression optimized for surveillance applications was explored in [1], which focuses on concentrating (consolidating) bitrate on a Region of Interest (ROI) in the context of MPEG. More recently in [2] a more elaborate approach was proposed that adds higher level elements such as motion field correction filtering in the context of H.263. In [3] 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) to limit computation.

Within the scope of reducing bitrate and increasing video quality, a number of approaches have been suggested to reduce noise as much as possible [4] or to take into account the fact that the camera is stationary. However, the gains are in the domain of fidelity rather than automated trackability. In [5], automated tracking accuracy is proposed as a target metric in lieu of PSNR.

In this work we present an algorithm that when combined with the one in [5] allows for compression resources to be focused on video elements of tracking interest via the use of quantization tables. The process of optimizing quantization tables can be executed online or offline. The online implementation initializes the encoding procedure for a specific scene, thereby aggressively minimizing the bitrate requirement for that particular scene, but introduces delay. On the other hand, the offline procedure produces globally optimum quantization tables where the optimization occurs for a collection of video sequences. 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 inter-operability with other systems.

In Section 2 we briefly discuss the effects of video compression on the efficiency of tracking algorithms and present metrics for measuring the magnitude of these effects. In Section 3 we propose our joint method of tracking-optimized video processing and compression quantization, for which we show experimental results in Section 4. We present concluding remarks in Section 5.

2. COMPRESSION DISTORTION OF TRACKING

While the field of video object tracking contains a large variety of algorithms, most of these systems share some fundamental concepts. In a recent review of object tracking algorithms presented in [6] it is shown that most algorithms operate by modeling and segmenting foreground and background objects. The segmentation models and key features for a particular tracking application are chosen based on the application's goals and parameters. For example, color histograms can be useful when tracking highway vehicle activity during the day, but less useful under low light conditions at night. Most tracking algorithms, such as the Mean Shift tracker proposed in [7], account only for the native statistics of video objects, and distortion of these statistics by operations such as compression may severely degrade their accuracy. In [8], we discuss the effects of video compression which are especially debilitating for tracking algorithms, concluding that artifacts caused by higher compression ratios in general reduce tracking efficiency to a greater extent.

We refer to the closeness of the match between targets tracked in the uncompressed and compressed videos as tracking accuracy, which we measure using the *Overlap* ($OLAP$), *Precision* ($PREC$) and *Sensitivity* ($SENS$) metrics presented in [5]. In terms of the Ground Truth (GT) and Algorithm Result (AR), True Positives (TPs) are objects present in both the GT and AR, False Positives (FPs) are objects present in the AR but not in the GT, and False Negatives (FNs) are objects present in the GT but not in the AR. We define

$$OLAP = (GT_i \cap AR_i)/(GT_i \cup AR_i) \quad (1)$$

$$PREC = \overline{TP}/(\overline{TP} + \overline{FP}) \quad (2)$$

$$SENS = \overline{TP}/(\overline{TP} + \overline{FN}), \quad (3)$$

where GT_i are the segmented objects tracked in uncompressed video, the AR_i those tracked in compressed video, $\cap$ the intersection of the two regions, and $\cup$ their union. An FP is flagged if an object detected in the AR does not overlap an equivalent object in the GT, while an FN is flagged 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 these metrics we define the *aggregate tracking accuracy* A as

$$A = (\alpha * OLAP) + (\beta * PREC) + (\gamma * SENS), \quad (4)$$

where α, β and γ are weighting coefficients, such that $\alpha+\beta+\gamma = 1$.

3. PROPOSED METHOD

In this section we propose a system combining the algorithm in [5] with a new algorithm that determines a set of Quantization Table (QT) and quantization parameter (QP) pairs. Each pair maximizes tracking accuracy A for a given bitrate R. Below we propose the core algorithm used in the system, and then present online and offline initialization variants suitable for different application requirements.

3.1. Core Algorithm

First, we make use of the video processing algorithm presented in [5]. This temporal filtering algorithm, referred to here as *Temporal Deviation Thresholding* (TDT), operates in two parts which we briefly describe here. The first part models, detects, and removes temporal events of low tracking interest (considered to be "noise") prior to compression, operating via an iterative filter described as

$$\mathbf{M}_t = |\mathbf{V}_t - \mathbf{V}_{t-1}| > C * n_t \quad (5)$$

$$\hat{\mathbf{V}}_t = \mathbf{M}_t * \mathbf{V}_t + \overline{\mathbf{M}_t} * \hat{\mathbf{V}}_{t-1} \quad \forall\ t \geq B, \quad (6)$$

where for frame t of video sequence $\mathbf{V}$, $\mathbf{M}_t$ is a logical bitmap, $\overline{\mathbf{M}_t}$ is its logical inverse, C is a constant, n_t an estimated noise standard deviation, B the number of buffered frames being analyzed, and $\hat{\mathbf{V}}_t$ the filtered output given to the encoder. The noise standard deviation is estimated by observing frame statistics over the last B frames. After the video is decoded at the receiver, n_t (which was transmitted with the bitstream) is used to synthesize and re-insert noise prior to tracking. Part (a) aims to minimize the bitrate requirement, while part (b) aims to improve post-compression tracking results. For further details see [5].

Using TDT video, we propose an iterative gradient search algorithm which automatically identifies and concentrates bit allocation to frequencies useful to tracking, making encoder quantization decisions based on tracking accuracy as opposed to the traditional rate-distortion method. During each search iteration, the encoder quantization scheme for each individual frequency is modified, and tracking accuracy is measured for a sample video clip. From these results, only those frequencies which provide the highest tradeoff between bits and tracking accuracy are kept, and subsequent iterations proceed cumulatively. Details for the algorithm are presented below.

The quantization scheme is varied by the algorithm via the QT specified as part of the H.264/AVC Sequence and Picture Parameter Set structures. Each entry of the QT partially defines quantization of a coefficient resulting from the 4x4 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 QP and QT pairs for each bitrate.

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

$$\mathbf{QT} = [\phi_0, \phi_1, \phi_2, ..., \phi_{15}] \quad (7)$$

$$quant_j = q * \phi_j / 16, \quad (8)$$

where each 8-bit ϕ_j operates on a rasterized 4x4 H.264/AVC residual transform coefficient (e.g., ϕ_0 operates on position [1,1] and ϕ_{15} on position [4,4]). Therefore $2^{8\cdot16}$ distinct QTs are possible. We define 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 then define τ, a scalar to indicate the "structure" of the QT, as follows:

$$\tau = \sum_{j=0}^{15} 2^j * P_j, \quad where\ P_j = \begin{cases} 1 \text{ if } \phi_j = 16 \\ 0 \text{ if } \phi_j = 255 \end{cases} \quad (9)$$

The value of τ ranges from 1 to $2^{16}-1$, and each value uniquely determines the values of P_j. For example, $\tau = 2^{16}-1$ (all bit positions equal to 1) would indicate the use of a "flat" QT, whereas $\tau = 1$ (all bit positions but the zeroth equal to 0) would indicate a DC-only QT. We define the mapping operation f_{enc} as:

$$\{R, A\} \overset{f_{enc}}{\leftarrow} \{\mathbf{V}, q, \tau\}, \quad (10)$$

where R is the 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 with $\{R, A, q, \tau\}$, and denote as $\{\mathbf{R}, \mathbf{A}, \mathbf{q}, \boldsymbol{\tau}\}$ a collection 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:

$$\{\mathbf{R}^*, \mathbf{A}^*, \mathbf{q}^*, \boldsymbol{\tau}^*\} = f_{opt}(\mathbf{R}, \mathbf{A}, \mathbf{q}, \boldsymbol{\tau}). \quad (11)$$

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. We define the function f_{branch}, which generates the QT modifications necessary for the search iterations, as

$$\boldsymbol{\tau}_n = f_{branch}(\boldsymbol{\tau}^*_{n-1}), \quad (12)$$

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

$$\mathbf{q}_o = [QP_1, QP_2, ..., QP_M] \quad (13a)$$

$$\tau_{flat} = 2^{16} - 1 \quad (13b)$$

$$\boldsymbol{\tau}_o = [\tau_{flat}, \tau_{flat}, ..., \tau_{flat}] \quad (13c)$$

$$\{\mathbf{R}_o, \mathbf{A}_o\} \overset{f_{enc}}{\leftarrow} \{\mathbf{V}, \mathbf{q}_o, \boldsymbol{\tau}_o\} \quad (13d)$$

$$\{\mathbf{R}^*_o, \mathbf{A}^*_o, \mathbf{q}^*_o, \boldsymbol{\tau}^*_o\} = f_{opt}(\mathbf{R}_o, \mathbf{A}_o, \mathbf{q}_o, \boldsymbol{\tau}_o). \quad (13e)$$

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:

$$\mathbf{q}_n = \mathbf{q}^*_{n-1} \quad (14a)$$

$$\boldsymbol{\tau}_n = f_{branch}(\boldsymbol{\tau}^*_{n-1}) \quad (14b)$$

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

$$\{\mathbf{R}^*_n, \mathbf{A}^*_n, \mathbf{q}^*_n, \boldsymbol{\tau}^*_n\} = f_{opt}(\mathbf{R}_n, \mathbf{A}_n, \mathbf{q}_n, \boldsymbol{\tau}_n). \quad (14d)$$

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

Briefly explained, the core QT search 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 Eq. 13. 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, for every QT from the previous iteration-optimal data points each of the coefficients is in turn set to 16 and 255 as per Eq. 12. We then proceed as per Eq. 14 by evaluating each data point and afterwards finding those that are iteration-optimal among them.

Observe that given the operation of f_{opt} this algorithm uses a gradient search – only iterative results showing improvement (i.e., higher A for the same or lower R) are evaluated in subsequent iterations. Note that the QT search is performed simultaneously across a range of q and τ's. 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. Note also 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 online for a specific video scene for a QT-LUT specifically tailored for this scene, or offline for a large number of different video scenes for a more generalized QT-LUT that performs well for any scene. We consider TDT to be a key component of the system because it reduces noise which affects tracking accuracy. A QT search performed on unprocessed video will result in an attempt to remove temporal noise in the frequency domain, potentially attenuating high frequencies and reducing the tracking accuracies of the final QT-LUT.

3.2. Online Initialization System

The *online* method of QT initialization can initially require a long initialization delay and high bandwidth, but given that it is tailored to the specific scene for which it is initialized allows for the best performance. During initialization the online system encodes the captured sample video at a high bitrate and transmits this bitstream at the channel rate. At the receiver, TDT processing is applied, after which the reconstructed video is used as a source video estimate 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 the search the final QT-LUT is sent via the uplink to the remote node, which uses it for encoding during runtime until the next initialization phase.

3.3. Offline (Global) Initialization System

The less aggressive but easier to deploy and operate method of QT utilization is possible via the *global QT* (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 online system. 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 above by replacing Eq. 10 by:

$$\{\overline{R}, \overline{A}\} \overset{f_{enc}}{\leftarrow} \{[\mathbf{V}^1, \mathbf{V}^2, ..., \mathbf{V}^K], q, \tau\}, \tag{15}$$

where instead of a single video sequence $\mathbf{V}$ a set of K video sequences $[\mathbf{V}^1, \mathbf{V}^2, ..., \mathbf{V}^K]$ are encoded, and the average bitrate $\overline{R}$ and tracking accuracy $\overline{A}$ for all sequences is reported. The rest of the algorithm as described in Eqs. 13-14 proceeds as normal using f_{enc}, resulting in a Global QT well suited for the variety of video sequences being considered.

Table 1. Global QT-LUT found after 3 iterations of training, and the corresponding sample tracking accuracies.

R (kbps)	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

4. EXPERIMENTAL RESULTS

To demonstrate the gains possible with our algorithm a sample implementation of our system was tested using multiple sequences with different characteristics such as viewing angle, video quality and type of 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 [9] and the JM 16.0 H.264 reference decoder [10]. The open-source OpenCV [11] "blobtrack" module was used as the object tracker, which relies on the Mean Shift object tracking algorithm [7].

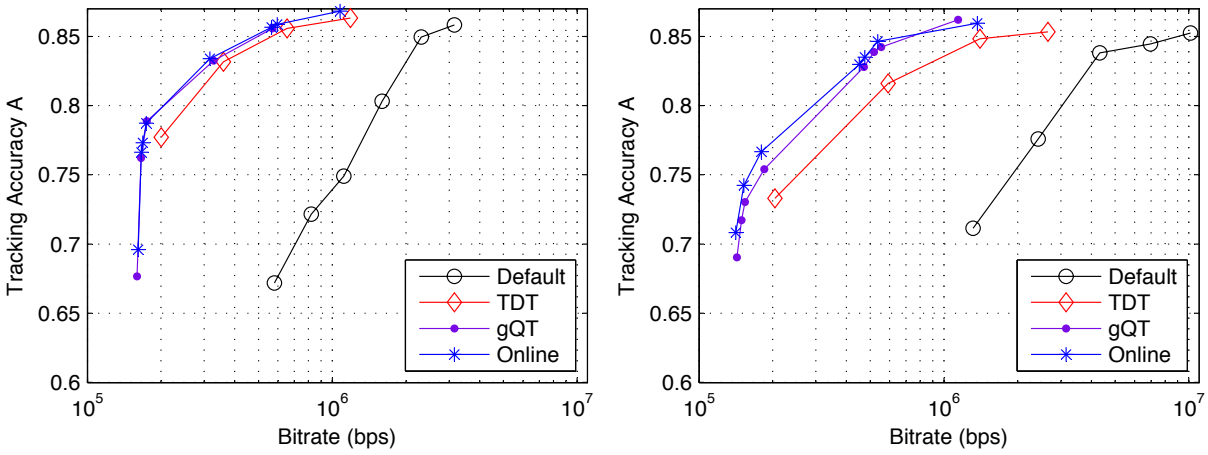
For comparison we chose the LMMSE filtering algorithm presented in [4]. This algorithm is similar to those presented herein in that it is a low complexity video processing module aimed at removing noise from video. For a given bitrate LMMSE aims to maximize reconstructed video PSNR, while our algorithm specifically aims to preserve automated tracking accuracy. The LMMSE implementation used here was based on the JM 8.2 available at [10].

We used the following parameters for our experiments. For TDT, we generated 10 realizations of Gaussian noise per sequence, which were used for all experiments performed using that sequence. As TDT experimental constants we used the threshold $C = 2$ and buffer size $B = 7$. 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 (i.e., no frame or macroblock level rate control). The training for the gQT system was done jointly over a database of videos (including the sequences used in this work) at various spatial resolutions.

The following videos were used for testing. The "Golf" sequence (720x480) was shot on DV tape and is a relatively high fidelity source, showing a local road intersection with steady non-rush traffic. As part of the scene there are trees and parking lots for office buildings and a strip mall. The "Camera6" sequence (640x480) was used under the NGSIM license courtesy of the US FHWA. It shows an intersection with light traffic, with trees swaying in the wind and buildings casting reflections of passing cars as part of the scene. This video was MPEG4 intra-only compressed during acquisition and is thus significantly noisier than the "Golf" sequence.

The global QT-LUT shown in Table 1 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 Eqs. 7-8. Such a global QT-LUT can be built into any gQT system, where the remote node would simply choose the appropriate QP/QT pair for compression based on the bitrate indicated by the available channel bandwidth.

Experimental results from our test framework are presented in Fig. 1. Note that the TDT operating points form the seed values for the QT search, and therefore are the basis from which online and



(a) "Golf" sequence (b) "Camera6" sequence

Fig. 1. Tracking accuracy for default, TDT (from [5]), gQT and online systems using the (a) "Golf" and (b) "Camera6" sequences.

Table 2. Comparison of algorithms and results.

System	Bitrate Gain %	Suitable Deployment
LMMSE [4]	-11.3 $\pm$ 51.6	Only human observation intended
TDT [5]	84.6 $\pm$ 3.8	No uplink or no QT capability
gQT	87.6 $\pm$ 4.4	No uplink or no init. tolerable
Online	88.1 $\pm$ 4.2	Uplink available, long init. tolerable

gQT experiments are performed. A reduction of up to 50% of the TDT bitrate is possible using QT search. Overall gQT performs as well as the online system. The performance of gQT depends on the sequences used offline to train the system. For our experiments we used various sequences (including the two discussed herein).

Table 2 summarizes the various aspects of the algorithms presented in this work. 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 the table, PSNR-optimized algorithms such as the LMMSE filter [4] do not necessarily allow better automated tracking performance. The online system involves a potentially lengthy initialization time depending on the available channel rate and the chosen "high" bitrate for the ground truth estimation video. Where such long initialization times and initial bandwidth requirements are acceptable, the online system 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.

Drastic changes such as the onset of heavy fog or rain require re-initialization in the online system. If such changes are frequent, the system downtime required for initialization can be impractical. On the other hand gQT does not require any such downtime and can thus be considered as a more practical alternative. To illustrate the loss of performance suffered by the online system in the absence of proper re-initilization we used the QT-LUT which was optimized on the "Golf" sequence to compress the "Camera6" sequence. This scenario was outperfomed by gQT on average by 63%.

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

5. CONCLUSION

We have 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 propose online and offline alternatives for system initialization, each appropriate for systems with different requirements. Using H.264/AVC video coding and a commonly used tracker, we show that while maintaining comparable tracking accuracy our system allows for over 50% bitrate savings on top of existing 90% savings from TDT processing.

Acknowledgements: The authors would like to thank Dr. Liwei Guo and Prof. Oscar C. Au for their LMMSE implementation. This work was supported in part by the Northwestern Center for the Commercialization of Innovative Transportation Technology (CCITT).

6. REFERENCES

[1] N. Zingirian, P. Baglietto, M. Maresca, and M. Migliardi, "Video object tracking with feedback of performance measures," in *Proc. ICIAP*, vol. 2, Florence, Italy, 1997, pp. 46–53.

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

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

[4] L. Guo, O. C. Au, M. Ma, and P. H. W. Wong, "Integration of recursive temporal LMMSE denoising filter into video codec," *IEEE Trans. On Circ. And Sys. For Video Tech.*, vol. 20, no. 2, pp. 236–249, Feb. 2010.

[5] 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*, Athens, Greece, Dec. 2010, pp. 380–383.

[6] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, pp. 13.1–13.45, 2006.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. CVPR*, vol. 2, Hilton Head, SC, USA, 2000, pp. 142–149.

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

[9] (2010, Dec.) The open-source x264 video codec. [Online]. Available: http://www.videolan.org/developers/x264.html

[10] (2010, Dec.) The open-source H.264/AVC verification model. [Online]. Available: http://iphome.hhi.de/suehring/tml/

[11] (2010, Dec.) The OpenCV real-time computer vision library. [Online]. Available: http://opencv.willowgarage.com