COMPENSATION METHODS USING SIGNAL PROCESSING AND ADAPTIVE QUANTIZATION FOR BETTER MEAN SHIFT TRACKING ON COMPRESSED VIDEO

Salman Aslam, Aaron Bobick, Christopher Barnes

Georgia Institute of Technolgy

ABSTRACT

In this paper, we discuss methods for coding MPEG-4 video to enable better computer vision on the compressed video. This problem is important since the amount of video content being generated from a variety of sources is increasing very rapidly, almost all this video is in compressed form, and computer vision applications such as video analysis, mining and querying running on these videos are becoming more commonplace. If the particular computer vision algorithm that is likely to be run on the compressed video is known apriori, then steps can be taken during the encoding process to facilitate the performance of the algorithm. In this paper, the algorithm we focus on is Mean Shift tracking, a robust and widely used tracker. We show that by performing signal processing on the input signal before it is encoded, or by adaptively changing the parameters of the encoding process, we can make the resulting signal more robust to degradations in the encoding process. The result is better tracking on the compressed video, at the same bitrate, but with some loss in PSNR.

Index Terms— Computer vision, video compression, surveillance, MPEG-4, Mean Shift tracking

1. INTRODUCTION

1.1. Problem statement

Figure 1 presents the overall framework we are interested in. It can be summarized in three points (a) A video signal from a given source needs to be transmitted to another point or stored in some medium. (b) Due to bandwidth or storage limitations, we would like to encode the video. (c) Depending on the application, some measure of information extracted from the original video needs to be preserved in the decoded video.

1.2. Motivation

Examples where the above situation may arise are many. The increasing trend in video sharing brought on by the popularity of sites such as YouTube [1] and Facebook [2] or applications such as MMS, has brought with it novel ways of analysing and mining content. In these scenarios, computer vision algorithms such as face detection or face finding running on compressed videos from these sources is becoming common. In more traditional settings, video cameras in surveillance applications normally send back compressed video to a central server for analysis. Although there is a rising trend in edge processing, i.e. processing at the camera itself, the need to transmit compressed video to a central server for human consumption and data fusion from other sensors still remains important. Another increasing trend in surveillance applications, usage of live webcams streaming data over the internet, remains relevant to our discussion since these webcams also encode data before transmission. In satellite imaging applications, compressed video downlinked to ground stations may be analyzed for atmospheric and geological patterns, disaster zone identification, vegetation analysis or structure classification. In medical imaging, there is a rising trend in remote diagnosis and surgery, with compressed video being sent to doctors in remote locations.

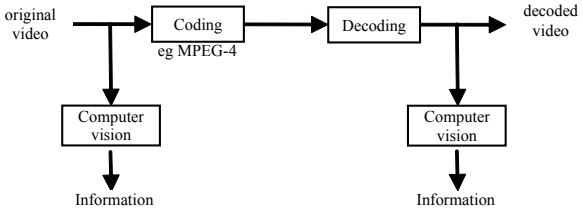


Fig. 1. Information in the decoded video must be close in some sense to information in the original video.

1.3. Possible Approaches

In the applications mentioned above, and in hundreds of other applications where it is important to preserve some measure of information in the video signal other than visual quality, the goal of coding video can be addressed in the following ways,

- **Specific codec for every application.** Design a video codec tailored for each application that efficiently concentrates bits where they matter most. The disadvantages in terms of cost, maintainability, upgradeability and cross operability are obvious.
- **Standard codec with metadata.** Provide a general

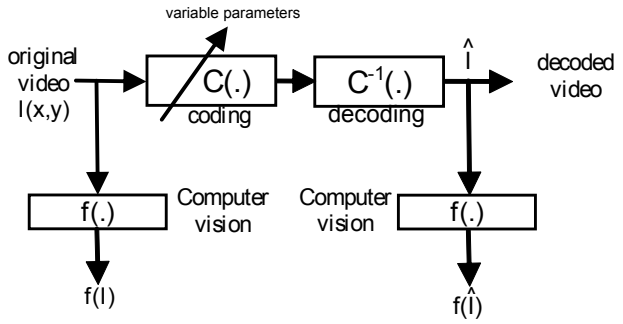


Fig. 2. Encoder $VarPar$ (variable parameter) compensation.

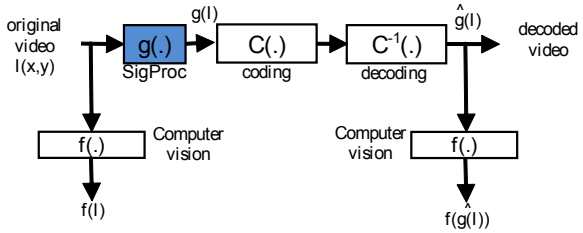


Fig. 3. $SigProc$ (signal processing) compensation.

framework for transmitting both video and information extracted from the video in a coherent and integrated manner as part of the codec specification. This approach was adopted by MPEG in 1993 and resulted in MPEG-4. However, in practice, the scene construction and description features of this standard did not gain much acceptance.

- **Standard codec.** Use standard universally accepted rectangular based MPEGx and H.26x codecs and change their parameters during the encoding process to preserve the desired information. We call this approach $VarPar$ (variable parameter) compensation.

- **Signal processing on the input.** In places where access to the encoding process is not available or not desired, apply signal processing on the input signal to make it more robust to degradations during the encoding process. We call this approach $SigProc$ (signal processing) compensation.

We are not interested in the first two methods discussed above due to their limitations. We now discuss $VarPar$ and $SigProc$ compensation in the following paragraphs.

1.3.1. VarPar Compensation

This approach is depicted in Figure 2. In this approach, the encoding parameters are adjusted adaptively based on the input signal to maximize some measure of information in the input signal. $VarPar$ compensation has been used in the literature for computer vision algorithms that rely on motion. In

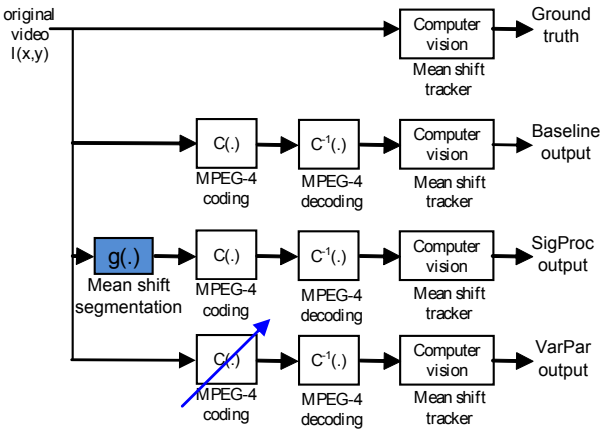


Fig. 4. Experimental setup.

such cases, researchers have preferentially encoded areas with high motion, or in more sophisticated cases, preferentially encoded the foreground [3]. Very little literature is available for computer vision algorithms that do not rely on motion.

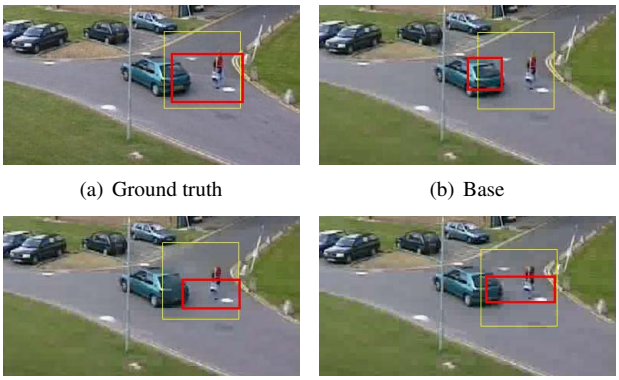
1.3.2. SigProc Compensation

This approach is depicted in Figure 3. Very little literature is available on this approach. The idea here is to transform the input signal so that it better preserves some measure of information as it undergoes distortions in the encoding process. $SigProc$ can be applied in two ways:

- No inverse signal processing required. Signal processing can be applied on the input signal so as to maximize a similarity measure between input and output. In such cases, the output signal will remain intelligible to a human observer. This is important where no further processing after the decoding process is desired.

- Inverse signal processing required. In cases where human consumption of the decoded signal is not required, or where extra processing after the decoding step is possible, one may apply signal processing to severely distort the input signal in a way that it better preserves some measure of information during the encoding process. MPEGx and H.26x, the two prevalent standard video coding standards compress video so that low frequency scene reconstruction elements most useful for the human visual system remain preserved. One can then try to process the input signal so that desired information content is encoded at lower frequencies.

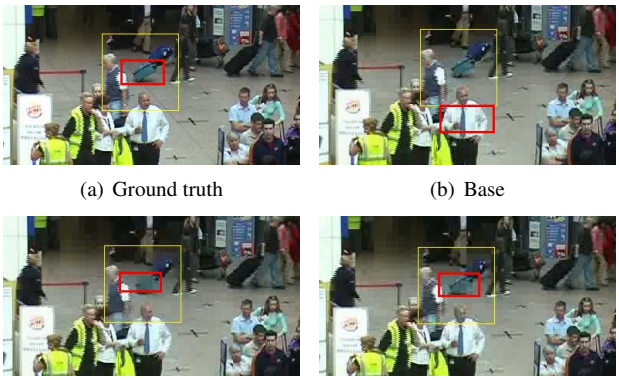
  More formally, $SigProc$ can be cast as a calculus of variations problem where we seek to minimize the functional J(g)



(a) Ground truth (b) Base

(c) SigProc (d) VarPar

Fig. 5. PETS2001



(a) Ground truth (b) Base

(c) SigProc (d) VarPar

Fig. 6. PETS2007

$$J(g) = \int \int_D \|f(I) - f(\hat{g}(I))\| + \|p(I) - p(\hat{g}(I))\| \, dxdy \tag{1}$$

$\hat{g}(I)$ can be written as $C^{-1}(C(g(I(x,y))))$ where $C(x,y)$ is

$$C(x,y) = \sum_{x_0} \sum_{y_0} T(Q(D(I)))\delta(x - kx_0, y - ky_0) \tag{2}$$

Here $I(x,y)$ is the original image. $D(x,y)$ is the DCT transform of a kxk block of image data or motion compensated residuals. The value of k is normally 8. $Q(x,y)$ is a quantization function and can depend on quantization matrices and the quantization parameter Qp. $T(x,y)$ is a thresholding function. We omit the lossless parts of the compression system, such as Huffman Coding, Arithmetic Coding etc, since they do not result in distortions in our setup. There is an additional element of numerical precision errors, particularly in the DCT setup that we choose to ignore for simplicity. $f(x,y)$ is the computer vision algorithm that we're interested in.

Equation 1 then says that over the space of all transformations that can be applied to the input image $I(x,y)$, we would like to find the signal processing function $g(x,y)$ that minimizes the distance between the output of a computer vision algorithm $f(x,y)$ running on the original image and running on the compressed image, and that also maximizes some similarity measure $p(x,y)$ of the image. It is clear that Equation 1 is not differentiable due to the discontinuities on the kxk DCT block boundaries. However, even if numerical schemes are resorted to, the Euler Lagrange approach to solving the variational problem is a necessary but not sufficient condition for global optimality. In such cases, one can resort to finding a signal processing function $g(x,y)$ either experimentally through Monte Carlo simulations, or using prior knowledge of the nature of the computer vision algorithm we are interested in.

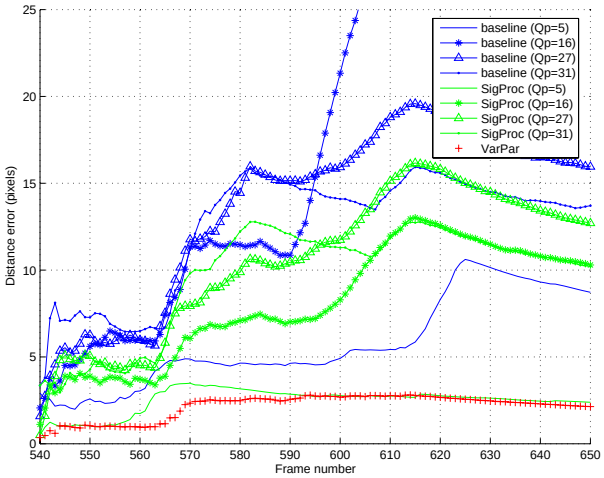
2. EXPERIMENTS

After having discussed the general approaches to our problem, we now discuss the specific approach that we use in this paper. The computer vision algorithm that we use is Mean Shift tracking. Before proceeeding with other specifics of our approach, we give an overview of this algorithm.

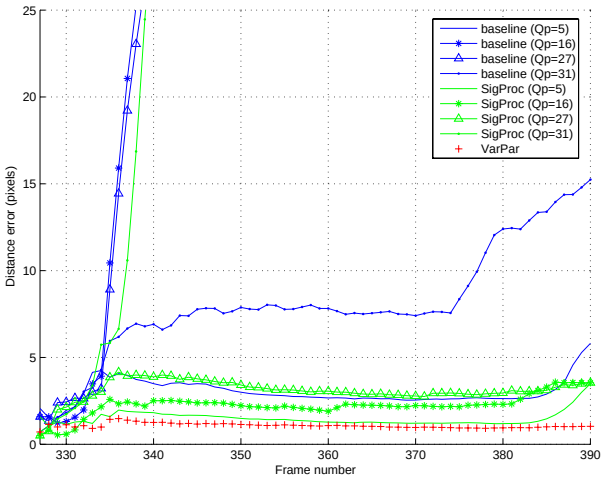
$$\begin{aligned} M_{00} &= \sum_x \sum_y I(x,y) \\ M_{10} &= \sum_x \sum_y xI(x,y) \\ M_{01} &= \sum_x \sum_y yI(x,y) \\ x_c &= \frac{M_{10}}{M_{00}} \\ y_c &= \frac{M_{01}}{M_{00}} \end{aligned} \tag{3}$$

2.1. Theory: Mean Shift Tracking

The goal of the Mean Shift algorithm is to find the peak of a distribution over time [4]. An initial reference histogram is created in some desired space, of the object to be tracked. After creating the initial histogram, subsequent images are backprojected on this histogram to create a likelihood function. A window

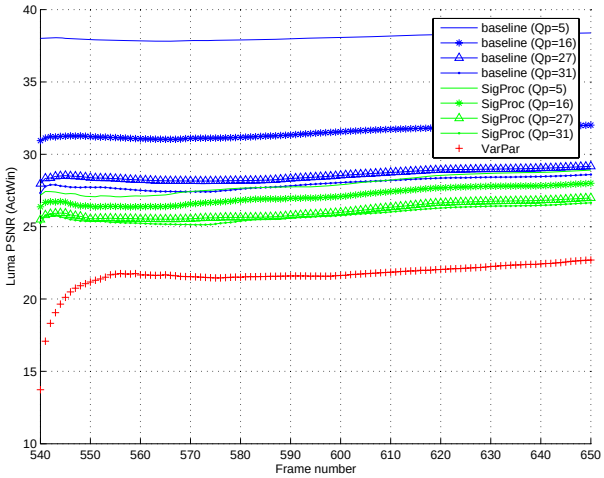


(a) PETS2001

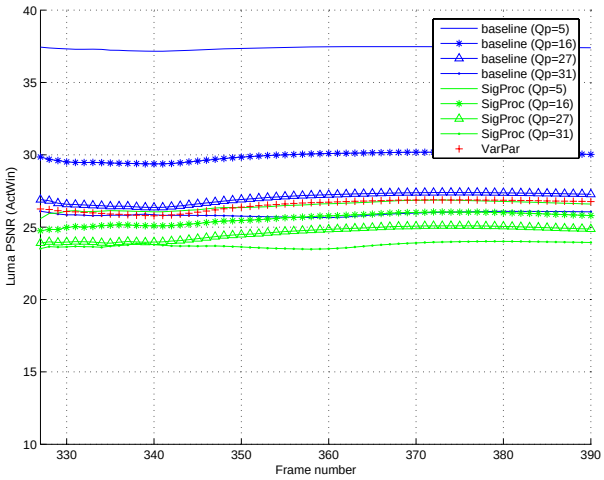


(b) PETS2007

Fig. 7. Tracking accuracy.



(a) PETS2001



(b) PETS2007

Fig. 8. PSNR comparison in $ActWin$.

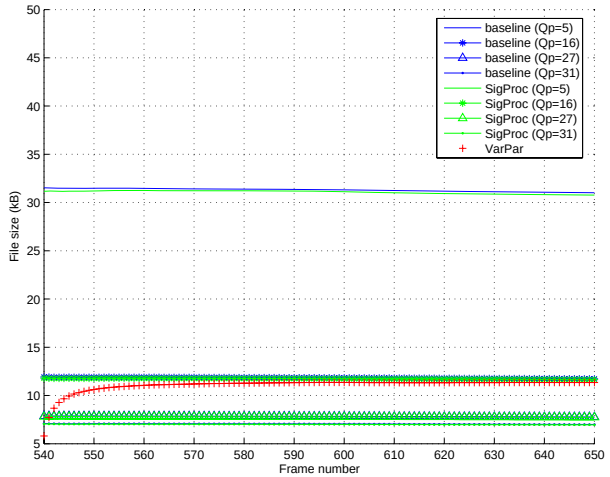
placed over the initial location of the target object is then placed on this backprojected window and moved to a new place as computed by the mean shift vectors, x_c and y_c (Equation 3). If a rectangular kernel for the window is used [5], [6], then the mean shift computations reduce to finding the zeroth and first order image moments as shown in Equation 3. This step is repeated until the mean shift vectors converge.

2.2. Components Of Experimental Setup

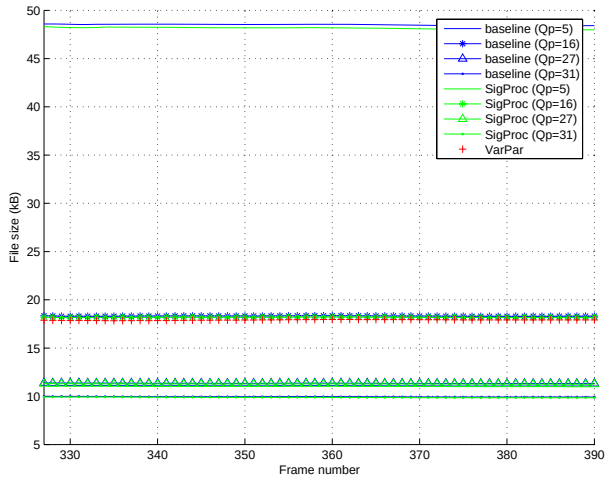
Our experimental setup is given in Figure 4. The different components in this setup are explained below.

– **Input data.** For the input data, we used image sequences from two standard databases, PETS2001 and PETS2007 (Performance Evaluation of Tracking and Surveillance). In PETS2001, we track a person in a sparse background as she gets occluded by a car with a similar color distribution. In PETS2007, we track a bag in a dense environment. These two scenarios were chosen since they are quite different from each other and would provide a better test of our approach. Additionally, in both cases, the tracked object is occluded by another object with a similar color distribution. This of course, presents one of the most significant challenges in tracking applications.

– **Computer vision algorithm.** As mentioned earlier, the specific computer vision algorithm that we chose to experiment with is the Mean Shift tracker. The implementation used is from the standard open source Intel OpenCV library. The feature distribution used is the hue component of

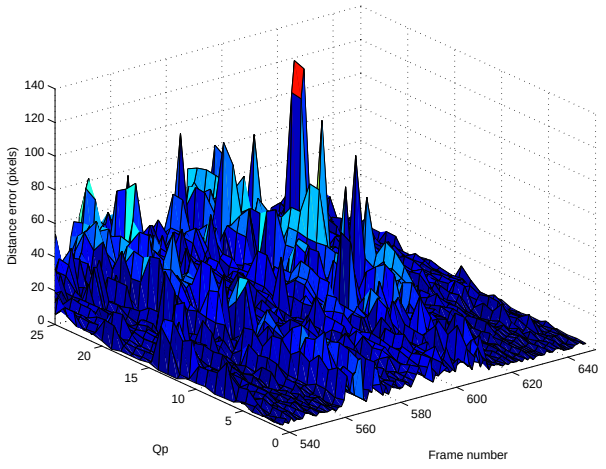


(a) PETS2001

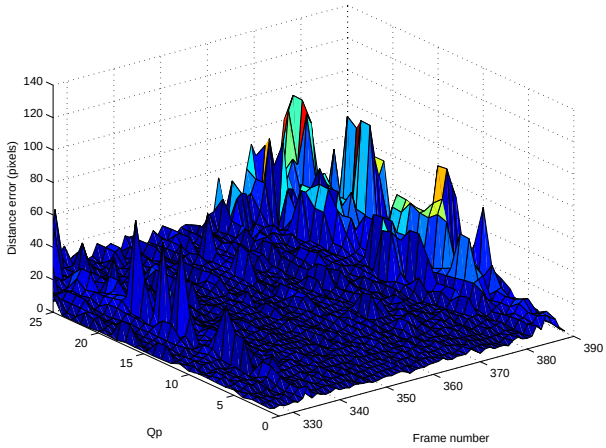


(b) PETS2007

Fig. 9. File size comparison.



(a) PETS2001



(b) PETS2007

Fig. 10. $VarPar$, tracking accuracy at different values of Qp.

the HSV space. Convergence takes 3 to 4 steps on the original and compensated images and up to 8 to 9 steps for the uncompensated images. We manually initialize the object to be tracked. In the case of static cameras, it could come from a background modeling process such as the Multi Gaussian algorithm [7], [8]. However, we make no assumptions about this step, and our approach is equally valid for moving cameras.

– **Codec.** The codec used is a standard MPEG-4 Part 2 Visual codec obtained from ISO. The reason for not using an H.264 codec is that some of our initial experiments tried to use object based functionality of this codec, which is not provided in H.264.

– **Compensation methods.** The compensation methods, as discussed above were $SigProc$ and $VarPar$, both applied independently. $SigProc$ was applied in a way to maximize similarity with the input image so that no further processing would be required after the decoder.

– **Outputs.**

∗ Ground truth: Ground truth tracking output was obtained by running Mean Shift tracking on the input sequences.

∗ Baseline: Baseline tracking output was obtained by running Mean Shift tracking on MPEG-4 coded input sequences. The quality of the video was changed using the quantization parameter Qp. Quantization matrices were not adjusted. Qp was adjusted to take on one of the following 4 values: 5, 16, 27 and 31.

∗ $SigProc$ output: $SigProc$ tracking output was obtained by running Mean Shift track-

ing on MPEG-4 coded, signal processed, input sequences. The signal processing algorithm that we picked is Mean Shift segmentation, which uses the same technique as the Mean Shift tracker to center its window on local histogram peaks. At this stage, we made three decisions. First, the class of the signal processing algorithm to be used, i.e. segmentation. Second, the particular algorithm within the class, i.e. Mean Shift segmentation. Third, for the parameters of the algorithm, we iterated over the process of segmentation, compression, decompression, tracking, comparing results with the ground truth, and then choosing segmentation parameters that produce the best results. All this was done automatically. The reason for choosing Mean Shift segmentation was that since the Mean Shift tracker needs the backprojected window to compute its motion vectors, Mean Shift segmentation should provide a better backprojection. Even though that backprojection would undergo distortions through the MPEG-4 codec, it would still better preserve local hue information. We did not segment the whole image, but only a window placed around the object of interest. This window is different from the Mean Shift tracker window and is called the Active Window, or $ActWin$ hereafter, for clarity. $ActWin$ was initialized in the same step as the Mean Shift tracker, and from there on, its position was updated automatically.

  - $VarPar$ output: $VarPar$ tracking output was obtained by running Mean Shift tracking on MPEG-4 coded input sequences. Qp was iterated over inside $ActWin$ and the best Qp chosen to encode the area inside $ActWin$. The Qp for the rest of the image outside $ActWin$ was kept same as for the baseline sequence.

– **Metrics.** We compare tracking performance for $SigProc$ and $VarPar$ with baseline tracking output using 3 metrics:

  - Performance. This is a comparison with baseline Euclidean distance from ground truth (of bounding box center, measured in pixels).
  - Quality. This is a comparison with baseline PSNR inside $ActWin$.
  - Bandwidth/bitrate. This is a comparison with baseline Intra coded file size.

Accuracy				
Dataset	Qp	Baseline	$SigProc$	$VarPar$
PETS2001	5	8.7	2.4	2.1
	16	75.4	10.3	2.1
	27	16.0	12.7	2.1
	31	13.7	10.2	2.1
PETS2007	5	5.8	3.5	1.0
	16	85.1	4.2	1.0
	27	71.8	3.5	1.0
	31	15.2	35.8	1.0
Average	-	36.5	10.3	1.6

Table 1. Distance from ground truth.

Quality				
Dataset	Qp	Baseline	$SigProc$	$VarPar$
PETS2001	5	38.4	28.9	22.7
	16	32.0	28.0	22.7
	27	29.2	27.0	22.7
	31	28.6	26.6	22.7
PETS2007	5	37.4	26.6	26.8
	16	29.9	25.7	26.8
	27	27.3	24.9	26.8
	31	26.1	23.9	26.8
Average	-	31.1	26.45	24.72

Table 2. PSNR(dB).

3. RESULTS

Tracking results are given in Tables 1, 2 and 3. Table 1 shows the Euclidean distance in pixels, of bounding box centers, produced by the baseline, $SigProc$ and $VarPar$ outputs. We can see that compared to the baseline, $SigProc$ produces 3.5 times and $VarPar$ produces 22.8 times more accurate results. This increased accuracy comes at a price. Table 2 shows that price. We see that $SigProc$ loses 4.65 dB inside $ActWin$ compared with the baseline, while $VarPar$ loses 6.38 dB. As far as file size is concerned, which we use

Bitrate				
Dataset	Qp	Baseline	$SigProc$	$VarPar$
PETS2001	5	31.0	30.8	-
	16	11.7	11.6	11.3
	27	7.8	7.7	-
	31	7.0	7.0	-
PETS2007	5	48.4	48.0	-
	16	18.3	18.2	17.9
	27	11.3	11.3	-
	31	9.9	9.8	-
Average	-	18.2	18.0	-

Table 3. File sizes.

as an estimate of the bitrate, $SigProc$ has 0.99 of the baseline's bitrate, while $VarPar$ has 0.97 of the baseline's bitrate. Comparison of $VarPar$ is at $Qp = 16$ as shown in Table 3. So the tradeoff is not in terms of bitrate, where $SigProc$ and $VarPar$ actually do better but in terms of PSNR inside $ActWin$.

In Figures 7(a) and 7(b), Figures 8(a) and 8(b) and Figures 9(a) and 9(b), we show accuracy, PSNR and filesize for PETS2001 and PETS2007 respectively plotted against time, i.e. frames. As far as accuracy results are concerned, it is clear from these figures that $VarPar$ consistently does better than the baseline. $SigProc$, does better than the baseline in all but one situation. This situation is shown in Figure 7(b) for $Qp = 31$. The reason is that the bag being tracked is occluded by a person with a similar color distribution. At some values of Qp, part of the object being tracked and part of the occluding object which end up in the same macroblock, may be coded uniformly throwing the tracker of course. This presents a fundamental challenge in block based codecs since the block boundaries are not aligned with the computer vision algorithm boundaries.

Figures 5 and 6 show situations in the tracking sequence where the baseline tracker is thrown off track while $SigProc$ and $VarPar$ hold track. The baseline tracker looses track in Figure 5 as the woman emerges from behind a car whose color distribution has some elements close to her own distribution. A similar situation is shown in Figure 6 as the blue bag emerges from behind a man wearing a blue sweater. The tracker then locks on to a blue tie. The target bounding box is shown in red, while the $ActWin$ boundary is shown in yellow.

Figure 10 shows the tracking accuracy of $VarPar$ as it iterates over values of Qp in $ActWin$. The best value of Qp is picked. It is clear that the tracking doesn't necessarily get worse at higher values of Qp, which is interesting.

4. CONCLUSIONS

Our goal in this paper was to understand how to make a video signal robust to degradations in the encoding process with respect to some information measure extracted by a computer vision algorithm. In this paper, we used the Mean Shift tracker as the computer vision algorithm. We used two methods $SigProc$, signal processing on the input video signal, and $VarPar$, variable parameters of the encoder, to achieve this purpose. In both cases we saw that tracking accuracy improved, albeit at a loss in PSNR. It is now up to the application at hand to see if the tradeoff is justifiable. We on our part, will experiment with a broader class of computer vision algorithms, a broader class of compensation schemes, and a larger variety of datasets.

5. REFERENCES

[1] "Youtube," http://www.youtube.com/.

[2] "Facebook," http://www.facebook.com/.

[3] Jrme Meessen, Christophe Parisot, Xavier Desurmont, and Jean franois Delaigle, "Scene analysis for reducing motion jpeg 2000 video surveillance delivery bandwidth and complexity," in *In IEEE International Conference on Image Processing (ICIP 05*, 2005, pp. 577–580.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 5, pp. 564–577, 2003.

[5] Gary R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," *Applications of Computer Vision, IEEE Workshop on*, vol. 0, pp. 214, 1998.

[6] Gary Bradski and Adrian Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, 2008.

[7] Chris Stauffer, W. Eric, and Eric, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 747–757, 2000.

[8] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: a systematic survey," *Image Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 294–307, March 2005.