AN IMPROVED VIDEO ENCODER WITH IN-THE-LOOP DE-NOISING FILTER FOR IMPULSE NOISE REDUCTION

Snehashis Roy, Somnath Sengupta

Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
India 721302

ABSTRACT

It is seen that the performance of a video codec drastically degrades for noisy sources. In this paper, we propose a novel and easy-to-implement de-noising filter, integrated within the motion estimation and motion compensation blocks of an H.264 video encoder, to reduce impulse noise from video. Performance of our proposed filter is compared with a spatial domain two-pass adaptive median filter([1]) that uses spatial information of noise. Results are obtained for several video sequences, contaminated with different levels of additive random noise, thus establishing the effectiveness of the proposed technique.

1. INTRODUCTION

The signal-to-noise ratio of video sources can worsen due to various factors, such as additive noise introduced by sensors, poor lighting conditions, camera shutter speed etc. The analog video, when digitally encoded, reproduces the high frequency noise as sharp impulses in digital video. Efficient entropy coding of such impulse noise tainted video is not possible due to widely uncorrelated nature of the noise.

The impulse noise is assumed to follow uniform spatial distribution. The magnitude is usually modeled as Gaussian distribution with parameters \( N(0, \sigma^2) \). Previous approaches to source noise reduction uses pre-filtering in spatial domain. The most widely used is the median filter. A two-pass adaptive median filtering algorithm is presented in [1] and [2]. Speckle reduction using median filter bank approach is discussed in [3]. Median filter with multiresolution analysis has been discussed in [4]. To estimate noise for efficient pre-filtering, motion estimation based approach has been discussed in [5]. It primarily deals with estimation of noise so that the noise can be reduced by efficient pre-filters.

In conventional impulse noise reduction techniques, a spatial domain adaptive noise estimator is devised. It selectively replaces a pixel based on some threshold. In this paper, instead of any pre-filtering, we integrate the de-noising filter within an H.264 video encoder loop itself. The in-built Motion Estimation and Motion Compensation blocks of the video codec are modified so that they generate Motion Vectors as well as act as impulse noise filter. The proposed approach also keeps the H.264 coding architecture unaltered. The motivation behind such filtering is that if noise is imperceptible, the degradation of the quality should also be imperceptible.

2. THE IN-LOOP DE-NOISING FILTER

2.1. In-Loop filter Architecture

The proposed modified H.264 encoder with in-Loop de-noising filter is shown in Fig.1.

2.2. Problem Formulation

Consider three temporally ordered consecutive frames, \( F_1, F_2 \) and \( F_3 \). \( F_3 \) is to be encoded based on the motion vectors obtained from both the previous frames. Consider all of them to be contaminated with additive impulse noise, with parameters \( N(\mu, \sigma^2) \). The distribution of noise magnitude is modeled as Gaussian and the spatial probability distribution is assumed to be uniform. Let \( p \) be the probability that a pixel will be contaminated with noise. If \((i,j)^{th} \) pixel intensity is \( f_{ij} \) and the added noise is \( n_{ij} \), the noise-tainted pixel intensity is : \( g_{ij} = f_{ij} + n_{ij} \). Using Block Matching Algorithm (BMA), best match for a block \( A_3 \in F_3 \) is found in \( A_2 \in F_2 \) and best
match of \( A_2 \in F_2 \) in \( A_1 \in F_1 \). The determination of best matched block is included in Motion Estimation (ME) block. Two error matrices \( E_1 \) and \( E_2 \) are calculated from \( A_3, A_2 \) and \( A_1 \) as: \( E_{2i,j} = A_{3i,j} - A_{2i,j} \) and \( E_{1i,j} = A_{2i,j} - A_{1i,j} \).

Using a fixed threshold on \( E_1 \) and \( E_2 \), de-noised version of \( A_3 \) is generated, \( \forall A_3 \in F_3 \). This is included inside the Motion Compensator block, which takes the motion vector information from Motion Estimation block. It is assumed that due to random nature of noise, it is highly unlikely that the same pixel will be corrupt in all the three consecutive frames.

### 2.3. Algorithm

We now describe the algorithm of the in-loop filter.

**MODIFIED\_MOTION\_ESTIMATE(F_1, F_2, F_3)\**

1. while(\( A_3 \neq \emptyset \)) {
2. \( MV_2 = Get\_Motion\_Vector(A_3 , F_3 , F_2) \);
3. \( A_2 = Find\_Block(F_2, MV_2) \);
4. \( MV_1 = Get\_Motion\_Vector(A_2 , F_2 , F_1) \);
5. \( A_1 = Find\_Block(F_1, MV_1) \);

**MODIFIED\_MOTION\_COMPENSATE(F_1, F_2, F_3, MV_1, MV_2)\**

1. // \( A_1 \in F_1, A_2 \in F_2, A_3 \in F_3 \)
2. while(\( A_3 \neq \emptyset \)) {
3. \( E2 = A_3 - A_2 \); \( E1 = A_2 - A_1 \);
4. if(\( E_{2i,j} \geq T \))
5. \( B_{2i,j} = A_{2i,j} \);
6. elseif(\( E_{2i,j} \leq -T \))
7. \( B_{2i,j} = A_{3i,j} \);
8. else
9. if(\( -T \leq E_{2i,j} < T \))
10. \( B_{2i,j} = \frac{1}{2}(A_{3i,j} + A_{2i,j}) \);
11. endif
12. endif
13. \( E_1 = A_{1i,j} \);
14. elseif(\( E_{1i,j} \leq -T \))
15. \( B_{1i,j} = A_{2i,j} \);
16. else
17. if(\( -T < E_{1i,j} < T \))
18. \( B_{1i,j} = \frac{1}{2}(A_{2i,j} + A_{1i,j}) \);
19. endif
20. endif
21. \( B_3 = \) the positive noise filtered version of \( A_3 \)
22. if(\( E_{3i,j} \geq T \))
23. \( B_{3i,j} = A_{3i,j} \);
24. elseif(\( E_{3i,j} \leq -T \))
25. \( B_{3i,j} = A_{2i,j} \);
26. else
27. if(\( -T < E_{3i,j} < T \))
28. \( B_{3i,j} = \frac{1}{2}(B_{1i,j} + B_{2i,j}) \);
29. endif
30. endif
31. // thresholding the negative noise elements \( n_{ij} \) < 0 in \( A_2 \),
32. //\( B_3 \) is the final filtered version of \( A_3 \)
33. } // end of thresholding, reconstruction ends

/* SubRoutines */

- \( Get\_Motion\_Vector(A_3, F_3, F_2) \): Takes two frames \( F_3 \) and \( F_2 \) and a block \( A_3 \in F_3 \), returns the motion vector joining \( A_3 \in F_3 \) to \( A_2 \in F_2 \), where \( A_2 \) is the best matched block for \( A_3 \). It is similar to the standard H.264 Motion Estimation block.

- \( Find\_Block(F_1, MV_1) \): Takes a frame \( F_1 \) and a motion vector \( MV_1 \) and returns the block \( A_1 \in F_1 \), which is determined as the end-point of the vector \( MV_1 \). It is as per with the standard Motion Compensator block of H.264 encoder.

### 2.4. Justification of the Algorithm

A pixel having intensity \( f_{ij} \) becomes \( g_{ij} = f_{ij} + n \) after being corrupt with additive noise. Let \( g_{ij} \in A_1 \) has the best match in \( g_{ij} \in A_2 \). They are found by block matching algorithm in Motion Estimation block. If original value of the pixel is \( \alpha \), then \( g_{ij} \) or \( \hat{g}_{ij} \) can have values \( \alpha + n, \alpha, \alpha - n \), assuming \( n > 0 \). Using the spatial randomness property of noise, we assume that \( g_{ij} \) and \( \hat{g}_{ij} \) have same values with very small probability. Define \( E_{ij} = g_{ij} - \hat{g}_{ij} \). If \( E_{ij} > 0 \), then there are two possibilities, either \( g_{ij} = \alpha + n \) and \( \hat{g}_{ij} = \alpha \) or \( g_{ij} = \alpha \) and \( \hat{g}_{ij} = \alpha - n \). So we choose \( \hat{g}_{ij} \) as new filtered pixel if \( E_{ij} > T \), where \( T \) is a fixed pre-defined threshold. Here all positive noise components, as in \( g_{ij} = \alpha + n \), are reduced. Thus two similar blocks, where pixels are with only negative noise components, are generated. Then they are compared in a similar fashion but with threshold \(-T \). This is psycho-visually decided such that two pixels with intensity \( f_{ij} \) and \( f_{ij} + T \) will be visually more or less indistinguishable. This completes the motivation behind the algorithm.

As such the frame ordering is unimportant as long as the motion estimation reproduces the current frame faithfully with the help of reference frames. So for filtering of \( F_3, F_2 \) and \( F_1 \) need not be immediate past frames. In H.264, future frame reference is allowed. This is also applicable here, as temporally ordered consecutive frames \( F_2, F_3, F_1 \) will produce same result. This is in accordance with the assumption that noise is spatially uncorrelated in temporally consecutive frames.

### 2.5. Algorithm Complexity

The algorithm does not change the complexity of the H.264 codec. As the filter is realized by a selective replacement algorithm with fixed threshold, the complexity of the algorithm is same as the complexity of Motion Estimation block or that of Block Matching algorithm.

### 3. RESULTS

Fig.2 and Fig.3 show the visual improvement of a noisy frame after applying the Adaptive(2-pass) Median filter given in [1]
and the proposed In-Loop filter. The frames are shown after decoding.

(a): Original frame #38 (36.21 dB), (b): Noisy, unfiltered #38 (25.22 dB)

(c): Noisy, AM filtered #38 (28.53 dB), (d): Noisy, IL filtered #38 (32.22 dB)

Fig. 2: Container sequence, Noise added - \( N(0, 45^2) \), \( p = 0.2 \),
PSNR value of each frame is mentioned within braces

(a): Original frame #45 (37.24 dB), (b): Noisy, unfiltered #45 (25.83 dB)

(c): Noisy, AM filtered #45 (26.50 dB), (d): Noisy, IL filtered #45 (33.69 dB)

Fig. 3: Salesman sequence, Noise added - \( N(0, 30^2) \), \( p = 0.15 \),
PSNR value of each frame is mentioned within braces

Table 1. average PSNR values (in dB) and bitrate (kbps) at 30fps, over 100 frames

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Coded</th>
<th>Noisy</th>
<th>Adaptive Median Filtered</th>
<th>In-Loop Filtered</th>
<th>Bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>36.22</td>
<td>23.76</td>
<td>25.28</td>
<td>31.16</td>
<td>170.85</td>
</tr>
<tr>
<td>Container</td>
<td>36.81</td>
<td>22.14</td>
<td>25.65</td>
<td>32.55</td>
<td>135.30</td>
</tr>
<tr>
<td>Hall</td>
<td>35.88</td>
<td>22.98</td>
<td>25.68</td>
<td>31.27</td>
<td>153.93</td>
</tr>
<tr>
<td>Salesman</td>
<td>35.48</td>
<td>25.55</td>
<td>26.41</td>
<td>33.43</td>
<td>184.46</td>
</tr>
<tr>
<td>News</td>
<td>32.79</td>
<td>23.13</td>
<td>24.74</td>
<td>29.87</td>
<td>218.51</td>
</tr>
<tr>
<td>Mother</td>
<td>35.54</td>
<td>23.96</td>
<td>25.41</td>
<td>30.93</td>
<td>66.67</td>
</tr>
</tbody>
</table>

As seen from the graphs, PSNR values improve significantly, from 28dB (Adaptive Median Filtered) to 34dB (In-Loop filtered) for a noisy video of PSNR 25dB (Container Sequence), at similar bitrate. The improvement by adaptive median filter is 3dB whereas the improvement by in-loop filter is 9dB for the same sequence. Also the PSNR values of in-loop filtered video is much closer to decoded version of original video.
3.1. Comparative Studies

The proposed filter has several advantages over the adaptive filters described in [1] - [4]. Primary advantage is that it is hardware efficient. Other than the components of H.264 encoder, it only requires an adder and a comparator as additional hardware. As no adaptive processor is needed, critical path problem is easily avoided.

The most important feature of this filter is that it does not distort a frame if the frame is devoid of any noticeable noise. This has been realized by suitable choice of threshold $T$. Any spatially adaptive median filter degrades the video to some extent, but according to the algorithm, in-loop filter does not replace any pixel if noise is imperceptible. Fig.6 shows an uncontaminated decoded frame after passing through the adaptive filter and in-loop filter.

![Fig.6](image)

(a): Original frame#65(33.82dB), (b): AM filtered#65(27.66dB), (c): IL filtered#65(32.21dB)

Fig.6 : News sequence, Noise added - $N(0, 0), p = 0.0$, PSNR value of each frame is mentioned within braces

This property has been illustrated by PSNR values of the same video in Fig.7.

![Fig.7](image)

Fig.7 : PSNR of H.264 decoded sequences

4. CONCLUSION

We have described a fully adaptive, hardware efficient denoising algorithm applied on noisy video sequences. The noise-reduction is performed in-loop with the motion estimation and compensation blocks of H.264 video codec. It is seen that the performance improves over that of pre-filtered noisy video followed by video codec.

5. REFERENCES


