Scalable video compression for embedded systems in surveillance

Citation for published version (APA):
Scalable Video Compression for Embedded Systems in Surveillance

Marijn J.H. Loomans, Cornelis J. Koeleman, Peter H.N. de With

1 VDG Security BV  
2 Eindhoven Univ. of Techn. / LogicaCMG
2718 TA Zoetermeer, NL  
5600 MB Eindhoven, NL
marijn@vdg-security.com

Abstract

In this paper, we explore coding architectures for camera surveillance applications. For this purpose we propose a Scalable Video Codec (SVC) based on wavelet transformation. The emphasis in the research is on trading-off the coder complexity, e.g. motion-compensation stages, compression efficiency and end-to-end delay of the video chain. Within our architecture of choice, t+2D, many complexity-performance trade-offs can be made by configuring the lifting-based motion-compensated temporal filtering (MCTF). We discuss various configurations and elaborate on the required resources for implementation. We have found an SVC (t+2D) that has a scalable complexity and performance, which enables embedded system applications. Even though the complexity of our SVC is many times lower, performance at full resolution is close to H.264 SVC (within 0.2-3 dB depending on the input sequence) and at lower resolutions sufficient for our video surveillance application.

1 Introduction

The amount of cameras in a surveillance system is growing continuously, requiring efficient coding of the generated video streams. Since the encoding will be inside the camera, resources will be constrained, imposing strong restrictions on encoder complexity. Another system aspect is the desire for scalable video representations, as surveillance system users tend to view the recordings on different devices, such as full size monitors, overview monitors and hand-held devices. Moreover, scalability can be interesting for efficient long term storage of the video information.

In this paper, we explore coding architectures for camera surveillance applications. For this purpose we propose a Scalable Video Codec (SVC) based on wavelet transformation. Wavelet coding has been extensively explored in the academic domain and it is used in e.g. the JPEG 2000 standard. For video applications, it is interesting to combine wavelet coding with forms of motion compensation. Motion Compensated Temporal Filtering (MCTF) was first proposed by Ohm [1] and later refined by Choi and Woods [2]. Improvements were made by Secker and Taubman [3] and Pesquet-Popescu and Bottreau [4] by utilizing lifting, proposed by Sweldens [5], in the temporal wavelet decomposition. Later, a more flexible framework named Unconstrained Motion Compensated Temporal Filtering (UMCTF) was proposed by Van der Schaar [6]. Some papers also look at implementation aspects of 3D subband coders. Turaga and Van der Schaar [7] proposed a method to reduce the motion-estimation complexity. Pau, Viéron and Pesquet-Popescu [8] looked at encoder, decoder and end-to-end delay in the UMCTF framework. However, none of these papers look at a complete scope of trade-offs that can be made between coder complexity and performance. Therefore, in
this paper extensive focus will be on the trade-offs between implementation cost and performance of our SVC. We have found an SVC (t+2D) that has a scalable complexity and performance, which at full resolution, performs close to H.264 SVC (within 0.2-3 dB depending on the input sequence).

This paper is organized as follows. Section 2 elaborates on a selection of temporal configurations that are possible in the t+2D coding architecture. We describe the experimental setup in Section 3 and present our results in Section 4. The paper concludes in Section 5.

2 Coding and Temporal Configurations

We have explored three possible architectures: t+2D, 2D+t and a pyramid-based coding system. Our SVC is based on the t+2D architecture, shown in Figure 2. Systems in the class t+2D first perform temporal wavelet decomposition, and thereafter a 2D spatial decomposition. Systems labeled 2D+t first perform the spatial decomposition, thereafter the temporal decomposition. Pyramid-based systems first down-sample the input signal to various resolutions (e.g. 4CIF, CIF and QCIF) and for each of the resolutions perform a t+2D decomposition. Information from lower-resolution layers is then used as a prediction for higher-resolution layers. This inter-layer prediction can consist of intra-prediction, motion-vector prediction and residual prediction. A more detailed comparison is beyond the scope of this paper.

![Figure 1: The t+2D coding architecture.](image)

Within the t+2D architecture many complexity-performance trade-offs can be made. The temporal decomposition is performed using motion-compensated temporal filtering (MCTF), where motion compensation is performed within the lifting steps of the temporal wavelet filter. Different temporal configurations can be explored, depending on which predict and update steps should be included. In literature, various 3D subband coding schemes have been proposed. Two frameworks are most prominent: LIMAT by Secker and Taubman [3], and UMCTF (Unconstrained Motion Compensated Temporal Filtering) by Van der Schaar [6]. We have adopted the latter proposal as a guideline for exploration, as it allows different configurations with different complexities and end-to-end delay. The complexity of the UMCTF depends on the implementation of the prediction and update steps within the temporal lifting. Figure 2 shows four possibilities for the temporal transform, \(P_2 U_2, P_2 U_1, P_2 U_0\) and \(P_1 U_0\).

Let us use the following notation as defined in [8]. Parameter \(x_t\) denotes the input video frame at time \(t\), \(l_t\) denotes the temporal low-pass subband frame and \(h_t\) denotes the temporal high-pass subband frame. As motion-compensated lifting is used, predictions of odd frames \(x_{2t+1}\) are made from even frames \(x_{2t}\) and \(x_{2t+2}\). Two motion vector fields are estimated: forward prediction from \(x_{2t}\) to \(x_{2t+1}\) denoted by \(v_{2t+1}\), and back-
ward prediction from $x_{2t+1}$ to $x_{2t+2}$ denoted by $v_{2t+1}^-$. Assuming $n$ the spatial index in a frame, the motion-compensation operator $C$ is defined by $C(x, v)(n) = x(n - v(n))$.

The four different temporal filtering methods $P_2U_2$, $P_2U_1$, $P_2U_0$ and $P_1U_0$ as shown in Figure 2 are expressed in Equations (1) through (4), by

$$h_t = x_{2t+1} - \frac{1}{2}(C(x_{2t}, v_{2t+1}^+) + C(x_{2t+2}, v_{2t+1}^-)) \land$$

$$l_t = x_{2t} + \frac{1}{4}(C^{-1}(h_{t-1}, v_{2t-1}^-) + C^{-1}(h_t, v_{2t+1}^+)), \quad (1)$$

$$h_t = x_{2t+1} - \frac{1}{2}(C(x_{2t}, v_{2t+1}^+) + C(x_{2t+2}, v_{2t+1}^-)) \land$$

$$l_t = x_{2t} + \frac{1}{2}C^{-1}(h_{t-1}, v_{2t-1}^-), \quad (2)$$

$$h_t = x_{2t+1} - \frac{1}{2}(C(x_{2t}, v_{2t+1}^+) + C(x_{2t+2}, v_{2t+1}^-)) \land$$

$$l_t = x_{2t}, \quad (3)$$

$$h_t = x_{2t+1} - C(x_{2t}, v_{2t+1}^+) \land$$

$$l_t = x_{2t}. \quad (4)$$

We can now build an $N$-level temporal transform with the four configurations of Figure 2. In [8], the transform is parameterized by $(P, Q)$, with $P$ denoting the number of $P_2U_2$ transforms at the finest level(s) and $Q$ denoting the number of $P_2U_1$ transforms at the following level(s). If any, the remaining level(s) use the $P_1U_0$ transform. \(^1\) Five configurations, defined by $(P, Q)$, are shown in the top part of Table 1. We propose another two, both without update steps. First, a configuration with all the predict steps activated $(P_2U_0)$, referred to as $\text{PredOnly}$ or $\text{PO}$. Second, a configuration which uses bidirectional prediction at the two lowest levels $(P_2U_0)$, and single prediction at the two highest levels $(P_1U_0)$ to reduce the end-to-end delay. This configuration will be referred to as $\text{BiDirLowDel}$ or $\text{BDDL}$. Both configurations are shown in Table 1.

Note that for $(P, Q) = (4, 0)$, the transform is equal to the well-known $5/3$ temporal transform and for $(P, Q) = (0, 0)$ it is similar to the Haar transform, but without the update steps. The $\text{BiDirLowDel}$ and $\text{Pred}$ configurations are similar to $(P, Q) = (0, 2)$ and $(P, Q) = (4, 0)$ configurations, respectively, but without the update steps.

\(^1\)The $P_2U_0$ transform is not used in [8].
Table 1: Chosen configurations for a 4-level temporal transform.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>$P_2U_3$ levels</th>
<th>$P_2U_1$ levels</th>
<th>$P_2U_0$ levels</th>
<th>$P_1U_0$ levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(P, Q) = (4, 0)$</td>
<td>4</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>$(P, Q) = (1, 2)$</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>$(P, Q) = (0, 2)$</td>
<td>0</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>$(P, Q) = (0, 1)$</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>$(P, Q) = (0, 0)$</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>

| PredOnly | - | - | 4 | 0 |
| BiDirLowDel | - | - | 2 | 2 |

3 Experimental Setup

Implementation and sequences
Spatial (H+V) wavelet decomposition is performed by filtering the image by the well-known 9/7 Daubechies filter bank. For broadcast TV resolution, a 5-level spatial wavelet transform is used. The GOP-size is 16 frames, with an adaptive 4-level temporal wavelet transform. Motion is estimated to 1/8 accuracy using a block-based motion estimator with a fixed block size of $16 \times 16$ pixels. Rate control is performed by simply reducing the number of bit-levels decoded. To provide finer granularity in this simple rate control scheme, it is possible to drop a part of the bits within a certain bit level. The temporal low-pass frame is encoded with one more bit-level than the other temporal high-pass frames. Entropy coding of the wavelet coefficients is performed with a modified SPIHT encoder. No arithmetic coding stages are used and the cost of coding motion vectors has not been taken into account. The following sequences were used for testing and validation: City (704x576, 25fps), Crew (704x576, 25fps) and CrossingA (704x576, 25fps). The last sequence, CrossingA, is a self-recorded sequence which is more representative for a typical security surveillance video.

Performance Metrics
Motion estimation is one of the most computationally expensive algorithms to implement inside a video compression algorithm. We look at the number of times motion estimation and compensation is performed on a full frame.
Memory usage for embedded systems is a critical parameter, especially the memory bandwidth. The temporal transform requires various frame filtering operations and also the motion compensation is applied in the temporal domain.
End-to-end delay is defined as the time it takes between capturing a frame and viewing it on a display. The temporal transform is the most delay-consuming stage. In our experiments we have ignored the local delay for coding, quantization and transmission.
Frame accessibility is important for a security surveillance system because of the various types of viewing, e.g. normal playback, high speed playback, reverse playback, frame-by-frame access and arbitrary access.
PSNR is used as an objective measure for quality (also subjective evaluation is done). To determine the PSNR for sequences decoded at lower resolutions, reduced-resolution references are obtained using the low-pass filter of the 9/7 wavelet filter kernel.
4 Results and Discussion

Complexity Analysis
The most important performance metrics on computation and memory usage are shown in Table 2 for various temporal configurations. The table shows that the (4, 0) configuration is clearly more expensive than the other proposals and because of the large end-to-end delay it is not suitable for the desired application. The other configurations have comparable complexity parameter values so that a comparison in quality performance is required.

With respect to searching in a recording database, arbitrary frame access is the most critical feature resource usage. For example, when accessing a single frame, it may be required to fetch up to 13 pictures and perform the complete temporal decomposition. This requirement favors the choice for the (0, 0) and BDLD configuration.

Table 2: Quantified performance metrics for different temporal configurations. The last column refers to the average number of frames accesses for an arbitrary search.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,0)</td>
<td>30</td>
<td>60</td>
<td>30</td>
<td>41</td>
<td>46</td>
<td>106</td>
<td>30</td>
<td>23</td>
<td>45</td>
<td>13</td>
</tr>
<tr>
<td>(1,2)</td>
<td>29</td>
<td>51</td>
<td>29</td>
<td>17</td>
<td>45</td>
<td>96</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>8.5</td>
</tr>
<tr>
<td>(0,2)</td>
<td>27</td>
<td>39</td>
<td>27</td>
<td>11</td>
<td>43</td>
<td>86</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>(0,1)</td>
<td>23</td>
<td>31</td>
<td>23</td>
<td>8</td>
<td>39</td>
<td>69</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4.5</td>
</tr>
<tr>
<td>(0,0)</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>4</td>
<td>31</td>
<td>46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>PO</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>32</td>
<td>46</td>
<td>76</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>BDLD</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>8</td>
<td>43</td>
<td>70</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Temporal configurations
Figure 3 shows the rate distortion curves for the City and Crew. A few observations can be made from these curves. First, by carefully choosing the temporal configuration, increasing complexity can improve coder performance, as seen by a higher PSNR at the same bit rate. It is interesting to note that for most cases, the update steps do not improve performance, but degrade it, which can be observed by comparing the BiDirLowDel and (0, 2) curves. A possible explanation is that due to inaccuracies in the motion estimation, the update steps introduce artifacts in the low-pass bands. This emphasizes that selecting the right temporal configuration is not a trivial task.

Reduced Resolutions
Due to the choice of the t+2D architecture, spatial aliasing will be evident in sequences decoded at lower resolutions. This reduces the PSNR significantly, however, visually this is perceived as a slightly blurring effect.

An interesting case occurs at the City sequence. Due to the high amount of fine structures, aliasing can be clearly observed at one of the buildings in the background. Figure 5 shows frame 64 of the City sequence. The original sequence is down-scaled to 352×288 resolution and depicted in (a), with the building enlarged in (b). The compressed sequence decoded at 352×288 resolution is depicted in (d), with the building enlarged in (c). In the compressed image, the coding blurs certain parts of the aliasing pattern present in the original sequence. As a result, the PSNR drops significantly as can be seen in the rate-distortion curve in Figure 4b. For the 176×144 resolution, most of the fine structure in this building has disappeared in the process of down-sampling.
and the same phenomenon is less intrusive, which is also visible in the slightly higher rate distortion curve for this resolution. The CrossingA sequence does not contain these structures and therefore shows more reasonable rate-distortion curves as shown in Figure 4a.

Figure 4: Rate distortion curves of (a) CrossingA and (b) City decoded at full and reduced resolutions.

**Comparison between H.264 SVC and proposed SVC**

We compared the H.264 SVC CE2.1 codec with our proposed codec for the City and the Crew sequences. As motion vector coding is not yet fully implemented in our proposed SVC, we omit the cost of coding the motion vectors. To compensate for this difference, we reduced the rate by approximately 10% compared to H.264. PSNR curves and resulting pictures are shown in Figure 6 and Figure 7 for the City and Crew sequences, respectively. For the City sequence, H.264 outperforms the proposed SVC, most likely due to the highly advanced motion-estimation model. This is visually confirmed as the H.264 sequence shows more details in background buildings and fine structures. For the Crew sequence, our proposed SVC shows slightly higher PSNR than H.264. Visually, H.264 looks less noisy than the proposed SVC, but at the cost of
Figure 5: Spatial aliasing in the *City* sequence decoded at 352×288: (a) original frame, (b) detail of original frame, (c) detail of compressed frame and (d) compressed frame.

more blurring. Our SVC shows more details at the cost of being slightly more noisy, however, this is preferred for our surveillance application.

5 Conclusion

In this paper, we have shown complexity/performance trade-offs for our SVC (t+2D) using various temporal configurations, where our preference goes out to the BiDirLowDel configuration. This enables implementation of our advanced scalable video codec on a highly resource-constrained embedded system, such as a network camera. The coding performance at full resolution is close to H.264 SVC (within 0.2-3 dB depending on the input sequence) and at lower resolutions sufficient for our video surveillance application, but our SVC has a much lower complexity. The most dominant reductions are found in the non-hierarchical motion model, the straightforward t+2D architecture and the simple entropy coding scheme.

References

Figure 6: Comparison of H.264 SVC and our proposed SVC for the City sequence: (a) PSNR per frame, (b) H.264 SVC frame 36 and (c) proposed SVC frame 36.

Figure 7: Comparison of H.264 SVC and our proposed SVC for the Crew sequence: (a) PSNR per frame, (b) H.264 SVC frame 36 and (c) proposed SVC frame 36.