Abstract: In this paper, we propose a partial decoding method with limited memory usage for high-speed thumbnail extraction. The proposed method performs a partial inverse transform and a partial intra prediction in order to reconstruct pixels for intra prediction and thumbnails. Thereafter, the reconstructed pixels at the bottom and right line of the block are stored in the line buffer and the thumbnail buffer without being stored in the decoded picture buffer with full resolution. H.264/AVC, HEVC and VP9 video codecs have different coding structures, prediction and transforms; however, the proposed algorithm can be applied to the corresponding codecs in the same manner. In order to evaluate the performance of the proposed method, we implemented the proposed algorithm for H.264/AVC, HEVC and VP9. We found that the thumbnail extraction time of the proposed method decreased by 66% in H.264/AVC, 52% in HEVC and 48% in VP9 as compared to the full decoding method.

Keywords: thumbnail; partial decoding; real-time processing; memory reduction; H.264/AVC; HEVC; VP9

1. Introduction

With the recent development of video capture, displays and processing capability, there is a growing demand for high-definition, high-quality video services in the market. Along with the market requirements, the ISO/IEC moving picture expert group (MPEG) and video coding expert group (VCEG) have developed subsequent video codec standards such as MPEG-2, H.264/AVC and high-efficiency video coding (HEVC) [1–3]. In addition, private entity codecs such as VP8 and VP9 [4,5], among others, have been developed. As the codecs have evolved, their coding performance has increased by a bit-saving factor of two for the same visual quality. These advances have been made possible by adding a number of new tools with high computational complexity. Many tools have been developed for and added to intra frames. However, to reduce the amount of spatial redundancy of the frame, coding tools are interdependent, making it difficult to accelerate the video codecs in real time with a minimal computational load.

In today’s world, we are flooded with video content from multiple media services through broadcasting, over-the-top media service, the internet, and so on. Easy selection of these contents is necessary, and thumbnail display is used as a part of user interfaces to allow users to visually select the content they wish to engage with. However, video resolution is also increasing rapidly up to 8K or higher, resulting in a significant increase in the hardware requirements for fully decoding video content. Several attempts have been made, such as parallel decoding and decoder implementation using single Instruction multiple data (SIMD) instructions, to improve the decoding speed [6–9]. However, it is still almost impossible to fully decode multiple 4K or 8K videos simultaneously in limited hardwares to enable thumbnail display. In addition, we need to reduce the required amount of memory used for thumbnail processing on embedded systems due to their memory limitations. Some attempts have been made to extract thumbnail images in the frequency...
domain [10,11] based on Chen’s transform domain intra prediction method [12]. However, these methods require additional look-up tables, which increases the amount of memory use and is very hard to apply to other codecs. As such, a partial decoding method has been proposed [13]. This method restores only the right and bottom boundary pixels in 4 × 4 units, which is the minimum transform unit (TU) size of HEVC. This method can be easily applied to other codecs; however, it always operates in 4 × 4 units regardless of the block size. Therefore, unnecessary pixels that are not used for the thumbnail output or reference pixels for intra prediction are restored. Figure 1 shows the pixels used for a reference pixel or thumbnail output (gray) and the unnecessarily reconstructed pixels (yellow) of the method [12]. The number of unnecessarily restored pixels for the N × N TU block are as follows:

\[
\frac{3N^2 - 12N}{8}
\]  

(1)

Figure 1. The restored pixels for the intra prediction or thumbnail output (gray) and unnecessarily restored pixels (yellow) in a 16 × 16 block.

In this paper, we present a fast thumbnail decoding method using a small amount of memory and partial decoding according to the prediction block size for intra frames of H.264/AVC, HEVC and VP9 with minimal visual quality loss and without any error propagation. The proposed partial decoding method restores the pixels constituting the thumbnail and the reference pixels used in the intra prediction by replacing the full inverse transform and intra prediction with the partial inverse transform and partial intra prediction. The computational load and memory usage are greatly reduced by omitting both the reconstruction process and the storage process for the other pixels. HEVC employs large transformations with dimensions such as 32 × 32; thus, we reconstructed several pixels inside the block to preserve the visual quality of the reconstructed thumbnails. The memory structure for the proposed partial decoding method uses the minimal thumbnail buffer and the reference line buffer rather than the decoded picture buffer (DPB). Memory is not allocated for pixels whose restoration is omitted, and reference pixels that are no longer required are removed by storing the restored reference pixels of the next block, thereby reducing the memory allocation required. In addition, a down-sampling process using a thumbnail buffer to store thumbnail pixels for output is not performed, thereby reducing computational complexity, memory usage and memory access. Video codecs of H.264/AVC, HEVC and VP9 have different coding structures and transforms. However, the proposed algorithm can be applied in the same manner for these codecs. In order to evaluate the performance of the proposed method, we implemented the proposed algorithm for H.264/AVC, HEVC and VP9. We found that the thumbnail extraction time of the proposed method decreased by 66% in H.264/AVC, 52% in HEVC and 48% in VP9 compared to the full decoding method.
This paper is organized as follows: Section 2 explains the proposed partial decoding method for fast thumbnail extraction and an efficient memory structure. Section 3 shows the experimental results of the proposed algorithm and the existing implementation of open software in terms of running times and visual quality. Section 4 concludes the paper.

2. Proposed Partial Decoding of H.264/AVC, HEVC and VP9 for Thumbnail Extraction

A proposed thumbnail extraction method includes a video decoding process and a down-sampling process. The video decoding process consists of entropy decoding, inverse transformation, intra prediction and in-loop filtering for intra frames. During the thumbnail extraction, entropy decoding, inverse transformation and intra prediction are performed to reconstruct the full image frame. Then, down-sampling is performed with the thumbnail size. However, the decoding and down-sampling processes of thumbnail extraction have high computational complexity and memory usage. In this paper, we propose a partial decoding method according to prediction block size and a memory structure for high-speed thumbnail extraction and compact memory usage with minimal visual degradation.

The proposed thumbnail extraction method replaces the inverse transform and intra prediction in the existing video decoding process with a partial inverse transform and partial intra prediction. The proposed algorithm uses a low-capacity thumbnail buffer optimized for the reconstructed pixels in the partial decoding process, along with two-line buffers. Figure 2 shows the block diagram of the proposed method for fast thumbnail extraction. This diagram shows a thumbnail extraction method in which the partial inverse transform and partial intra prediction are replaced with the original inverse transform and intra prediction, and the restored pixels are stored in the line buffer and thumbnail buffer to omit the down-sampling process.

In this section, we describe a partial decoding method that reduces the computational complexity of the inverse transform and intra prediction process during the decoding process, with minimal visual degradation. The proposed partial decoding method restores only the pixels necessary for thumbnail extraction in order to reduce the complexity of the inverse transform and intra prediction. The pixels necessary for the thumbnail extraction are not only the pixels to be output to the thumbnail but also those necessary to avoid error propagation in the intra-picture prediction. The pixels required for intra prediction in the subsequent blocks are the right-boundary and lower-boundary pixels required for intra prediction in all the pixels of a transform block. If these pixels have errors or are not restored, the errors propagate and accumulate for the entire image, greatly reducing the visual quality of the reconstructed image.

For videos with a resolution of less than Full HD, the decoding speed is sufficiently fast even in a limited hardware environment; thus, we targeted videos with a resolution of 4K or higher. As demonstrated by the experiments, the subjective quality was good enough to display the thumbnails even if the thumbnails were generated at 1/64 size of the original resolution from 4K or higher UHD videos. This paper is based on the 1/64 size thumbnail extraction method, and in order to extract the thumbnails at the 1/2^n size of the original resolution according to the user’s preference, it can be applied similarly by restoring the right and bottom boundary pixels and one pixel for each $2^\frac{n}{2} \times 2^\frac{n}{2}$ blocks.
In order to extract a thumbnail with a size corresponding to $1/64$ of the original image resolution, one pixel is required for each $8 \times 8$ block. Therefore, all the pixels output to the thumbnail correspond to one pixel at the bottom right of the $8 \times 8$ unit block. However, HEVC and others employ larger transform sizes; thus, we were required to reconstruct several pixels inside the transform blocks for better visual quality. H.264/AVC, HEVC and VP9 video compression standards have various prediction and transform sizes and shapes (from $4 \times 4$ to $32 \times 32$, and square and non-square blocks). When the proposed partial decoding method was applied to the H.264/AVC, HEVC and VP9 video compression standards, the restored pixel positions within the block were as shown in Figure 3.

![Figure 3. Pixels to be reconstructed for the proposed thumbnail extraction for various block shapes.](image)

Table 1 lists the numbers of reconstructed pixels in the block according to the transform block size when the proposed method is applied. The reference pixels in the table are pixels to be reconstructed and stored for intra prediction of the next block, and the pixels inside the block are the pixels that are reconstructed to be output as a thumbnail. The reconstruction pixel ratio represents the ratio of the reconstructed pixels of the proposed method to the total number of pixels in each block. According to the table, the reconstruction ratio of the proposed algorithm is reduced from 44% to as low as 7%, depending on the block sizes. Because the reconstruction ratio is reduced in proportion to the block size, lower computational complexity of the thumbnail extraction method is necessary when larger-sized blocks are included in an image. In addition, the memory usage can be reduced for large blocks. We are required to develop partial inverse transformations and partial intra prediction, which are described in the subsections along with the minimal memory structure. The proposed partial decoding for chroma samples can be easily derived in a same way, even for a 4:2:0 format.
The reconstruction pixel ratio represents the ratio of the reconstructed pixels of the block sizes. As shown in the table, blocks that are larger and have greater horizontal lengths can be accelerated more.

### Table 1. The number of pixels to be reconstructed for the block sizes.

| Block Size/Shape | # Pixels | # Pixels to Be Reconstructed | Total # of Pixels to Be Reconstructed | Reconstruction Pixel Ratios |
|------------------|----------|------------------------------|---------------------------------------|-----------------------------|
|                  |          | Reference Pixels Inside Block |                                      |                             |
| 4 × 4            | 16       | 7                            | 7                                     | 44%                         |
| 4 × 8 (8 × 4)    | 32       | 11                           | 11                                    | 34%                         |
| 8 × 8            | 64       | 15                           | 15                                    | 23%                         |
| 8 × 16 (16 × 8)  | 128      | 23                           | 23                                    | 18%                         |
| 16 × 16          | 256      | 31                           | 32                                    | 13%                         |
| 16 × 32 (32 × 16)| 512      | 47                           | 50                                    | 10%                         |
| 32 × 32          | 1024     | 63                           | 72                                    | 7%                          |

### 2.1. Partial Transformations

The inverse transformation process is to transform frequency domain transform coefficients, obtained through the entropy decoding process, into magnitudes in the pixel domain by performing a 2D inverse discrete cosine transform (IDCT). In order to reduce computational complexity, the inverse transformation of H.264/AVC, HEVC and VP9 decoders consists of a butterfly structure in which 2D IDCTs are divided into two vertical and horizontal 1D IDCT operations, and each 1D IDCT is added and multiplied [14,15].

The proposed transform is to inversely transform the lowermost and rightmost pixels according to reference pixels, in order to avoid error propagation. In addition, if a transform block is larger than 8 × 8 pixels, one pixel is recovered per 8 × 8 sub-block inside the larger block to avoid interpolation in the thumbnail. We employ the two-stage inverse transformation based on the separable characteristics. To perform the horizontal 1D transformation, the vertical 1D IDCT should be fully performed. However, the horizontal 1D transform can be partly performed for the reference pixels and one pixel per internal 8 × 8 sub-block. As shown in Figure 4, the 16 × 16 block is inversely transformed for the yellow pixels of the second stage.

![Figure 4. Two-stage partial reconstruction for 16 × 16 inverse transformation.](image-url)

The partial horizontal 1D-IDCT is performed by removing some part of the butterfly structure. Depending on the transformation sizes, the 8th, 16th, 24th and 32nd pixels should be reconstructed. Figure 5 shows the operations required to restore the last 16th pixel in the original 16-point 1D-IDCT. The computation amount required for the last pixel reconstruction of the horizontal 16-point 1D-IDCT is 15 additions and 24 multiplications. The popular butterfly structure of 1D-IDCT requires 64 additions and 72 multiplications. Table 2 shows the numbers of additions and multiplications for the proposed partial reconstruction depending on the transformation block sizes. As shown in the table, blocks that are larger and have greater horizontal lengths can be accelerated more.
Figure 5. Partial reconstruction flow of the butterfly structure for a 16-point 1D inverse discrete cosine transform (ID-IDCT).

Table 2. The numbers of additions (Add) and multiplications (Mul) of the full and partial inverse transformations.

| Transform Size | Full 2D-IDCT | Partial 2D-IDCT | Reduction Ratio |
|----------------|--------------|-----------------|-----------------|
| 4 × 4          | 64           | 52              | 19% 23%         |
| 4 × 8          | 224          | 105             | 53% 44%         |
| 8 × 4          | 224          | 212             | 5% 7%           |
| 8 × 8          | 384          | 265             | 31% 26%         |
| 8 × 16         | 1216         | 530             | 56% 50%         |
| 16 × 8         | 1216         | 1097            | 10% 7%          |
| 16 × 16        | 2048         | 1362            | 33% 29%         |
| 8 × 32         | 5184         | 2412            | 53% 54%         |
| 32 × 16        | 5184         | 4498            | 13% 9%          |
| 32 × 32        | 8320         | 5548            | 33% 32%         |

2.2. Partial Intra Prediction

The intra prediction process was designed to improve the compression efficiency by eliminating redundancy among adjacent pixels in an image. In the H.264, HEVC and VP9 decoders, reference pixels of neighbouring blocks are filtered and the filtered pixel values are then used for predicted signals. The predicted and residual signals from the inverse transformation are added.

For thumbnail extraction, partial prediction can be performed from the filtered reference samples. In a manner similar to the partial inverse transformation, the bottom and rightmost pixel lines as well as one pixel per inner 8 × 8 sub-block are predicted. As a result, it is possible to omit the memory copy operations as much as the number of unnecessary pixels for thumbnail. Figure 6 shows the necessary pixels to be predicted and reconstructed for the thumbnail extraction process for a 16 × 16 block.
In real-time thumbnail extraction, it is important to reduce memory access and requirements. In this section, the proposed partial decoding memory structure is described. In the proposed partial decoding method, a very small number of pixels are restored over the original number of image pixels. Furthermore, some part of the restored pixels is used for the thumbnail image. Therefore, the method of storing the reconstructed pixels in the original memory structure for partial decoding is inefficient because it leaves a large amount of memory unused. The proposed memory structure does not allocate memory for pixels whose restoration operations are omitted. Because reference pixels are not re-used, a new reference pixel line can be overwritten. For the proposed thumbnail extraction, one thumbnail buffer and two reference line buffers are employed rather than a full reconstruction frame buffer. The thumbnail buffer resolution is 1/64 of the original one. The reference line buffer is composed of the left and top line buffers. The left line buffer has the maximum block height, and the top line buffer has a pixel value that corresponds to the width of the original image.

After the partial intra prediction and the partial inverse transformation are performed, the predicted and residual signals are summed up, thereby restoring the thumbnail and reference pixels. Among the restored pixels, the thumbnail pixels are to be included in the output thumbnail image, and one pixel per inner $8 \times 8$ sub-block is extracted and stored in the thumbnail buffer. The reference pixels are referred to as an input of the intra prediction, and they are divided into two reference line buffers and stored. The restored rightmost pixels of the block are stored in the left reference line buffer, and the lowest-order pixels are stored in the upper reference line buffer. The reference line buffers are used for consecutive block reconstructions.

Figure 7 shows an example of the proposed memory structure when extracting thumbnails with a resolution of $3840 \times 2160$. The thumbnail buffer has a resolution of $480 \times 270$ (129,600 pixels). The upper and left reference line buffers have 3840 and 64 pixels, respectively; therefore, the required memory space for the restored pixel is reduced by 98%, from 8,294,400 to 133,504.
Figure 7. The proposed memory structure for a 3840 × 2160 input bitstream.

3. Experimental Results and Discussion

In order to evaluate the performance of the fast thumbnail extraction method proposed in this paper, the proposed algorithm for H.264/AVC, HEVC and VP9 was implemented on FFmpeg version 4.2.2 [16]. We also used ffmpegthumbnailer [17] to fairly evaluate the performance of thumbnail extraction with the FFmpeg decoder. The open software ffmpegthumbnailer drives the FFmpeg decoders and down-sampler for thumbnail extraction, as shown in Figure 8. Thus, the proposed algorithm also employs the open software for the same interface and fair evaluation.

Figure 8. A block diagram for performance evaluation.

The experiment was conducted in a virtual Linux environment with a 3.40 GHz processor, 16.0 GB of memory and Windows 10 64-bit operating system, as shown in Table 3.

Table 3. Experimental conditions.

| CPU          | Intel® Core™ i7-8700K Processor @3.70 GHz |
|--------------|------------------------------------------|
| RAM          | 16.0 GB                                   |
| OS           | Windows 10 64-bit Windows Subsystem       |
|              | for Linux Ubuntu 18.04 LTS               |
| Software     | FFmpeg 4.2.2                              |
The experiment was conducted in a virtual Linux environment with a 3.70 GHz processor, 16.0 GB of memory and Windows 10 64-bit operating system, as shown in Table 3. For test sequences, six video sequences from class A1 and A2 with 4K resolution under the common test conditions of versatile video coding (VVC) were selected [18], and they were coded by H.264/AVC, HEVC and VP9 encoders. Table 4 shows the bit rates of the test bitstreams.

Table 4. Bitrates of the coded test bitstreams by H.264/AVC, HEVC and VP9 for test sequences.

| Sequence Name    | Bitrate (Mb/s) | H.264/AVC | HEVC | VP9 |
|------------------|----------------|-----------|------|-----|
| Campfire         | 54.7           | 24.6      | 45.0 |
| FoodMarket4     | 19.4           | 6.4       | 6.5  |
| Tango2          | 23.6           | 10.1      | 16.4 |
| CatRobot        | 20.7           | 9.8       | 20.3 |
| DaylightRoad2   | 25.1           | 12.0      | 28.7 |
| ParkRunning3    | 72.4           | 39.3      | 47.9 |

For performance evaluation, the thumbnail of the first frame of each test sequence was extracted for H.264/AVC, HEVC and VP9 bitstreams, and the extraction times were compared. The time saving (TS) for measuring the thumbnail extraction speed comparison was calculated, as defined by:

\[
TS = \left( \frac{T_{org} - T_{proposed}}{T_{org}} \right) \times 100
\]

Tables 5–7 show the comparison of the thumbnail extraction time of each codec compared to the conventional method. The acceleration rate of the proposed thumbnail extraction algorithm was 66% for H.264/AVC bitstreams, 52% for HEVC bitstreams and 48% for VP9 bitstreams. The acceleration ratios may differ slightly depending on the codecs. The computing time to decode the inverse transform and intra prediction of H.264/AVC was higher than that in the others; in addition, the speed factors differed depending on image characteristics because the ratios of transform skip or zero residuals influenced the thumbnail extraction computing time. The proposed algorithm focuses on the removal of the inverse transformation and intra prediction portions.

Table 8 shows the peak signal-to-noise ratio values of the thumbnails compared with the conventional full decoding method for each codec. The result shows that the PSNR value significantly differs depending on the sequences. This is because the proposed method stores only pixels necessary for the thumbnail in the thumbnail buffer, and the down-sampling process is removed. Therefore, in the case of sequences containing complex textures, they suffer aliasing; thus, PSNR can be lower than for others. Nevertheless, the thumbnails generated by the proposed method had a sufficient level of visual quality for commercial use in thumbnail applications.

Table 5. Comparison of the extraction times for H.264/AVC bitstreams. TS: time saving.

| Sequence Name    | Original (ms) | Proposed (ms) | TS  |
|------------------|---------------|---------------|-----|
| Campfire         | 256           | 88            | 66% |
| FoodMarket4     | 191           | 69            | 64% |
| Tango2          | 209           | 72            | 66% |
| CatRobot        | 238           | 84            | 65% |
| DaylightRoad2   | 234           | 84            | 64% |
| ParkRunning3    | 319           | 103           | 68% |

Average 66%
Table 6. Comparison of the extraction times for HEVC bitstreams.

| Sequence Name    | Original (ms) | Proposed (ms) | TS |
|------------------|---------------|---------------|----|
| Campfire         | 178           | 94            | 47%|
| FoodMarket4      | 159           | 75            | 53%|
| Tango2           | 156           | 66            | 51%|
| CatRobot         | 178           | 84            | 53%|
| DaylightRoad2    | 184           | 88            | 58%|
| ParkRunning3     | 247           | 122           | 52%|
| **Average**      | **52%**       |               |    |

Table 7. Comparison of the extraction times for VP9 bitstreams.

| Sequence Name    | Original (ms) | Proposed (ms) | TS |
|------------------|---------------|---------------|----|
| Campfire         | 300           | 175           | 42%|
| FoodMarket4      | 259           | 116           | 45%|
| Tango2           | 256           | 141           | 54%|
| CatRobot         | 297           | 163           | 55%|
| DaylightRoad2    | 344           | 178           | 45%|
| ParkRunning3     | 444           | 206           | 48%|
| **Average**      | **48%**       |               |    |

Table 8. PSNR (dB) compared with the conventional method for each codec.

| Sequence Name    | H.264 | HEVC | VP9 |
|------------------|-------|------|-----|
| Campfire         | 24.02 | 25.05| 24.07|
| FoodMarket4      | 31.48 | 31.84| 31.63|
| Tango2           | 28.17 | 28.62| 28.07|
| CatRobot         | 23.74 | 24.37| 23.93|
| DaylightRoad2    | 25.26 | 25.70| 25.29|
| ParkRunning3     | 21.82 | 22.53| 22.49|
| **Average**      | **25.75**| **26.35**| **25.83**|

Figures 9–11 show the thumbnails from the proposed method alongside those from the conventional method, which performs down-sampling after decoding, for the ‘Tango2’, ‘Campfire’ and ‘ParkRunning3’ sequences. They were 4K sequences of 3840 × 2160 size. Figures 9–11 show the thumbnails extracted by the proposed algorithm and exiting software, respectively. The width and height of the thumbnails were 1/8 of the original ones, respectively. Since the intra prediction was performed with the reconstructed pixels of the upper and left neighboring boundaries, errors at the boundary pixels propagated to the consecutive blocks. In the worst case, the error could propagate up to the last coding block of the slice or picture. The proposed algorithm was designed to reconstruct the boundary pixels with the same inverse transforms and prediction. The reconstructed pixels were efficiently stored in the down-sampled reference buffers. In addition, the thumbnail was a low-resolution image, the degradation of the image quality in the proposed method was insignificant and it was difficult to see their visual difference.
Figure 9. Subjective quality performance for the 'Tango2' sequence.

Figure 10. Subjective quality performance for the 'Campfire' sequence.
Figure 11. Subjective quality performance for the ‘ParkRunning3’ sequence.

4. Conclusions

The conventional thumbnail extraction method consists of the decoding and down-sampling stages for a thumbnail. However, the decoding and down-sampling processes have high computational complexity and memory usage. In this paper, we proposed a partial decoding method and a memory structure for high-speed thumbnail extraction. The proposed partial decoding method replaces the inverse transform and intra prediction in the decoding process with partial inverse transform and partial intra prediction. The computation complexity is reduced by restoring the one-pixel rule per inner 8 × 8 sub-block along with the rightmost pixels and the lowermost pixels. In addition, we designed a memory structure suitable for the partial decoding process. The proposed memory structure reduces 98% of the restoration buffer required for 4K videos by replacing the restoration buffer with a low-resolution thumbnail buffer and two reference line buffers for intra prediction. In order to evaluate the performance of the proposed fast thumbnail extraction method, we implemented the proposed algorithm with the FFmpeg H.264/AVC, HEVC and VP9 decoders and compared them with the speed of the conventional thumbnail extraction algorithm implemented on FFmpeg. For 4K resolution videos, we compared running times by extracting the thumbnail of the first frame of the test sequences. With the proposed method, we reduced 66% of the process time for H.264/AVC, 52% for HEVC and 48% for VP9. In addition, we reduced the amount of required memory without visual quality loss. The proposed algorithm was commercialized and implemented on an ARM processor for 2019 and 2020 LG televisions.

Author Contributions: Conceptualization, J.L., D.S.; methodology, D.S.; investigation, S.J., J.L.; software, J.B., S.J.; writing—original draft preparation, J.B., D.S.; writing—review and editing, J.B., D.S.; project administration, K.K.; supervision, D.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2020-2016-0-00288)
supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP) and LG Electronics “Development of extraction technique of high-speed video thumbnail”.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Tudor, P. MPEG-2 video compression. Electron. Commun. Eng. J. 1995, 7, 257–264. [CrossRef]
2. Wiegand, T.; Sullivan, G.; Bjontegaard, G.; Luthra, A. Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. 2003, 13, 560–576. [CrossRef]
3. Sullivan, G.J.; Ohm, J.; Han, W.J.; Wiegand, T. Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. 2012, 22, 1649–1668. [CrossRef]
4. Bankoski, J.; Wilkins, P.; Xu, Y. Technical overview of VP8, an open source video codec for the web. In Proceedings of the 2011 IEEE International Conference on Multimedia and Expo (ICME), Washington, DC, USA, 11–15 July 2011; pp. 1–6.
5. Mukherjee, D.; Bankoski, J.; Grange, A.; Han, J.; Koleszar, J.; Wilkins, P.; Xu, Y.; Bultje, R. The latest open-source video codec VP9—an overview and preliminary results. In Proceedings of the Picture Coding Symposium (PCS), San Jose, CA, USA, 8–11 December 2013; pp. 390–393.
6. Jo, H.H.; Sim, D.G.; Jeon, B.W. Hybrid parallelization for HEVC decoder. In Proceedings of the 2013 6th International Congress on Image and Signal Processing (CISP), Hangzhou, China, 16–18 December 2013.
7. Ryu, H.C.; Ahrn, Y.J.; Mok, J.S.; Sim, D.G. Performance Analysis of HEVC Parallelization Methods for High-Resolution Videos. IEEE Trans. Smart Process. Comput. 2015, 4, 28–34. [CrossRef]
8. Lee, J.Y.; Moon, S.K.; Sung, W.Y. H.264 decoder optimization exploiting SIMD instructions. In Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS), Tainan, Taiwan, 6–9 December 2004.
9. Chi, C.C.; Alvarez-Mesa, M.; Bross, B.; Juurlink, B.; Schierl, T. SIMD acceleration for HEVC decoding. IEEE Trans. Circuits Syst. Video Technol. 2014, 25, 841–855. [CrossRef]
10. Kim, E.S.; Um, T.W.; Oh, S.J. A fast thumbnail extraction method in H.264/AVC video streams. IEEE Trans. Consumer Electron. 2009, 55, 1424–1430. [CrossRef]
11. Kim, M.H.; Lee, H.J.; Stull, S.H. Fast thumbnail generation in integer DCT domain for H.264/AVC. IEEE Trans. Consumer Electron. 2011, 57, 589–596. [CrossRef]
12. Chen, C.; Wu, P.H.; Chen, H. Transform-Domain Intra Prediction for H.264. In Proceedings of the IEEE International Symposium on Circuits and Systems, Kobe, Japan, 23–26 May 2005; pp. 1497–1500.
13. Lee, W.J.; Jeon, G.G.; Jeong, J.C. Fast thumbnail extraction algorithm for HEVC. In Proceedings of the 2015 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 9–12 January 2015; pp. 321–322.
14. Chen, W.H.; Smith, C.H.; Fralick, S.C. A fast computational algorithm for the discrete cosine transform. IEEE Trans. Commun. 1977, 25, 1004–1009. [CrossRef]
15. Shen, S.; Shen, W.; Fan, Y.; Zeng, X. A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards. In Proceedings of the IEEE International Conference on Multimedia and Expo, Melbourne, VIC, Australia, 9–13 July 2012; pp. 788–793.
16. FFmpeg Software. Available online: https://www.ffmpeg.org/ (accessed on 2 February 2021).
17. Ffmpegthumbnailer Software. Available online: http://code.google.com/p/ffmpegthumbnailer/ (accessed on 2 February 2021).
18. Bossen, F. Common test conditions and software reference configurations. In Proceedings of the Joint Collaborative Team on Video Coding 12th Meeting, Geneva, Switzerland, 14–23 January 2015.