Wavelet-Based Fast Decoding of 360° Videos

Colin Groth, Sascha Fricke, Susana Castillo, and Marcus Magnor

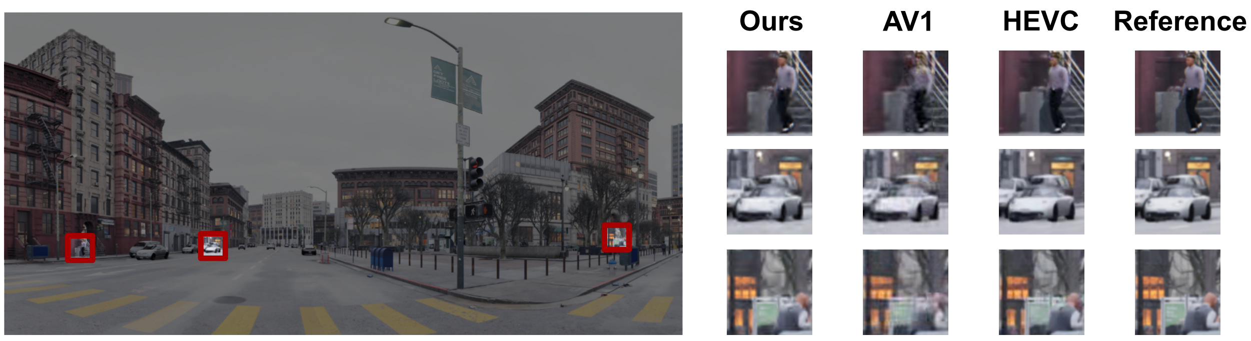


Fig. 1: In this paper, we propose a wavelet-based video codec that is able to load and decode the data of 360° videos viewport-dependently by exploiting the properties of the wavelet transform. While the quality is on par with state-of-the-art video codecs, we can achieve significantly faster playback times. Here, a visual comparison of our codec is given with HEVC, AV1 and the uncompressed reference. The samples are out of the computer-generated *City* video (left), all with 8k full-frame resolution.

Abstract— In this paper, we propose a wavelet-based video codec specifically designed for VR displays that enables real-time playback of high-resolution 360° videos. Our codec exploits the fact that only a fraction of the full 360° video frame is visible on the display at any time. To load and decode the video viewport-dependently in real time, we make use of the wavelet transform for intra- as well as inter-frame coding. Thereby, the relevant content is directly streamed from the drive, without the need to hold the entire frames in memory. With an average of 193 frames per second at 8192×8192 -pixel full-frame resolution, the conducted evaluation demonstrates that our codec's decoding performance is up to 272% higher than that of the state-of-the-art video codecs H.265 and AV1 for typical VR displays. By means of a perceptual study, we further illustrate the necessity of high frame rates for a better VR experience. Finally, we demonstrate how our wavelet-based codec can also directly be used in conjunction with foveation for further performance increase.

1 Introduction

Codecs provide efficient compression allowing to store hundreds of videos on a single drive. This efficiency results from a precise adaptation to their specific application. Video codecs like HEVC/H.265 or AV1 use the discrete cosine transform (DCT) and motion compensation for high compression rates at a reasonable perceptual quality. The high compression, however, requires complex coding procedures. The specific hardware decoders of modern graphics cards compensate for some of this decoding load.

360° videos are a sophisticated form of viewing experience which have become known with the spread of virtual reality (VR) technology. In 360° videos the user can change their view anywhere, since the information of the entire space is available (three degrees of freedom (DOF)). This free exploration is not possible with traditional videos where the field of view (FOV) is limited. Accordingly, 360° videos are best suited for an immersive user experience for video playback in VR. For comparable quality, the resolution of 360° videos must significantly exceed those of videos with a discrete view, since the representation on the display device only corresponds to a fraction of the whole video frame. For a modern VR headset with 2000x2000-pixel resolution per eye and 90° FOV, a comparably resolved 360° video would require 8000x8000 pixels (stereo) with up to 120Hz temporal resolution. However, only the part of the frame that lies within the device's viewport at the time of decoding is relevant for rendering. Current DCT-based codecs do not allow to load or decode only a defined part of a frame due to their complex non-linear structure. Accordingly, with 360° videos the common practice is to load, upload, and decode the entire 360° frame, even though only a small part of the frame is considered for the rendering. Furthermore, the recording quality of 360° cameras is limited. Consequently, the resolution and frame rate of 360° videos nowadays do not come close to the quality of the renderings of virtual environments. Although there are some existing ideas that already try to improve the compression – e.g. tiling the frame into separate regions – these approaches often go against the basic compression concept of DCT and are only a compromise at the expense of compression efficiency.

An alternative to the DCT for compression is the wavelet transform, which offers two decisive advantages over the former: (1) different parts of the image can be loaded and decoded individually, e.g. the viewport of an head-mounted display (HMD); (2) the encoding is performed in frequency layers, which are each halved in frequency. Decoding an area in fewer steps is equivalent to displaying the image area at a lower resolution. Early attempts to use wavelet transform for image compression were limited to traditional presentations with a discrete FOV and have not gained wide use.

In this paper, we propose a wavelet-based codec for the compression of 360° videos. We particularly aim for high display speeds of high-resolution videos. Our implementation of the wavelet-based codec uses the wavelet transform for inter- and intra-frame compression. To the best of our knowledge, this is the first codec for 360° videos based on wavelet transforms. In comparison with modern codecs (HEVC/H.265 and AV1) and related work, we show that our wavelet-based approach offers a significant speed advantage while providing a comparable video quality and reasonable compression rate. In addition, we introduce foveated decoding. With foveated decoding the properties of the wavelets are used to gradually decrease the resolution with the distance from the focal point. Such foveation is, so far, only known from virtual scenes and offers further opportunities to improve both decoding

• *All authors are with the Institute for Computer Graphics at TU Braunschweig, Germany.
E-mail: {groth, fricke, castillo, magnor}@cg.cs.tu-bs.de.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

speed and image perception. The Code for our codec is available at `https://github.com/ColinGroth/Wavelet_Codec_360`.

The contributions of the paper are summarized as follows:

- a novel approach to encode and decode videos for fast viewport-dependent playback based on wavelet transforms
- wavelet-based inter-frame transform without keyframes
- a technique to implicitly apply foveated rendering for wavelet-encoded videos during run-time
- objective and perceptual evaluation and comparisons with state-of-the-art video codecs and related work

2 Related Work

Loading the entire frame for 360° video playback in VR is inefficient since only a fraction of the frame is actually rendered. However, this procedure is the most common practice, as it reduces the need to adjust the standard video pipeline. In the following, we first discuss previous work aiming at a more resource-efficient presentation by adapting existing codecs and, in the second part, we introduce former attempts for wavelet-based codecs.

2.1 Viewport-Adaptive Display Techniques for Videos

Zare *et al.* [36] proposed to use a tiling scheme to increase the decoding speed of streamed 360° HEVC videos. Their experimental setup consisted of a pipeline with a dedicated server and client side. On the server side, the same video was encoded in high and low resolution. With the motion constrained tile sets (MCTS) extension of HEVC the tiling was enabled for both versions of the video. The client, on the other hand, requested the required tile sets from the server based on the viewport. The authors tested different tiling schemes. The scheme with the most tiles (18 tiles) showed the highest bitrate savings (-40% based on Bjontegaard Delta Bitrate BD-BR [2]). However, the compression losses increase proportionally with the number of used tiles, since all tiles are saved independently.

The tiling approach was later officially formulated by the Moving Picture Experts Group (MPEG) into the omnidirectional media format (OMAF) standard for storage and distribution of 360° videos [5]. The idea of OMAF is comparable to the work of Zare *et al.* [36] and is applied to HEVC or AVC video codecs. The viewport-dependent streaming also uses MCTS to split the frames into tiles, each encoded in different qualities [11].

Sreedhar *et al.* [29] also recognized the technical challenges of bandwidth associated with high-resolution 360° videos. The main focus of their work was the mapping techniques in which the recorded spherical scenes are packed in a rectangular frame. The most used mapping techniques are equirectangle and cubemap projection, which were also found as the most effective in their scenario. For the comparison, the authors presented a methodology of the rate-distortion performance of the schemes.

In the work of Corbillon *et al.* [7] the 360° videos are separated in individual tiles and offered in different resolutions. Unlike former works, the single tiles are created in different versions with only a selected part of every tile in a better visual quality. While the 360° video is streamed, the client communicates its viewpoint to the server which selects the tiles so that the viewpoint is in the higher quality region. The paper does not specify actual display speeds, but it should be clear that the technology can save bandwidth.

The performance of 360°videos can be improved not only on the software side. Recent works investigate how computation reduction and energy efficiency can be achieved through hardware design. Sun *et al.* [31] designed special hardware to accelerate the perspective projection on a FPGA. The results show significant energy reduction without a loss of performance. Zhao *et al.* [38] leverages redundancies across frames and tiles of left and right eye projection. Results indicate a computation reduction of 34% and energy saving of 17% for implementations on a GPU or FPGA. Leng *et al.* [20] propose energy-efficient VR video processing by optimising the projective transformation with semantic-aware streaming on the server-side and hardware-accelerated rendering on the client. The authors demonstrate that up to 42% energy of the VR device can be saved for 360° video content using their system design.

2.2 Wavelet based codecs

Probably the best known use of wavelets for imagery is the JPEG2000 image compression standard [22, 32]. At the turn of the millennium, it initiated a new form of image compression and was meant to replace DCT-based image compression formats. JPEG2000 supports lossless and lossy compression. The wavelet transform operates with the biorthogonal wavelets, either the Cohen–Daubechies–Feauveau (CDF) $9/7$ wavelet [6] for lossy compression or the LeGall-Tabatabai (LGT) $5/3$ wavelet [18] for lossless compression. The standard has four levels of decomposition as a default since there is not significant improvement in using higher decomposition levels when compressing images [15]. One general advantage of wavelet compressed imagery is the progressive decoding, so that the quality of the visualisation improves progressively when more information is received. This progressive decoding is also supported in JPEG2000.

The JPEG2000 image standard was later extended to include video files. The extension is known as *Motion JPEG2000* and is based on the MP4 format. This video standard uses the JPEG2000 coding for the compression of the individual frames. An inter-frame compression does not take place. Thus, Motion JPEG2000 is more of a container format for the joint wrapping of JPEG2000 compressed frames. Note that our video codec differs clearly to the Motion JPEG2000 standard. We apply a wavelet-based inter-frame transform to successive frames and use a special data structure that allows for viewport-adaptive data streaming. However, we took inspiration from JPEG2000 to design the frame-wise transform of our pipeline, e.g. we use the CDF 9/7 wavelet to obtain frequency information of single frames.

Efforts to create video codecs based on wavelet compression are rare and nowadays exclusively experimental. The most extensive attempt to create a wavelet-based video format to date was undertaken by BBC Research in 2008 [33]. The resulting versions of the codec were named *Dirac* and *Schrödinger* in honour of the Nobel Prize-winning physicists. Dirac supports lossy and lossless coding for which it uses the same wavelets as JPEG2000 (CDF 9/7 wavelet or LGT 5/3 wavelet). The motion compensation is performed with the overlapped-block motion compensation (OBMC) logic for an effective inter-frame prediction [24]. Unfortunately, this overlap also prevents effective intra-prediction, since there are no unique separations for overlapping blocks, as is the case with common DCT-based codecs. The overlapping-block logic further prevents an efficient use of viewport-dependent decoding and is only designed for full frame data retrieval.

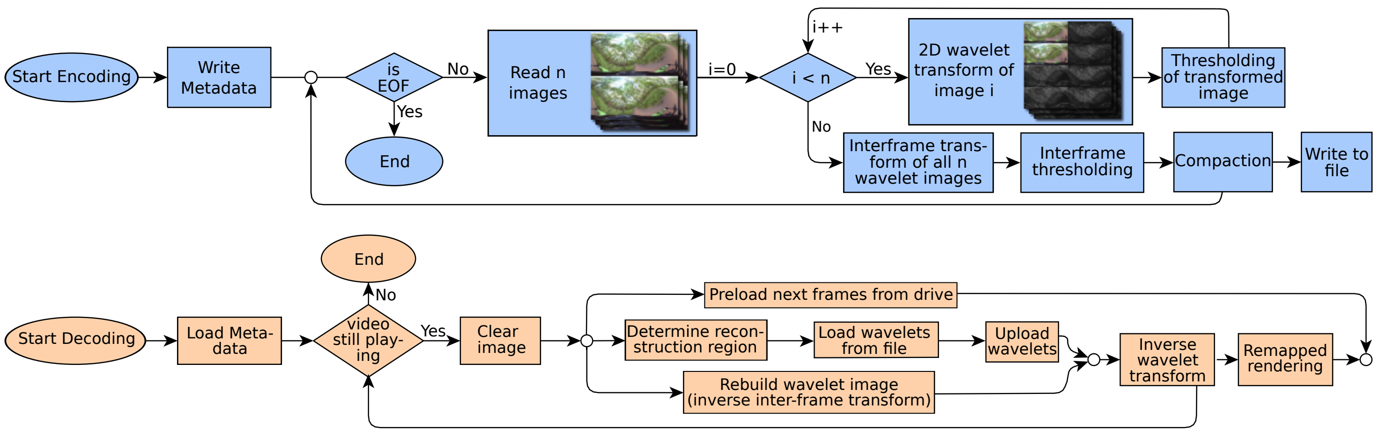
However, the codec could not gain wide popularity and further development was discontinued more than a decade ago. The reasons for the codecs limited success are not entirely clear, but may be related to an inability to provide significant improvement over established codecs like H264. Dirac and Schrödinger are now abandoned and no longer available.

3 Method

Two concepts that most video codecs apply for data compression are intra- and inter-frame coding. In practice, these methods are commonly applied with some information loss to achieve better compression ratios. *Intra-frame coding* usually refers to the transformation of the data of one frame to a different representation that can be compressed more efficiently. *Inter-frame coding* utilizes redundancies between multiple frames to reduce the data size. In the following, we describe how we implement both concepts with wavelet transforms. Figure 2 shows an overview of how this transformations are integrated in our program flow for the encoding and decoding of 360° videos.

3.1 Frame-wise Transform

The core of the frame-based compression of our codec is a 2D fast wavelet transform (FWT). Similar to other codecs, the transform of the frame data allows for a better compression, which in the raw state is too large to be stored. For example, a one minute 360° video in 8k resolution would contain around 300GB uncompressed data. We



Start Encoding
Write Metadata
is EOF
No
Yes
End
Read n images
i=0
i++
i < n
Yes
No
2D wavelet transform of image i
Thresholding of transformed image
Interframe transform of all n wavelet images
Interframe thresholding
Compaction
Write to file
End
No
Start Decoding
Load Meta-data
video still playing
Yes
Clear image
Preload next frames from drive
Determine reconstruction region
Load wavelets from file
Upload wavelets
Rebuild wavelet image (inverse inter-frame transform)
Inverse wavelet transform
Remapped rendering

Fig. 2: Our program flow for encoding (blue) and decoding (orange) a video with our wavelet-based codec.

transform an input frame s of $(N \times M)$ pixels for a discrete 2D position (x, y) and frequency γ by the wavelet transform W with the mother wavelet ψ.

$$W_\psi s(\gamma, x, y) = \sum_{i=0}^{N} \sum_{j=0}^{M} \psi_{\gamma,x,y}(i,j)s(i,j) \tag{1}$$

To compress the transformed frame $W_\psi s$ all coefficients below a certain threshold T_s are set to zero, so that:

$$W'_\psi s := \begin{cases} W_\psi s, & if|W_\psi s| > T_s, \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

For a properly chosen T_s, this operation has only minor implications for the quality of the reconstructed image. This is especially true for high frequencies and is a general characteristic of the frequency domain. Usually, in natural images most of the information is contained in the low frequencies, which are represented by only a small number of coefficients [34]. As usual for the FWT, with every step of the 2D transformation, the resolution of the image approximation is halved in both dimensions. In W_ψ this is addressed by the frequency layers over γ.

Former research has shown that the discrete wavelet transform can achieve better image reconstruction than a DCT-based method at high bit compression ratios [3]. For the frame-wise transformation of the image we use the CDF 9/7 wavelet [6]. The CDF wavelet is known to perform especially good on natural imagery and is also used for lossy compression in the JPEG2000 standard [15].

During playback, the compressed video information is decoded with an inverse fast wavelet transform (iFWT) to obtain the original images. This reconstruction is not conducted for the entire image, but only for the part of the 360° panorama that lies in the viewport of the display device. For a viewport-dependent reconstruction, we define the location of the viewport on a low resolution representation of the frame in binary form. This binary mask is uploaded together with the wavelet coefficients and is used for the inverse wavelet transform. The resolution of the binary representation is 256×256 pixels for a 8k stereo frame and can define arbitrary reconstruction shapes.

In theory, the transform can be performed until only one pixel defines the lowest frequency over the whole image. However, the low-frequency levels of the wavelet transform contain fewer discrete data points since the high-resolution in the frequency domain results in a low resolution in time due to the Heisenberg theorem [12]. Also, the wavelet coefficient values of these pixels can only be compressed inefficiently because the low-frequency information is significantly more important than high frequencies in natural image reconstruction. By default, we perform the wavelet transform for $l_{max} = log_2(\frac{N}{32}) - 2$ levels. For the number of wavelet levels, we took inspiration from the JPEG2000 standard, but also performed several pilot studies. While we found 6 levels to be the optimal default solution for 8k videos, the number of wavelet levels can be chosen individually per video.

3.2 Inter-Frame Coding

Inter-frame coding describes the compression of temporal information. In videos, the time component t is represented implicitly by a set of successive frames. In modern codecs the inter-frame compression is performed with keyframes and motion vectors where only the information differences are encoded. While this technique offers impressive compression rates, a compression with keyframes has the disadvantage that its speed depends on the linear information retrieval. When the video is skipped, all information since the last keyframe has to be reloaded first. With our codec we wanted to get rid of this disadvantage and at the same time maintain a good compression rate between inter-frames. To achieve this purpose we apply a second one-dimensional wavelet transform that encodes the temporal pixel differences. The second wavelet transform is applied on a set of wavelet images resulting from the frame-wise wavelet transform $W_\psi s$. Here, we use a one-dimensional form of W_ψ with $s(\gamma, t)$ for the frequency γ of the temporal changes of every pixel over time. In other words, our wavelet based inter-frame transform encodes the frequency changes over time. Typically, even with movement in the frame the temporal information only changes on single frequencies. All frequencies that do not or only slightly change are compressed by our inter-frame transform. As result, the speed of the decoding is unaffected by the direction in which the viewport moves. In theory both, the frame-wise transform and the inter-frame transform, can be combined to one 3D wavelet transform. However, this 3D transform would not offer us the possibility to decode different areas of a frame in different resolutions for the same computational costs. Furthermore, the separation allows us to apply different wavelets and thresholds per transform and respond adaptively to individual circumstances.

Every inter-frame transform of n consecutive frames we call *inter-frame set*. Thereby, n is a power of two value. The number of frames per inter-frame set can be defined per video and may be bigger the less motion is in the video. In contrast to the frame-wise transform, the inter-frame transform is always executed to the last level. For the inter-frame wavelet transform we use the Haar wavelet [10]. The Haar wavelet is the only wavelet with no overlapping of the wavelet filters and can therefore be reconstructed by loading only one coefficient per level for the high and low pass filtering. Reconstruction of one specific pixel by a Haar wavelet transform with n levels only requires $log_2(n)$ additions of the correct wavelet coefficients multiplied by the high pass filter position (either -1 or 1). As a result, for the inverse inter-frame transform we can iterate over the uploaded wavelets rather than over all pixels of the target section. This characteristic is unique to the Haar wavelet and allows a rapid inter-frame reconstruction. The speed of the inter-frame reconstruction is important since the inverse inter-frame transform runs on $log_2(n)$ frames every time one frame is decomposed.

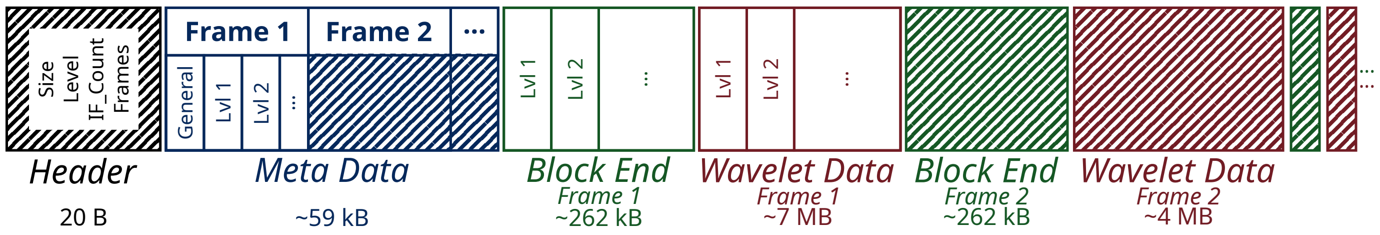


Fig. 3: Data arrangement of our video format. The sizes of each section are given by an example video in 8k resolution.

By iterating over the uploaded pixels rather than a target section we implicitly synthesise only the part of the image that is in our defined FOV.

3.3 Thresholding

In order to achieve the necessary storage savings, we have to determine which wavelet coefficients are least essential for the reconstruction. We will refer to this step as *thresholding*. The chosen threshold value is decisive for the intensity of the compression. Thereby, the threshold always represents a trade-off between quality of the reconstructed image and size of the video file. The threshold of 0 represents lossless compression while a threshold of 1 produces the smallest file. Reconstructing an image with too little frequency information may result in a blurry representation with less details. We derive a threshold T from a user-defined constant α and the level l of the transform:

$$T(x,y) = \alpha \left(\frac{l_{max} - l}{l_{max}} \right)^2 + H, \tag{3}$$

where H specifies a mapping factor which depends on the mapping technique (cf. Sec. 3.6). In the encoding, the frames are thresholded twice: once after the frame-wise 2D FWT, and again following the inter-frame transform. We use two separate threshold operations as both wavelet transforms aim for a different encoding: The frame-wise transform encodes the different frequency information in the respective spatial resolutions. The inter-frame transform encodes temporal frequency information of every wavelet coefficient. Thresholding the values only once is possible but, in our experience, can lead to unwanted interactions and a worse compression rate. Both thresholding operations are independent and have their own threshold value. While the first thresholding is applied on every frame independently, the thresholding of the inter-frames considers all n frames of the inter-frame set. In this latter thresholding operation, the different levels are defined by the relative frame number from t rather than the pixel position inside the frame.

High frequency information was found to be less important for the perceptual quality of an image than low frequency information [34]. We scale the threshold by the frequency level of each coefficient in a quadratic function (cf. Eq. 3). Accordingly, more coefficients may be zeroed out at high frequencies. This thresholding weighting follows common procedures of other codecs like JPEG2000 [22, 32].

Quantization: Similar to other codecs, we represent the color values of the pixels in the video file by one byte per color component. In the quantization, the 32 bit float color components of the wavelet transform are mapped to the byte representation of the compressed output. We use the extreme values of the wavelet coefficients for normalization in order to achieve the highest possible spatial resolution in this discretization. Therefore, one discrete color value c_d is defined by

$$c_d = \frac{c_n - c_{min}}{c_{max} - c_{min}} * 255 \tag{4}$$

with c_n as the floating-point representation of the n-th pixel and c_{min} and c_{max} as the minimum and maximum values of all coefficients, respectively. The normalization is performed with a separate minimum and maximum for the approximation area (last layer of the transform) and the wavelet layer and for each inter-frame. The normalization is inverted during reconstruction. The minimum and maximum values are stored with the metadata in the file.

3.4 File Format

The structure of our video file is illustrated in Figure 3. We designed the layout to allow for a fast and viewport-dependent streaming of the data. Starting with a file header, general information on the video is offered. This data includes the number of frames, size of the frames and number of levels of the wavelet transform. Following the header, metadata information on every single frame is provided. This frame-wise metadata includes information about where the frame starts and ends in the file or the overall number of wavelets. The frame metadata also provides information on the individual levels of this frame. The header as well as the entire metadata are preloaded when the video is started and are kept in the working memory.

The position (x,y) of one particular wavelet coefficient within the frame is given by an index that is saved along with the wavelet value. However, due to the compression, the position of individual coefficients within the file is unknown. While it is possible to find the data for (x,y) with binary search on the video data, this inconsistent access to the storage drive adds an unwanted delay to the loading process. Instead we divide the transformed frames into a logical grid of small blocks (default size 32×32-pixel). This block allocation is only relevant for the compression but does not affect the wavelet transforms which operate on the entire images. Note, that this is different from blocking in DCT. In the video file we store one pointer for each block, located in the *BlockEnd* section (see Figure 3). This pointer indicates where the last wavelet coefficient inside the respective block can be found in the video file. In the file the coefficients are stored block after block, which allows a whole series of blocks to be loaded by two of these block-end pointers. During decoding, the block pointers of a frame are preloaded before the frame is processed. By alternately storing the block-end and wavelet data packages of the frames in the video file we can avoid compute-intensive rearrangements of the file during encoding. Inside one block of wavelet coefficients or block-end information the data is ordered level-wise starting with the lowest frequency layer.

3.5 Parallel Wavelet Processing

Since VR users move their head, the part of a 360° video that is rendered can change continuously. These viewpoint changes complicate the buffering of subsequent frames with a viewport-dependent video stream. However, buffering is necessary to avoid load peaks and latencies that disrupt the virtual experience and can induce cybersickness [30]. Therefore, we load the viewport data of the next frames asynchronously while the current frame is decoded. Until the time of rendering, all frames are updated continuously in case that the look direction changes. The number of frames that are preloaded corresponds to the inter-frame size. Preparing more frames is not always useful, as the view direction may change strongly over longer periods of time. During our experiments no substantial latency was measured that arose from the buffering.

Due to the inter-frame compression, one frame is reconstructed by the wavelet coefficients stored in multiple inter-frames of the respective inter-frame set. We rebuild the wavelet representation $W_\psi s$ of one frame on the GPU while we already upload the wavelet data for the next inter-frame in parallel (see Figure 4). This rebuild already includes the inverse inter-frame transform, as described in Section 3.2. The reconstruction of the original frame by the 2D iFWT is performed once all inter-frames are processed and the inverse inter-frame transform is completed.

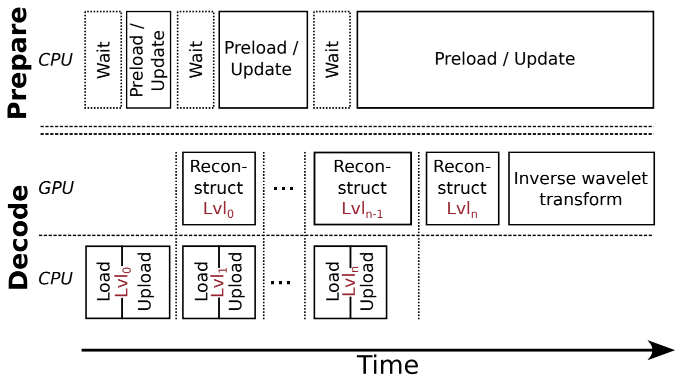


Fig. 4: Parallel processes for the reconstruction of one frame. The prepare thread buffers the next frames. In the *wait* sections the drive is occupied by the decoding process.

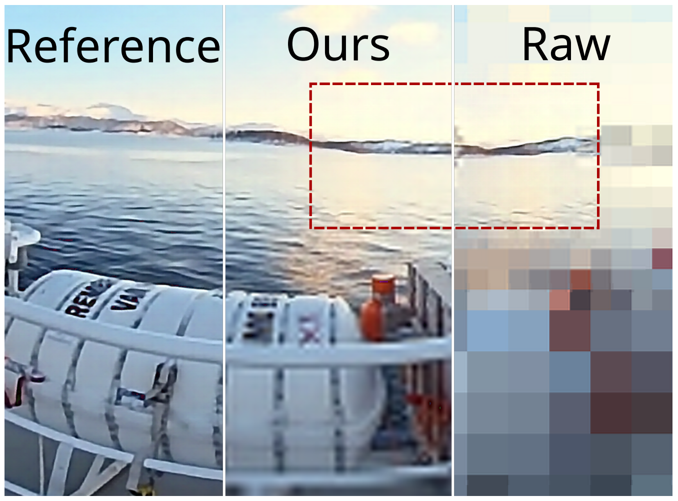


Fig. 5: Our foveated decoding compared with the full-resolution reference frame. On the right, the information density of the foveated decoding is visualised. The dashed line indicates the foveal region.

3.6 Frame Mapping

360° videos are representations of a recorded 3D sphere, brought to a rectangular frame by a projection. The position of the information in the projected frame is defined by its mapping. Among the most popular mapping techniques are equirectangular mapping and cubemaps. Our codec is defined to be independent of the mapping technique. As the reconstructed areas are given in a low resolution presentation of the frame (cf. Sec. 3.1), the areas needed for the frame mapping can be set in direct relation to the final reconstruction. We render the final re-projection of the FOV to the spherical presentation in an own shader which is run after the inverse wavelet transform is completed. This shader can react on multiple mapping types and also considers the stereo images. For the experiments we use the equirectangular projection. We tackle the redundancies in the pixel information near the poles by gradually increasing the mapping factor H towards the poles. For an equirectangular projected frame with the dimension S we define $H(y) = 1 - \sin(y\pi/S_y)$. With the adjusted threshold, we experience equal performance at all viewing angles, including upward views.

3.7 Foveated Decoding

The information density of visual representations of the human eye are not equally distributed [28]. The images created on the retina of the human visual system follow a qualitative decline starting from the eye's fixation point. While people can perceive the full resolution of about one sixtieth of a degree in the fovea around the focus point, the information in the peripheral visual area is significantly lower in resolution [17].

So far we only discussed full resolution reconstructions of the viewport. However, when the eye gaze direction of the observer is available by eye tracking, we can utilise the properties of the wavelet transform to achieve what we call *foveated decoding*. With foveated decoding the resolution gradually decreases with the distance from the fovea. Our method is comparable to classical foveated rendering, except that in the periphery the decoding is accelerated while the rendering load is constant. The results are bandwidth savings and higher playback speeds.

For the foveated decoding, we utilise the level-wise structure of the wavelet transform. As described in Section 3.1, a wavelet representation is composed of individual levels, each corresponding to a defined frequency interval γ. Instead of loading the same FOV at each level, the full FOV of the viewport is loaded and rebuild only at the lowest frequency layer. After that the data that is loaded at every level corresponds to a increasingly smaller FOV. As a result the foveal area is compact and can be loaded effectively. The sizes of the individual resolution levels of the wavelet transform are defined in regard to the properties of the human eye [19]. Like human perception, we decrease the quality of the frames at a logarithmic rate [17]. The area with the full video resolution which stimulates the most central foveola is only about two percent wide [28].

The inverse wavelet transform is executed over the same number of data points as for a full resolution viewport but assumes the coefficients to be zero for the surrounding regions. With foveated decoding, we achieve a peripheral reconstruction in a visually appealing quality with a small number of coefficients (see Figure 5). With the foveation, up to 80% less data has to be loaded. The reconstructed areas are in rectilinear form and follow recent findings, which indicate advantages over a log-polar presentation [21].

4 Experiments

In objective and subjective evaluations we compare our codec against the common HEVC and AV1 codecs. The benchmarks for display speeds also include a tiled HEVC implementation from related work [36].

4.1 Dataset

For the evaluation, we aim to analyse a particularly diverse set of videos. Both computer-generated imagery (CGI) content and real world recordings are considered. Furthermore, we investigate moving camera trajectories as well as fixed recording positions.

The CGI reference frames are created with *Unreal Engine 5* with a high level of photorealism. The CGI scenes cover urban and nature scenery. The *City* scene mainly contains geometrical structures and straight lines and the recording takes place in a virtual New York City. The *Mountain* scene depicts a natural landscape with trees and a lake. The content of this scene contains a more heterogeneous shape composition compared to urban imagery. In the CGI scenes the camera trajectory moves along a predefined quarter circle path. The frames are rendered as 360° stereo images with 8192×8192 pixel resolution.

The first set of real-world videos is recorded with a moving camera trajectory and the display of rapid motions. Here, we use our videos from previously published work [8, 9] which have a higher resolution than typical moving-camera 360° videos due to our custom camera setup. The second category considers videos with a fixed recording position (further denoted by *). These videos were originally recorded by Mühlhausen *et al.* [23]. The original real-world videos are recorded with stereoscopic information in 6400×6400-pixel resolution at 30 frames per second (FPS).

Reference Data Creation: Pre-recorded videos can cause two problems for the evaluation. For one, the frame rates typically do not match the refresh rates of VR devices. Additionally, the data is already lossy compressed and can include serious compression artifacts. In order to address both problems, we first downscale the video data to 1024×1024-pixel to get rid of high-frequency compression artifacts and then perform temporal interpolation and upscaling of the

Table 1: **Display speeds.** Tiling refers to Zare *et al.* [36]. The foveated decoding (*FD*) is run with the high resolution version of our codec (*Ours$_{HQ}$*), all values are averages over multiple runs and given in FPS.

Videos	AVG fps ↑					
	HEVC	Tiling	AV1	*Ours$_{LQ}$*	*Ours$_{HQ}$*	*Ours$_{FD}$*
City$_{CGI}$	57.32	88.24	52.7	220.96	194.17	**228.8**
Mountain$_{CGI}$	58.27	87.95	61.57	222.75	202.28	**227.46**
Downhill	60.21	93.21	48.32	193.88	180.70	**206.65**
Horse	62.6	95.94	50.45	197.17	181.18	**207.16**
Climbing	65.36	92.55	52.53	195.63	187.56	**207.3**
Walking	65.32	93.1	52.76	198.24	187.88	**205.89**
Cave*	66.53	111.08	53.93	209.22	207.12	**210.28**
Boat*	67.13	110.15	55.07	205.21	201.06	**208.76**

data with state-of-the-art neural network approaches. The resulting frames are used as reference for our evaluation. The original video data is downscaled with bicubic interpolation by *OpenCV*. The temporal interpolation is run on the downscaled frames with RIFE [14]. The information from both eyes is processed individually to avoid artifacts at the edge. We increase the frame rate from the original 30 FPS to 120 FPS, which should be in line with the frequency of most modern VR glasses. For the resolution upscaling we use Nvidia VFX to create the final reference frames in 8196 × 8196-pixel resolution.

4.2 Objective Evaluation

We consider a high and low quality version of the wavelet-compressed videos. The inter-frame transform is applied in sets of four frames and compressed with an inter-frame threshold of 0.005. The frame-wise threshold is chosen to be 0.1 and 0.25 for the high and low quality version, respectively. The HEVC and AV1 encodings are performed with FFmpeg. Regarding quality, for HEVC we use a constant rate factor (CRF) of 30 (range 0–51) and for AV1 a CRF of 50 (range 0–65). The videos of both codecs are encoded in the common YUV420 colour space. The OMAF inspired tiling method is realised with HEVC encoded tiles with the fastest tiling scheme of Zare *et al.* [36]. However, we extended their tiling scheme for stereoscopic videos to a 6-by-6 grid layout (6-by-3 per eye). Following the former work, the middle row of both eyes is chosen with 90° height and all other rows with 45° height for a better central view performance.

We conduct all of our experiments on a commercially available computer with a NVIDIA RTX 3090 graphics card and an AMD Ryzen 5950X processor. The video data is stored on an on-board SSD. A HTC Vive Pro Eye is chosen as output device. All videos are displayed in our self-programmed video player which uses the Vulkan API to utilise the GPU. The decoding of HEVC and AV1 video data is performed with the Nvidia NVDECODE API. Thereby, the HEVC and AV1 decoding benefits from the hardware acceleration on the GPU. All videos are created from 1200 reference frames with 8192x8192-pixel resolution. To assure an equal comparison of all experimental conditions, we use head and eye tracking data of participant recordings.

The results regarding **computational time** are shown in Table 1. Our proposed codec allows for an average increase in performance of 197% compared to HEVC and AV1 and an increase of 91% over the tiling technique. This increase is even more significant when the lower quality version of our codec is used. In the experiment, the foveated decoding (*Ours$_{FD}$*) is applied on the wavelet-based video with high quality settings. Due to the foveation, the performance increases by 223% over HEVC allowing a better performance than the lower quality wavelet-encoded videos. Please note that we used the hardware accelerated on the GPU for the decoding of HEVC and AV1. The dedicated decoding chips allow for significant increases in decoding speed compared to conventional decoding. Additionally, the compute shaders for the mapping and rendering can be executed in parallel to the decoding through the dedicated chips. A comparable chip for decoding wavelet transforms could also significantly improve the performance of a wavelet-based codec while the compute unit can be used for other tasks.

Table 2: **Objective Quality Comparisons**. The results from the computational metrics, WS-PSNR, SSIM (higher is better), and LPIPS (lower is better), on all codecs compared with the reference frames.

Scene	Metrics	HEVC	AV1	*Ours$_{LQ}$*	*Ours$_{HQ}$*
City$_{CGI}$	WS-PSNR ↑	**17.89**	17.24	16.13	17.13
	SSIM ↑	.905	.89	.817	**.916**
	LPIPS ↓	.124	.167	.268	**.114**
Mountain$_{CGI}$	WS-PSNR ↑	**18.95**	18.06	16.44	17.43
	SSIM ↑	.927	.925	.851	**.933**
	LPIPS ↓	.158	.157	.32	**.145**
Downhill	WS-PSNR ↑	17.86	14.64	16.51	**17.99**
	SSIM ↑	.954	.95	.921	**.961**
	LPIPS ↓	.082	.103	.161	**.081**
Horse	WS-PSNR ↑	**18.32**	15.35	16.82	18.02
	SSIM ↑	.968	.966	.939	**.97**
	LPIPS ↓	**.054**	.07	.118	.057
Climbing	WS-PSNR ↑	**18.97**	18.93	17.54	18.63
	SSIM ↑	.973	.971	.952	**.976**
	LPIPS ↓	**.051**	.066	.115	.056
Walking	WS-PSNR ↑	18.02	18.13	16.83	**18.22**
	SSIM ↑	**.97**	.969	.928	.967
	LPIPS ↓	**.05**	.064	.117	**.05**
Cave*	WS-PSNR ↑	20.89	**20.99**	18.88	20.15
	SSIM ↑	.971	**.972**	.96	**.972**
	LPIPS ↓	**.039**	.04	.08	.046
Boat*	WS-PSNR ↑	19.44	**19.86**	17.54	18.84
	SSIM ↑	.984	**.986**	.954	.978
	LPIPS ↓	.03	**.027**	.078	.036

Table 3: **Compression ratios** of the video files in relation to the uncompressed data. The compression ratios are with respect to the full 360° FOV.

	City$_{CGI}$	Mountain$_{CGI}$	Downhill	Horse	Climbing	Walking	Cave*	Boat*
HEVC	651:1	973:1	431:1	810:1	1059:1	1039:1	5120:1	7408:1
AV1	933:1	1482:1	725:1	1524:1	1994:1	1994:1	9888:1	12659:1
Tiling	102:1	102:1	80:1	90:1	83:1	90:1	204:1	340:1
Ours$_{LQ}$	176:1	250:1	147:1	187:1	250:1	185:1	714:1	312:1
Ours$_{HQ}$	52:1	75:1	77:1	100:1	128:1	100:1	416:1	117:1

We compared the results' **quality** of all codecs by the commonly used metrics WS-PSNR, SSIM [35], and LPIPS [37]. The given values are averages over all frames and compared with the uncompressed reference frames (see Table 2). In terms of image quality our method performs equally to the other codecs, HEVC/H.265 and AV1, when high quality settings are chosen. As can be expected, the image quality is on a lower level when the low quality parameters are chosen for the wavelet-based encoding.

The **compression rates** of the wavelet files in both quality configurations as well as the comparison techniques can be seen in Table 3. With our wavelet-based approach, we are able to compress the raw information to over one hundredth in size for most videos. Despite this significant reduction in file size, our codec does not yet achieve the compression efficiency of HEVC and AV1 in its current state. We would like to emphasize that the focus of this work is on decoding speed and that our wavelet codec is not optimized to produce the smallest possible files. The tiled HEVC videos by the technique of Zare *et al.* [36] are on average twice as large as our wavelet-compressed video files due to the significant compression losses of the tiling process.

4.3 Perceptual Evaluation

In a next step, we compare the codecs in terms of observer's preferences to reveal subtle perceptual differences that were not found by the objective metrics. In this perceptual experiment, we are particularly

Table 4: **Perceptual results.** The top half refers to the uncorrected data of all participants ($n = 23$). In the bottom half the results are corrected for consistency ($n_{Speed} = 21, n_{Quality} = 15$).

ANALYSIS	CONDITION	SCENE	% Preferences Ours vs. AV1	 Ours vs. HEVC	 AV1 vs. HEVC	ζ	u	χ^2	p
Raw Data	Speed	City	**95.65%**	**82.61%**	4.35%	0.96	0.68	48.13	<.0001
		Mountain	**100.00%**	**56.52%**	8.70%	0.96	0.55	39.09	<.0001
		AVG	**97.83%**	**69.57%**	6.52%	0.96	0.60	84	<.0001
	Quality	City	**56.52%**	**56.52%**	21.74%	0.87	**0.08**	8.13	**.0434**
		Mountain	**86.96%**	43.48%	26.09%	0.74	0.23	18.22	.0004
		AVG	**71.74%**	**50.00%**	23.91%	0.80	**0.13**	20.55	.0001
Corrected for Consistency	Speed	City	**100.00%**	**80.95%**	4.76%	1	0.72	46.24	<.0001
		Mountain	**100.00%**	**57.14%**	4.76%	1	0.59	38.62	<.0001
		AVG	**100.00%**	**69.05%**	4.76%	1	0.65	82.48	<.0001
	Quality	City	**80.00%**	**60.00%**	20.00%	1	**0.20**	11.4	.0097
		Mountain	**86.67%**	46.67%	6.67%	1	0.39	19.4	.0002
		AVG	**83.33%**	**53.33%**	13.33%	1	**0.31**	29.60	<.0001

interested in analyzing to what extent the higher frame rates we can achieve with our wavelet codec also contribute to a better VR experience. As before, HEVC and AV1 serve as comparison techniques. The wavelet videos use the high quality version that showed to be most comparable to the other codecs in quality while still maintaining high display speeds. Due to the better data utilization of a wavelet-based codec, a higher resolution of the video or a higher frame rate can be provided. Thus, there are two conditions to be considered: the videos are in the same resolution but have different frame rates (*Speed* condition), or the videos are in different resolutions but provide the same frame rate (*Quality* condition). In the *Speed* condition all videos are at a resolution of 8192×8192 pixels. While the comparison techniques provide common 30FPS in this condition, the wavelet videos provides double the frame rate (60FPS) based on the results of the display speed measurements. In the *Quality* condition the frame rate of all videos is set to 60FPS but the HEVC and AV1 videos, here, have a resolution of 4096×4096 pixels. The length of at videos is 10 seconds. For the perceptual evaluation we only use the CGI scenes. The upscaled real-world videos have a corrupted stereo view because the upscaling algorithms are not designed for stereo footage. This display error should be irrelevant for the image-based metrics, but makes perceptual experiments impossible.

As the abstract feeling of comfort in a VR experience cannot be represented by a linear scale, we utilize the paired comparisons technique [16]. Given the same video with two different encodings played immediately after each other, the participants were instructed to choose the video they would prefer for a presentation in VR. The question was intentionally kept open and participants were free to base their decision on the visual quality, the temporal smoothness of the video, or a lower incidence of cybersickness.

Given three codecs, there are three possible comparisons per video: Ours vs. AV1, Ours vs. HEVC, and AV1 vs. HEVC. This results in a total of 12 decisions per person given two scenes for both conditions. In the experiment, the order of the paired stimuli comparisons for each participant was counterbalanced to avoid side effects.

A total of 23 participants took part in the experiment (10 females, age range = $22-59$, avg. age = 34.83, SD = 13.04) resulting in 184 votes *per codec*. The videos were shown in a HTC Vive Pro Eye HMD. For every comparison, the participants were ask to maintain a fixed head position to compare the same part of the 360° video.

4.3.1 Analysis and Results

On a first analysis, the voting of the participants ($n = 23$) leads us to the displayed results in the top half of Table 4 ("Raw Data", column "% Preferences"). The results show the analysis for each scene as well as an analysis per condition combining both scenes (labelled as "AVG").

In order to assess the results of the paired comparisons we follow the methodology from Setyawan and Lagendijk [26]. We first study the consistency of choices within one participant as well as the agreement in choices among all participants. The coherence of the answers of the participants is indicated by the **coefficient of consistency** $\zeta \in [0,1]$, with $\zeta = 1$ implying perfect consistency. Low consistency values of single participants can indicate that these individuals had difficulties to differentiate between the stimuli and thus, we can expect their judgement abilities to be worse than the average. As the number of methods $m = 3$, $\zeta < 1$ can only arise with the occurrence of one kind of circular triad such that $C_1 \rightarrow C_2 \rightarrow C_3$ but $C_3 \rightarrow C_1$. In Table 4 we present the coefficient of consistency as an average over all participants. The consistency of choices of single participants does not necessarily mean that identical choices were made between all participants. The diversity of preferences for the number of participants n is described by the **coefficient of agreement** u. Complete agreement is achieved with $u = 1$, meaning that all participants favoured the same method for all decisions. High disagreement indicated by low u values, on the other hand, suggests that participants had difficulties to make a joint choice. Disagreement may suggests either that the stimuli were perceived as not distinguishable or a general split of opinions about the stimuli. Here, the minimum u, and accordingly complete disagreement, is given by $u_{min} = -\frac{1}{(n-1)} \approx -0.045$. By the definition of Kendall and Babington-Smith [16], the coefficient of agreement u is derived by

$$u = \frac{2\tau}{\binom{n}{2}\binom{m}{2}}, \text{where} \tau = \sum_{i=1}^{m}\sum_{j=1}^{m}\binom{a_{ij}}{2} \quad (5)$$

with a_{ij} is the number of times method i is chosen over method j, whereby $i \neq j$. As before, the number of participants is denoted by n and the number of methods by m.

We determine the statistical significance of u by testing against the null hypothesis that all votes were chosen randomly. For a significant u we can conclude the alternative hypothesis that the agreement is above the value one would expect from random choices. To determine the significance under the null hypothesis we perform a chi-squared test (χ^2). As proposed by former research [27], with our n we can derive χ^2 in simple form.

$$\chi^2 = \binom{m}{2}[1 + u(n-1)] \quad (6)$$

Since we compare three codecs, the χ^2 distribution is evaluated with $\binom{m}{2} = 3$ DOF. Therefore, the statistical significance at level p derives by χ_3^2.

The analysis of the raw data for consistency and agreement is shown in Table 4. While the participants show a high consistency, the coefficient of consistency ζ for the *Quality* condition is considerably lower than for the *Speed* condition. The difference in agreement is even more pronounced and the participants show clear disagreement in the ratings of *Quality* (numbers highlighted in red). This mismatch is most apparent in the city scene where results are not statistically significant anymore for the targeted threshold of $p < .01$.

A more in-depth analysis of the consistency of individual ratings shows the cause for this outcome. In the *Quality* condition, a total of eight participants were not able to properly distinguish between stimuli, leading them to produce triads. Also, two participants had difficulties emitting judgement in the *Speed* condition. All the rest of our participants show perfect judgement capability ($\zeta = 1$, no triads), indicating that the inconsistencies arised from the individual participant's ability to judge and not from a problem with a consensus between participants. Therefore, we correct the analysis for a consistent result and remove the votes for the entire condition of all those participants who did not show perfect consistency. By rerunning the analysis with the votings corrected for perfect consistency ($n_{Speed} = 21, n_{Quality} = 15$) we can obtain the values corresponding to participants with perfect discerning abilities. These results are show in the bottom half of Table 4 ("Corrected for Consistency"). While the *Quality* condition still seems to be more controversial, the agreement of the participants for this condition increases substantially (numbers highlighted in green). With the correction, all differences in preference are statistically significant for $p < 0.01$.

In general, the results of the perceptual study indicated that the videos using both our codec and HEVC were favored over AV1 for all conditions. The conclusions of the comparison between our codec and HEVC are dependent on the analyzed condition. While the participants clearly favoured our codec in the *Speed* condition, the perceived quality overall is on par with the H.265 compressed video. A more in-depth analysis reveals the choice of scene as a decisive factor for the participants' preference. Straight lines and geometric structures are key attributes of urban scenery. In these representations with man-made content, quality differences and artifacts seem to be more conspicuous. The *Mountain* scene, on the other hand, is a natural scene and, as such, its content is less structured and highly semantically homogeneous due to the recurring textures. In this scene, the participants showed difficulties to distinguish the quality of the different compression techniques. These observations are consistent with the findings of previous research [4, 25]. In the *Speed* condition, the scene attributes seem to became even more salient for the observers' choice, drifting their preferences towards our method (preference $> 80\%$).

5 Discussion and Limitations

In the following, we discuss further key points of consideration and address limitations of the current implementation

Experimental Results. We compared our wavelet-based codec against two common video codecs and previous work. For the evaluation, we considered a low and high quality version of the wavelet-encoded videos, because in a practical application either quality or speed may be prioritised. The results show that the codec can be optimised for such requirements by changing the encoding parameters. However, even at the highest quality, we achieve significantly higher decoding speeds than the other methods. The foveated decoding technique leverages the properties of the human visual system, resulting in peripheral resolution differences that are unnoticeable to VR users compared to fully resolved viewports [19]. At the same time, the foveated decoding allows for the highest decoding speeds. In a perceptual experiment, we studied the importance of the frame rate and video resolution for the perceived quality of the VR experience. The results suggest that for 360° videos in VR, the frame rate is of significantly greater importance to users than the image quality. This emphasis on frame rate for the user experience is consistent with former research [1].

Single Operation Point. Our codec is designed for a very specific use-case. The core motivation is to have 360° videos with a resolution and display speed that does not induce cybersickness and is pleasant to watch. In our investigations, we explored in detail the single operation point that best covers this scenario. For a fair comparison we chose the parameters of our codec so that the quality is on the same level. With this baseline, we then measured the speed of the methods. A broader range of quality-rate scenarios can be explored in the future to utilize wavelet-based coding for a verity of applications.

Streaming. So far, we have primarily addressed videos that are stored on a local drive. Online streaming is another common way to retrieve video data. With online streaming, the amount of data that is transmitted is much more relevant due to bandwidth limitations. For these limitations, a wavelet-based codec benefits from the direct viewport-dependent streaming from file. This property allows to reduce the transfer rates by up to ten times compared to the total size of the video. In its current form, our encoding is not yet optimized for real time execution, which we plan to address as a natural next step. Especially for standalone HMDs, over-air transmission latencies and data transfer rates are often a bottle-neck of the system. In this scenario it would be beneficial to move the decoding of the wavelet videos to be performed by the HMD hardware. The encoded version of the video is then streamed and decoded on the HMD, thus, reducing the needed bandwidth and transfer time. Given the current hardware development, decoding times can be expected to be inferior to a dedicated consumer GPU. Nevertheless, with further improvements of standalone VR HMDs, an on-device decoding will become highly interesting for cloud-operated video streaming.

File Size. Our wavelet format does not use any container format but is stored in simple binary form. Neither is a color transformation performed, for example to the YUV space. Such techniques are applied by other codecs to reduce their file sizes to the minimum while preserving the best possible quality. In this paper the major focus was on display speed. In future work such techniques may be introduced to further reduce the file sizes of wavelet-based video coding.

Reference Data. Our objective with the reference data scaling of the real-world videos was to generate uncompressed high-resolution, high frame-rate video data. We used a combination of downscaling followed by AI-based upscaling to remove compression artifacts from the original videos. This removal is not perfect and it can be assumed that the compression rate of a wavelet-based codec is significantly higher for raw footage. Such a use of a wavelet-based codec can only be achieved when the encoding is directly performed by the capturing device with the native color information.

Professional Filming. The videos from our experiment are considered as casual recordings. Nevertheless, 360° videos are not only used by amateurs, but also by professional filmmakers. For professional filming, it can be necessary to display different areas of a frame in different qualities, such as the background or the masks of an actor, which stands out as artificial in high resolutions. With conventional methods, this procedure requires post-processing or recapturing of the video. With a wavelet-based codec a pre-adjustment is not necessary and the video can be stored in full resolution. Individual quality levels may be chosen at decoding time for defined parts of the video, comparable to our foveated decoding approach (cf. Sec. 3.7).

Eyetracking. In VR, eye tracking is nowadays mostly used for computer-generated content, where foveation allows for significant increases in rendering speed. The foveated decoding of our codec opens up the opportunity for an broader use of eye tracking in VR where it can be used to increase the playback speed of 360° videos through unobtrusive quality gradation in the peripheral area.

Wide FOV. When the FOV gets unusually wide, this would affect the performance of our approach since the decoding is viewport dependent. The headset with the widest FOV currently on the market is the *StarVR One* with an overall horizontal FOV of 210° and a vertical FOV of 130° [13]. Exploring this scenario, we found that with wavelet coding we are still able to achieve > 100 FPS with the high quality configuration in all scenes. With foveated decoding applied, the frame rate is significantly higher. In comparison, the tiling approach with this wide FOV no longer yields any performance benefit and actually performs worse than the native full frame HEVC decoding (≈ 50 FPS).

6 Conclusion

In this paper we proposed wavelet-based video coding for fast and high-resolution playback of 360° videos. We showed that our wavelet-based compression approach allows for selective loading and decoding of arbitrary video regions, which in the case of 360° videos is key for a fast decoding. While in our experiment our codec reached display speeds at least two times higher than the other methods tested, the quality remained at a comparable level. The importance of high frame rates for a good VR experience is supported by the results of our perceptual experiment. In addition, with our codec we have introduced foveated decoding, allowing for an unobtrusive quality decrease in the outer regions of the view. Foveated decoding can be applied on run-time and further increases the decoding times. In conclusion, wavelet-based video approaches solve the problems that are raised by DCT codecs when a fast or viewport-dependent playback of 360° videos is required. Especially for VR environments, wavelet-based codecs show to be a valuable extension, offering the opportunity to display 360° videos in a quality and speed comparable to renderings of virtual worlds.

7 Acknowledgments

The authors gratefully acknowledge funding by the German Science Foundation (DFG MA2555/15-1 “Immersive Digital Reality”), and under Germany’s Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122, Project ID 390833453).

References

[1] A. Banitalebi-Dehkordi, M. T. Pourazad, and P. Nasiopoulos. The effect of frame rate on 3d video quality and bitrate. *3D Research*, 6(1):1–13, 2015. 8

[2] G. Bjontegaard. Calculation of average PSNR differences between RD-curves. *ITU-T SG16/Q6 input document VCEG-M33*, 2001. 2

[3] G. Boopathi and S. Arockiasamy. Image compression: Wavelet transform using radial basis function (rbf) neural network. In *India Conference*, pp. 340–344. IEEE, 2012. 3

[4] S. Castillo, T. Judd, and D. Gutierrez. Using eye-tracking to assess different image retargeting methods. In *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization*, APGV ’11, pp. 7–14, 2011. doi: 10.1145/2077451.2077453 8

[5] B. Choi, Y. Wang, M. Hannuksela, Y. Lim, and A. Murtaza. Information technology–coded representation of immersive media (mpeg-i)–part 2: Omnidirectional media format. *ISO/IEC*, pp. 23090–23092, 2018. 2

[6] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45(5):485–560, 1992. doi: 10.1002/cpa.3160450502 2, 3

[7] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *International Conference on Communications*, pp. 1–7. IEEE, 2017. 2

[8] C. Groth, J.-P. Tauscher, N. Heesen, S. Grogorick, S. Castillo, and M. Magnor. Mitigation of cybersickness in immersive 360°videos. In *IEEE Virtual Reality Workshop on Immersive Sickness Prevention (WISP)*, pp. 169–177. IEEE, 2021. doi: 10.1109/VRW52623.2021.0 5

[9] C. Groth, J.-P. Tauscher, N. Heesen, M. Hattenbach, S. Castillo, and M. Magnor. Omnidirectional galvanic vestibular stimulation in virtual reality. *Transactions on Visualization and Computer Graphics (TVCG)*, 28(5):2234–2244, 2022. doi: 10.1109/TVCG.2022.315 5

[10] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 71(1):38–53, 1911. 3

[11] M. M. Hannuksela, Y.-K. Wang, and A. Hourunranta. An overview of the omaf standard for 360 video. In *Data Compression Conference*, pp. 418–427, 2019. 2

[12] W. Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, 43:172–198, 1927. doi: 10.1007/BF01397280 3

[13] Starvr one. `https://www.starvr.com/`. Accessed: 2022-10-14. 8

[14] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou. Rife: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*, 2021. 6

[15] ISO. *ISO/IEC 15444-1:2019*, vol. 642. International Organization for Standardization, 2019. 2, 3

[16] M. G. Kendall and B. Babington-Smith. On the method of paired comparisons. *Biometrika*, 31:324–345, 1940. doi: 10.1093/biomet/31.3-4.324 7

[17] H. Kolb, R. F. Nelson, P. K. Ahnelt, I. Ortuño-Lizarán, and N. Cuenca. The architecture of the human fovea. *Webvision: The Organization of the Retina and Visual System*, 2020. 5

[18] D. Le Gall and A. Tabatabai. Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 761–764, 1988. 2

[19] R. Leigh and D. Zee. *The Neurology of Eye Movements*. Contemporary neurology series. Oxford University Press, 2015. 5, 8

[20] Y. Leng, C.-C. Chen, Q. Sun, J. Huang, and Y. Zhu. Energy-efficient video processing for virtual reality. In *International Symposium on Computer Architecture (ISCA)*, pp. 91–103, 2019. 2

[21] D. Li, R. Du, A. Babu, C. D. Brumar, and A. Varshney. A log-rectilinear transformation for foveated 360-degree video streaming. *Transactions on Visualization and Computer Graphics*, 27(5):2638–2647, 2021. 5

[22] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of jpeg-2000. In *Proceedings Data Compression Conference*, pp. 523–541. IEEE, 2000. 2, 4

[23] M. Mühlhausen, M. Kappel, M. Kassubeck, P. M. Bittner, S. Castillo, and M. Magnor. Temporal consistent motion parallax for omnidirectional stereo panorama video. In *ACM Symposium on Virtual Reality Software and Technology (VRST)*, 2020. doi: 10.1145/3385956.3418965 5

[24] M. Orchard and G. Sullivan. Overlapped block motion compensation: an estimation-theoretic approach. *Transactions on Image Processing*, 3(5):693–699, 1994. 2

[25] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir. A comparative study of image retargeting. *ACM Transactions on Graphics, Proceedings Siggraph Asia*, 29(5), 2010. 8

[26] I. Setyawan and R. L. Lagendijk. Human perception of geometric distortions in images. In *Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, pp. 256 – 267, 2004. doi: 10.1117/12.526726 7

[27] S. Siegel and J. Castellan, N. J. *Nonparametric statistics for the behavioral sciences*. Mcgraw-Hill Book Company, 2. ed., 1988. 7

[28] L. D. Silverstein. Foundations of vision. *Color Research and Application*, 21:142–144, 2008. 5

[29] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In *International Symposium on Multimedia*, pp. 583–586, 2016. 2

[30] J.-P. Stauffert, F. Niebling, and M. E. Latoschik. Latency and cybersickness: impact, causes, and measures. a review. *Frontiers in Virtual Reality*, 1:1–10, 2020. 4

[31] Q. Sun, A. Taherin, Y. Siatitse, and Y. Zhu. Energy-efficient 360-degree video rendering on fpga via algorithm-architecture co-design. In *International Symposium on Field-Programmable Gate Arrays*, pp. 97–103, 2020. 2

[32] D. Taubman and M. Marcellin. *JPEG2000 image compression fundamentals, standards and practice*, vol. 642. Springer Science & Business Media, 2012. 2, 4

[33] BBC Research. Dirac specification (version 2.2.3). `https://web.archive.org/web/20150503015104/http://diracvideo.org/download/specification/dirac-spec-latest.pdf`, 2008. 2

[34] M. Unser and T. Blu. Mathematical properties of the jpeg2000 wavelet filters. *IEEE Transactions on Image Processing*, 12(9):1080–1090, 2003. doi: 10.1109/TIP.2003.812329 3, 4

[35] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861 6

[36] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Hevc-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the International Conference on Multimedia*, pp. 601–605, 2016. 2, 5, 6

[37] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068 6

[38] S. Zhao, H. Zhang, S. Bhuyan, C. S. Mishra, Z. Ying, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das. Déjà view: Spatio-temporal compute reuse for energy-efficient 360° vr video streaming. In *International Symposium on Computer Architecture (ISCA)*, pp. 241–253, 2020. 2