Wavelet-Based Fast Decoding of 360° Videos

COLIN GROTH, SASCHA FRICKE, SUSANA CASTILLO, and MARCUS MAGNOR, Institute for Computer Graphics, TU Braunschweig, Germany

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 190 frames per second at 8192x8192-pixel full frame resolution, the evaluations demonstrate 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. Finally, we demonstrate how our wavelet-based codec can also directly be used in conjunction with foveation for further performance increases.

CCS Concepts: • **Computing methodologies** → **Image compression**; **Virtual reality**; Visibility; Perception.

Additional Key Words and Phrases: video compression, wavelet, virtual reality, 360 videos, foveated rendering

1 INTRODUCTION

Codecs provide efficient compression that allows storing 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 discrete cosine transform (DCT) and motion compensation for high compression rates at a reasonable perceptual quality. The high compression, however, requires complex techniques. 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 his view anywhere, since the information of the entire space is available (3 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 when prerecorded video data is presented. 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 structure. Accordingly, with 360° videos it is frequently practised to load, upload, and decode the entire 360° frame from the drive, 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 today do not come close to the quality of renderings of virtual environments. Although some ideas exist that already try to improve the compression, e.g. tiling the frame into separate regions, these ideas often go against the basic compression idea 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 FOV of VR glasses. (2) The encoding takes place 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 to compress imagery were limited 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 we provide a comparable video quality and reasonable compression rate. In addition, we introduce foveated decoding. With foveated decoding the properties of the wavelets are utilised to gradually decrease in resolution with the distance from the focal point. Such a foveation is so far only known from virtual scenes and offers further opportunities for decoding speed and image perception. Our code will be made publicly available upon acceptance.

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 frequently practised, as it reduces the need to adjust the standard video pipeline. We first discuss the works that aim at a more resource-efficient presentation by adapting existing codecs. In the second part we introduce former attempts for wavelet-based codecs.

2.1 Viewport-Adaptive Display Techniques for Videos

Zare et al. [2016] 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 requests 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 most bitrate savings (-40% based on BD-BR). However, the compression losses also increase the more tiles are used since all tiles are saved independently.

The tiling idea 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 [Choi et al. 2018]. The idea of OMAF is comparable to the work of Zare et al. 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 [Hannuksela et al. 2019].

Sreedhar et al. [2016] also recognized the technical challenges of bandwidth associated with high-resolution 360° videos. The main focus of their work are 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. [2017] 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.

2.2 Wavelet based codecs

Probably the best known use of wavelets for imagery is the JPEG2000 image compression standard [Marcellin et al. 2000; Taubman and Marcellin 2012]. At the turn of the millennium, it initiated a new form of image compression and was supposed 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 for lossy compression or the LeGall-Tabatabai (LGT) 5/3 wavelet for lossless compression. The standard has four levels of decomposition as a default since there is not much improvement in using higher decomposition levels when compressing images [ISO 2019]. 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.

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 [BBC Research 2008]. 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 [Orchard and Sullivan 1994]. 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.

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* utilises redundancies between multiple frames to reduce the data size. In the following we describe how we realised both concepts with wavelet transforms.

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 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:

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

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 [Unser and Blu 2003]. As usual for the FWT, with every step of the 2D wavelet transform 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 [Boopathi and Arockiasamy 2012]. For the frame-wise transformation of the image we use the CDF 9/7 wavelet [Cohen et al. 1992]. The CDF wavelet is known to perform especially good on natural imagery and is also used for lossy compression in the JPEG2000 standard [ISO 2019].

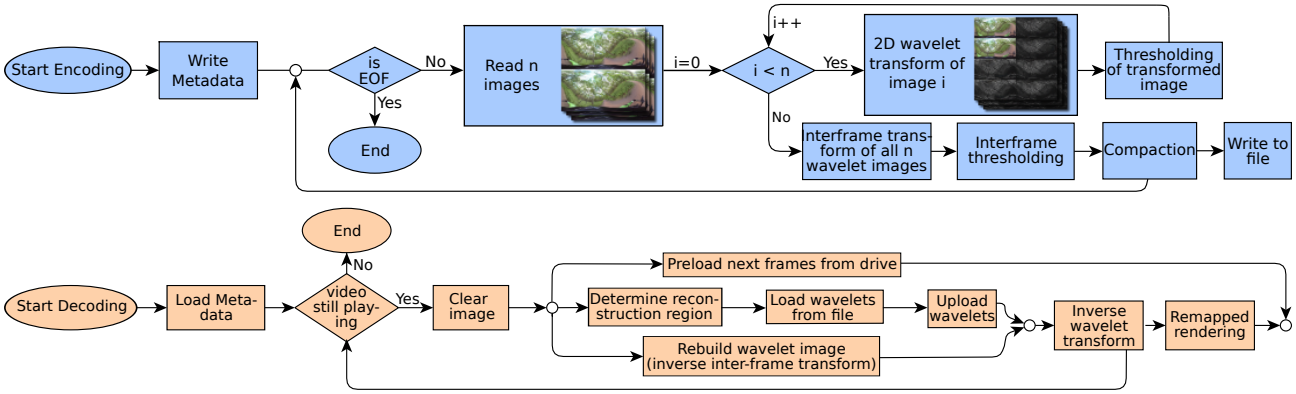

Fig. 1. Our program flow for encoding (blue) and decoding (orange) a video with our wavelet-based codec.

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 is 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.

In theory, the transform can be performed until only one pixel defines the 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 [Heisenberg 1927]. 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. Therefore, we only perform the wavelet transform until level $l_{max} = 6$ as a default.

3.2 Inter-Frame Coding

Inter-frame coding describes the compression of temporal information. The time component t is represented implicitly by a set of successive frames. In modern codecs the inter-frame compression is performed with keyframes encoding only information differences with the help of motion vectors. 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 wavelet transform to encode 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 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 is called *inter-frame set*. Thereby, n is a power of two values. 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 [Haar 1911]. 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 scaled by the high pass filter position (−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. 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. 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:

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

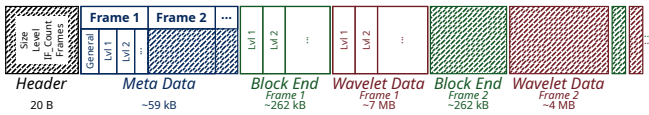

Fig. 2. Data arrangement of our video format. The sizes of each section are given by an example video in 8k resolution.

where H specifies a mapping factor which depends on the mapping technique. 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 defined by 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 [Unser and Blu 2003]. We scale the threshold by the frequency level of each coefficient in a quadratic function (cf. Equ. 3). Accordingly, more coefficients may be zeroed out at high frequencies. This thresholding weighting follows common procedures of other codecs like JPEG2000 [Marcellin et al. 2000; Taubman and Marcellin 2012].

Quantization: Similar to other codecs, we represent the colour values of the pixels in the video file by one byte per colour component. In the quantization, the 32 bit float colour 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 normalisation in order to achieve the highest possible spatial resolution in this discretization. Therefore, one discrete colour value c_d is defined by $c_d = (c_n - c_{min})/(c_{max} - c_{min}) * 255$ 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 normalisation 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 normalisation 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 Fig. 2. 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.

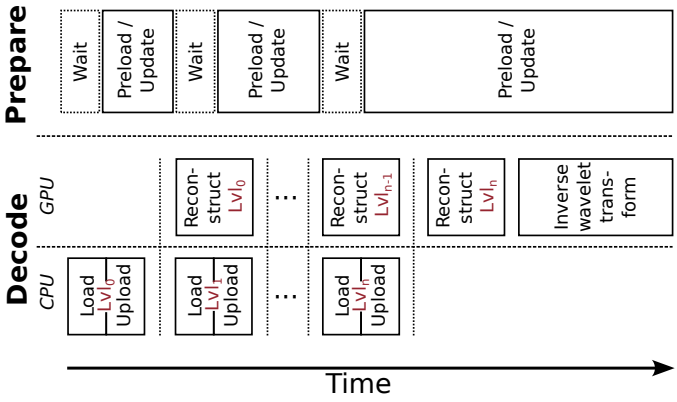

Fig. 3. Parallel processes for the reconstruction of one frame.The prepare thread buffers the next frames, but does not read from the drive in the *wait* sections for the decoding process to be faster.

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 32x32-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 Fig. 2). 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. On the other hand, without buffering the data has to be loaded from the drive at the time of rendering which is costly and slows down the process. Therefore, we preload the data of subsequent frames by estimations of the next head and eye positions. Thereby, we estimate the trajectory of the head and eye movement by the movement pattern of the previous positions. The data of the subsequent inter-frame set is preloaded

Fig. 4. Our foveated decoding compared with the full-resolution reference frame. On the right, the information density of the foveated decoding is visualised.

in parallel to the rendering of the current frames. Until the time of rendering, the preloaded data is updated continuously in case that the estimations differ from the actual movements.

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 prepare and upload the wavelet data for the next inter-frame in parallel (see Fig. 3). This rebuild already includes the inverse inter-frame transform, as described in Sec. 3.2. Also, the wavelet transforms for encoding and decoding are executed on the GPU. 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.

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.

Table 1. Display speeds. Tiling refers to Zare et al.. 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 frames per second (FPS).

Videos	AVG fps ↑					
	HEVC	Tiling	AV1	$Ours_{LQ}$	$Ours_{HQ}$	$Ours_{FD}$
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**

3.7 Foveated Decoding

The information density of visual representations of the human eye are not equally distributed [Silverstein 2008]. The images created on the retina of the human visual system follow a qualitative decline starting from the fixation point of the eye. 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 [Kolb et al. 2020].

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 wavelets and 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 a higher possible 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. The reconstruction is performed incrementally from a low-resolution version of the frame to the full resolution. Instead of reconstructing the same FOV at each level, the full FOV of the viewport is reconstructed only at the lowest frequency level and reduced with each level. The sizes of the individual resolution levels of the wavelet transform are defined in regard to the properties of the human eye [Leigh and Zee 2015]. Like human perception, we decrease the quality of the frames at a logarithmic rate [Kolb et al. 2020]. The area with the full video resolution which stimulates the most central foveola is only about two percent wide [Silverstein 2008].

The inverse wavelet transform is executed over the same number of data points as for a full resolution viewport but assumes zero coefficients for the surrounding regions. With foveated decoding, we achieve a peripheral reconstruction in a visually appealing quality with a small number of coefficients (see Fig. 4). 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 [Li et al. 2021].

4 EXPERIMENT

For the 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.

4.1 Dataset

For the evaluation, we analyse two categories of 360° videos. The first set of videos is recorded with a moving camera trajectory and the display of rapid motions. Here, we use the videos of Groth et al. [2021; 2022] which have a higher resolution than typical moving-camera 360° videos due to their 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. [2020]. All videos display natural, real-world scenes. We deliberately decided not to use rendered scenes, as we see the final application area to be real-world recordings. The original videos are recorded with stereoscopic information in 6400x6400-pixel resolution at 30 FPS.

Reference Data Creation: Pre-recorded videos can cause two problems for evaluation. For one, the frame rates typically do not match the refresh rates of VR devices. Additionally, the data is already lossy compressed and has partly serious compression artefacts. In order to address both problems, we first downscale the video data to 1024x1024-pixel to get rid of high-frequency compression artefacts and then perform temporal interpolation and upscaling of the data with state of the art (SotA) 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 [Huang et al. 2021]. The information from both eyes is processed individually to avoid artefacts 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 8196x8196-pixel resolution.

4.2 Evaluation

For the evaluation we compare our codec against the common HEVC and AV1 codecs and a tiled HEVC implementation [Zare et al. 2016]. 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 YUV420 colour space. The OMAF inspired tiling method is realised with HEVC encoded tiles with the fastest tiling scheme of Zare et al. [2016]. 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 with all experimental conditions, we use head and eye tracking data of participant recordings.

Table 2. Quality metrics of all codecs in comparison with the reference frames.

Scene	Metrics	HEVC	AV1	$Ours_{LQ}$	$Ours_{HQ}$
Downhill	PSNR ↑	34.77	34.17	31.4	**34.81**
	SSIM ↑	.954	.95	.921	**.961**
	LPIPS ↓	.082	.103	.161	**.081**
Horse	PSNR ↑	**37.05**	36.48	32.24	35.07
	SSIM ↑	.968	.966	.939	**.97**
	LPIPS ↓	**.054**	.07	.118	.057
Climbing	PSNR ↑	**37.3**	36.83	32.84	35.33
	SSIM ↑	.973	.971	.952	**.976**
	LPIPS ↓	**.051**	.066	.115	.056
Walking	PSNR ↑	**37.64**	37.06	32.04	35.45
	SSIM ↑	**.97**	.969	.928	.967
	LPIPS ↓	**.05**	.064	.117	**.05**
Cave*	PSNR ↑	**40.32**	40.24	37.02	39.58
	SSIM ↑	.971	**.972**	.96	**.972**
	LPIPS ↓	**.039**	.04	.08	.046
Boat*	PSNR ↑	39.2	**39.87**	33.01	35.88
	SSIM ↑	.984	**.986**	.954	.978
	LPIPS ↓	.03	**.027**	.078	.036

The results regarding **computational time** are shown in Tab. 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.

We compared the results' **quality** of all codecs by the commonly used metrics PSNR, SSIM [Wang et al. 2004], and LPIPS [Zhang et al. 2018]. The given values are averages over all frames and compared with the uncompressed reference frames (see Tab. 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

Table 3. Compression ratios of the wavelet video files in relation to the uncompressed data.

	Downhill	Horse	Climbing	Walking	Cave*	Boat*
$Ours_{LQ}$	147:1	187:1	250:1	185:1	714:1	312:1
$Ours_{HQ}$	77:1	100:1	128:1	100:1	416:1	117:1

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 can be seen in Tab. 3. With our wavelet-based approach, we are able to compress the raw information to over one hundredth in size for most videos. Compared to HEVC and AV1 compression we achieve about half the compression rates, depending on the quality of the video. The tiled HEVC videos by the technique of Zare et al. are on average three times larger than our wavelet-compressed video files due to the significant compression losses of the tiling process.

5 DISCUSSION AND LIMITATIONS

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 to a fully resolved viewport that are unnoticeable to the user [Leigh and Zee 2015]. Despite a perceived visual quality that is comparable to the highest quality videos, the foveated decoding allows for the highest decoding speeds. In VR, eye tracking is nowadays mostly used for computer-generated content, where foveation allows for significant increases in rendering speed. Our wavelet-based foveated decoding opens up the opportunity for an broader use of eye tracking in VR where it can also be used to increase the playback speed of 360° videos through unobtrusive quality gradation in the peripheral area.

Our objective with the reference data scaling 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 artefacts 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 colour information.

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).

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.

Our wavelet format does not use any container format but is stored in simple binary form. Neither is a colour 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 our wavelet-based video codec.

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. 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”).

REFERENCES

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

B Choi, YK Wang, MM Hannuksela, Y Lim, and A Murtaza. 2018. Information technology–coded representation of immersive media (MPEG-I)–part 2: Omnidirectional media format. *ISO/IEC* (2018), 23090–23092.

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

Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *International Conference on Communications*. IEEE, 1–7.

Colin Groth, Jan-Philipp Tauscher, Nikkel Heesen, Steve Grogorick, Susana Castillo, and Marcus Magnor. 2021. Mitigation of Cybersickness in Immersive 360°Videos.

In *IEEE Virtual Reality Workshop on Immersive Sickness Prevention (WISP)*. IEEE, 169–177. https://doi.org/10.1109/{VRW}52623.2021.00039

Colin Groth, Jan-Philipp Tauscher, Nikkel Heesen, Max Hattenbach, Susana Castillo, and Marcus Magnor. 2022. Omnidirectional Galvanic Vestibular Stimulation in Virtual Reality. *Transactions on Visualization and Computer Graphics (TVCG)* 28, 5 (2022), 2234–2244. https://doi.org/10.1109/{TVCG}.2022.3150506

Alfred Haar. 1911. Zur theorie der orthogonalen Funktionensysteme. *Math. Ann.* 71, 1 (1911), 38–53.

Miska M Hannuksela, Ye-Kui Wang, and Ari Hourunranta. 2019. An overview of the OMAF standard for 360 video. In *Data Compression Conference*. 418–427.

Werner Heisenberg. 1927. Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. (1927), 172–198.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. 2021. RIFE: Real-Time Intermediate Flow Estimation for Video Frame Interpolation. *arXiv preprint arXiv:2011.06294* (2021).

ISO. 2019. *ISO/IEC 15444-1:2019*. Vol. 642. International Organization for Standardization.

Helga Kolb, Ralph F Nelson, Peter K Ahnelt, Isabel Ortuño-Lizarán, and Nicolas Cuenca. 2020. The architecture of the human fovea. *Webvision: The Organization of the Retina and Visual System* (2020).

R.J. Leigh and D.S. Zee. 2015. *The Neurology of Eye Movements*. Oxford University Press.

David Li, Ruofei Du, Adharsh Babu, Camelia D Brumar, and Amitabh Varshney. 2021. A log-rectilinear transformation for foveated 360-degree video streaming. *Transactions on Visualization and Computer Graphics* 27, 5 (2021), 2638–2647.

Michael W Marcellin, Michael J Gormish, Ali Bilgin, and Martin P Boliek. 2000. An overview of JPEG-2000. In *Proceedings Data Compression Conference*. IEEE, 523–541.

Moritz Mühlhausen, Moritz Kappel, Marc Kassubeck, Paul Maximilian Bittner, Susana Castillo, and Marcus Magnor. 2020. Temporal Consistent Motion Parallax for Omnidirectional Stereo Panorama Video. In *Symposium on Virtual Reality Software and Technology (VRST)*. ACM, 1–9. https://doi.org/10.1145/3385956.3418965

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

Louis D. Silverstein. 2008. Foundations of Vision. *Color Research and Application* 21 (2008), 142–144.

Kashyap Kammachi Sreedhar, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2016. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In *International Symposium on Multimedia*. IEEE, 583–586.

David Taubman and Michael Marcellin. 2012. *JPEG2000 image compression fundamentals, standards and practice*. Vol. 642. Springer Science & Business Media.

BBC Research. 2008. Dirac Specification (Version 2.2.3). https://web.archive.org/web/20150503015104/http://diracvideo.org/download/specification/dirac-spec-latest.pdf.

M. Unser and T. Blu. 2003. Mathematical properties of the JPEG2000 wavelet filters. *IEEE Transactions on Image Processing* 12, 9 (2003), 1080–1090. https://doi.org/10.1109/TIP.2003.812329

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

Alireza Zare, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the International Conference on Multimedia*. 601–605.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 586–595. https://doi.org/10.1109/CVPR.2018.00068