

## Virtual Reality Video Compression

### Overview of VR and its Applications

Virtual reality (VR) is a virtual, or simulated, experience that attempts to take a user out of their environment and transplant them into a virtual environment. Much like traditional video, VR usually involves the sense of sight and sound, however many VR experiences also let the user interact with the simulated environment through movement and touch. Most VR today involves the user wearing some type of device over their eyes. This device is usually a head-mounted display that is tethered to a powerful desktop PC or allows for a smartphone to be placed within the head-mount so that the smartphone screen can act as the display. The main theme is that the user can no longer see anything around them in the real world so that their brain can be tricked into believing this virtual environment is real.



Figure 1: Virtual reality head-mounted displays (wsj.com)

Some of the main purposes of VR today are video games, teaching and learning, remote work, entertainment, and communication. Video games are what is often associated with VR and is an obvious medium for the technology to be explored. Gaming has always been a form of providing the user with a virtual experience where they can escape the restrictions of reality so it's a natural fit for VR.

The field of teaching and learning through VR has a lot of potential and will likely play a major role in the way that future generations learn. VR can provide the student with experiences that they would otherwise not be able to have access to due to financial reasons, risk, or the difficulty of creating the situation in the real world. Flying a plane

can be dangerous and very expensive, but by practicing in a flight simulator, a student or experienced pilot can gain familiarity for a lot less risk and cost. A surgeon can practice different surgery techniques within a VR environment before having to perform an operation on a live patient. Members of the military can simulate war zone situations from the safety of a virtual environment. Even general studies for people of all ages can be augmented and made more immersive and captivating by providing the lesson within one of these environments.



Figure 2: Boeing 737 VR flight simulator ([virginexperiencedays.co.uk](http://virginexperiencedays.co.uk))

Remote work is one of the lesser considered applications of VR, but has a lot of potential. Imagine that you have some heavy machinery that needs to be controlled by a highly skilled worker to perform a task. This worker may be hundreds or thousands of miles away so it will take time and money to bring them to the machinery. With VR, instead of traveling to the site, the worker could control the machinery and perform the task from an office or even their own home. This introduces a latency issue so it's not a foolproof option, but it's very interesting and worth further study.

Much like video games, movies and television have long been a way for a viewer to transport themselves to another reality. VR can help make that experience more immersive and real. VR could also be used to allow a viewer to feel like they are at a live event from the comfort of their own home. The viewer could walk into their living room, put on their VR headset and immediately feel like they are at a live concert. This means that a band or event could reach much larger audiences due to no longer being limited by the amount of space in the venue. It would also allow for people to experience the event from anywhere on the planet. The viewer could also choose any "ticket" in the venue based on where they want to view the event from. You could have millions of viewers all sitting in the best seat in the house.



Figure 3: Prototyped in 1956 and patented in 1962, the Sensorama was one of the earliest virtual reality machines. It included full colour 3d video, audio, vibrations, smell, and even wind (<https://virtualspeech.com/blog/history-of-vr>)

The telephone and the internet were massive advances with regards to how humans can communicate with each other. VR is poised to be the next leap. By providing avatars for each user and an environment for the avatars to interact with each other, VR can provide a much more immersive and “real” communication experience. This could be a video conference for work, futuristic online dating, a games night with friends, or just general social interaction. Many VR developers are working on technologies to allow for a translation of facial expressions or body language onto the avatars so that people can better express themselves.

The potential of VR is enormous and could change the world in a lot of very meaningful ways. The main challenges with making this a reality are computing power, internet bandwidth, and developers creating these environments. Computing power and internet bandwidth are physical and financial limitations that aren’t very easy to solve. However, we can limit the amount of bandwidth required if we can make the files smaller and we can limit the amount of computing power required if we use or design more efficient coding algorithms. This is where compression comes in.

## Why Compression is Important

Before discussing compression techniques, it is important to understand why compression is important. VR video compression uses much the same technology and techniques as 360 degree video streaming, which has considerably more publicly available research. Therefore, 360 video will be the major focus of this paper. Examples of 360 video streaming with tight constraints are videoconferencing and live event streaming; the user experience relies heavily on being able to receive the data fast enough to keep up with realtime. Streaming a live event could get away with a large buffer as most user experiences wouldn’t be affected by



Figure 4: A VR chatroom where people from all around the world can interact with each other (bizjournals.com)

being around 30 seconds behind realtime. However, this would just allow for bandwidth fluctuation and would still require an average transmission rate high enough to maintain the buffer. Videoconferencing on the other hand would suffer greatly by having a buffer of even a couple seconds. In either case, it's clear that we need to minimize the size of the files being streamed.

360 degree video is different from 2D video in that instead of covering a limited plane, it must cover the entire 360x180 degree viewing range. Because of this, the minimum resolution required for an image or video to not compromise the user experience is much higher. Most experts consider the minimum resolution for an immersive VR experience to be 4K (3840x1920, 3840x2048, or 4096x2048). Without modern compression techniques, streaming this amount of data wouldn't be possible. Fortunately, we already have very advanced, well researched, and well tested compression and coding algorithms for 2D video.

Some of the most popular codecs today include H.264, H.265, VP9, and AV1. The compression used in all of these codecs is considered lossy. This means that we cannot reconstruct the original source information from the compressed data and so there will be a degradation of quality. This is a necessary evil as lossless compression techniques have compression ratios which are magnitudes lower than that of the lossy codecs. Fortunately, a lot of very intelligent and clever people have been working on this topic for at least 40 years and they've come up with some brilliant and effective techniques to reduce file size without sacrificing too much in the way of video quality.

So why can't we just use a codec like H.265 for our 360 degree videos? Well, we could and we do, but there are some very important differences between 2D and 360 degree video that result in the compression techniques not translating as well from one to the other. What we really want is to be able to make modifications and additions to the current codecs so that they can work more efficiently with 360 degree video.

In most cases, 360 degree video frames are projected onto a 2D plane so that we can use modern 2D codecs. This projection can introduce some important problems: geometric distortion, redundancy, and discontinuity. Geometric distortion and redundancy occur when the projection from a panoramic scene to a planar scene causes large deformation in the pole area [1] [2]. This is caused by non-uniform sampling density. The proposed solutions to this problem mainly deal with motion estimation and motion compensation. Discontinuity occurs because 360 degree scenes are spherical which means they have no borders however they are being mapped to a 2D plane which has borders [3]. We'd also like to be able to view these streams on a smartphone so the decoding process can't require too much computation. These problems result in the current wave of codecs being suboptimal for 360 degree video compression. What we want from a compression scheme is for there to be a high compression ratio (new file size / original file size) while maintaining a high quality playback experience.

For the rest of the paper, we will discuss many of the proposed solutions to the issues mentioned in the previous paragraph. Many of the concepts addressed and explored in this paper were brought to our attention from reading [4] (“State-of-the-art in 360 video/image processing: perception, assessment and compression”) and many of the sources that were referenced in [4]. If you wish to explore some of the topics further, [4] would be a great place to start.





Projection	Illustration
Equirectangular Projection (ERP)	
Cubemap Projection (CMP)	
Truncated Square Pyramid Projection (TSP)	
Craster Parabolic Projection (CPP)	

Figure 5: Different types of projections from 360 degree video to 2D planes [4]



## Motion Estimation Adaption

Motion estimation is a compression technique that has been around for many generations of coding algorithms. Each generation has added some complexity and further optimization, but the general idea is that instead of encoding each frame independently as an image, we can encode the differences between subsequent frames. More specifically with regards to motion estimation and motion compensation, we can split the frames up into blocks and encode how far and in what direction each block of pixels moved between frames. It is an important step in removing or reducing temporal redundancy.

Traditional motion estimation hinges on two assumptions. Assumption 1 is that an object in motion within a frame can be represented by block translations in the 2D plane and its related motion vector [4]. Assumption 2 is that padding can replicate pixels near borders in the event that motion estimation uses any samples outside of the borders. Neither assumption 1 nor assumption 2 hold true for 360 degree video. Non-linear motion such as rotation and zoom brought in by geometric distortion are beyond the grasp of traditional motion estimation. Also, the variance of motion vectors is generally much harsher in 360 degree video which results in more bits being required to encode [2]. Due to 360 degree video not having true borders, the padding mentioned in assumption 2 is not appropriate and can lead to serious quality degradation [5].

## Motion Estimation Before Projection

One thought that has been proposed and built upon by many researchers is to perform the motion estimation within the spherical domain rather than doing it on the 2D projected plane. We have read through several of the papers exploring this idea, but the one that makes the most sense to us is “Spherical Coordinates Transform-Based Motion Model for Panoramic Video Coding” by Y. Wang et al. [6]. This paper expresses the idea that capturing motion between frames is the most important piece of compression in video. The current techniques for encoding this motion in 2D video do not translate well to 360 degree (panoramic) video so coming up with solutions better suited for panoramic video will have significant implications to the future of VR.

“Thus, we propose a new motion model based on spherical coordinates transform, termed SCTMM, for panoramic video coding. The basic assumption of SCTMM is that a block (in the mapped video is corresponding to a portion of sphere (in the panoramic video), and that portion of sphere has a translation in the 3D space. It then needs three parameters to describe that 3D translation.” [6] The type of projection that is focused on in the SCTMM paper is equirectangular projection (ERP) which is shown in Figure 5. Different projection schemes were analyzed, but all of them suffer from the same deformation between spherical and planar frames. The goal is to come up with a new motion system that will still be easy to integrate into existing coding schemes.



Figure 6: How 3D translation can deform a frame significantly with ERP [6]

Spherical coordinates transform is a fundamental geometric transform and can be written as:

$$\begin{aligned} x &= R \cos(g) \sin(f) \\ y &= R \sin(g) \sin(f) \\ z &= R \cos(f) \end{aligned}$$

where  $R$  is radius (depth),  $f$  is latitude, and  $g$  is longitude. This allows for a simple translation between a point in a cartesian coordinate system  $(x, y, z)$  and the same point in a spherical coordinate system  $(R, f, g)$ . The equations to convert a point on a sphere to a corresponding point in an equirectangular projection are:

$$\begin{aligned} f &= \pi(v/H) \\ g &= 2\pi(u/W) \end{aligned}$$

where  $(f, g)$  are still latitude and longitude,  $(H, W)$  are the height and width of the rectangle and  $(u, v)$  are the coordinates within the rectangle. Based on these two sets of equations, you can convert from  $(x, y, z)$  to  $(u, v)$  with the following,

$$\begin{aligned} u &= \frac{W}{2\pi} \cos^{-1}\left(\frac{x}{\sqrt{x^2+y^2}}\right) \text{ for } y > 0 \\ u &= \frac{W}{2\pi} \left(W - \cos^{-1}\left(\frac{x}{\sqrt{x^2+y^2}}\right)\right) \text{ for } y \leq 0 \\ v &= \frac{H}{\pi} \cos^{-1}(z/R). \end{aligned}$$

however  $R$  (depth or distance from the camera) is not known due to that information not being captured by the camera. Fortunately, this doesn't prove to be a big deal as in the paper they are able to show that the value of  $R$  is not important if you can make the assumption that the depth inside a small block of pixels is constant. This is a fair assumption to make for small enough blocks.

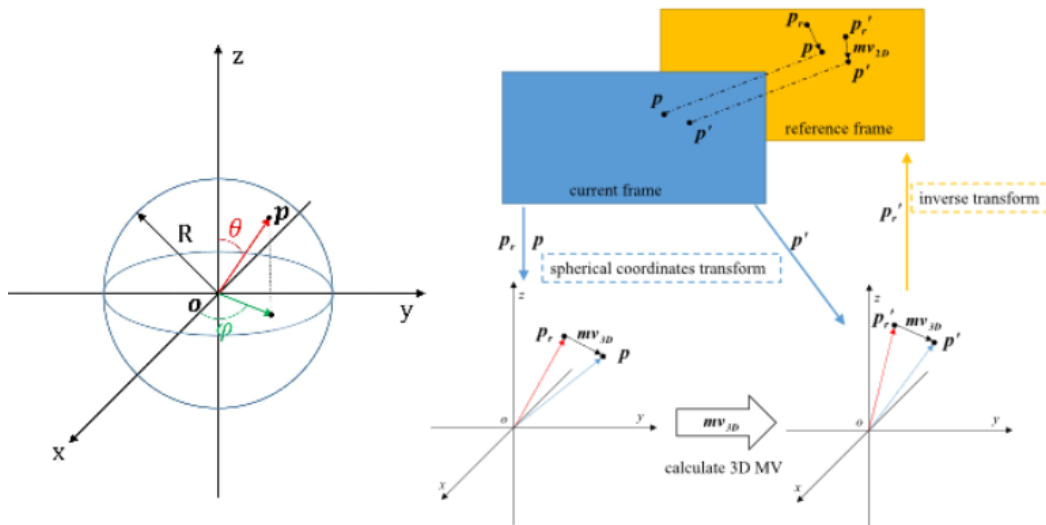


Figure 7: Spherical coordinates transform (left) and the spherical coordinates transform-based motion model (right) [6]

By implementing SCTMM into existing compression systems, the authors of this paper were able to produce an average of 2.6% and up to 10.7% BD-rate reduction compared to a baseline.

## Padding Method Adaption

No matter which type of projection that is being used, the frames are still going from a sphere to a plane. Unlike 2D planes, spheres are continuous and do not have borders or boundaries. Because modern video codecs were designed and optimized for planar video, they have techniques to deal with the boundaries. These techniques do not translate well to spherical projections so Y. He et al. wrote the paper ‘‘Motion compensated prediction with geometry padding for 360 video coding’’ as a proposal for a new way to deal with these boundaries [3].

Padding is necessary for when a reference block for a frame is outside of the frame’s boundary. The conventional way to handle this in modern codecs is for the reference sample to be determined by repeating the samples at the edge of the frame boundary.





Figure 8: Conventional repetitive padding being applied to an equirectangular projection [6]

Looking at Figure 8 and knowing that the true nature of this image is spherical, we can reason that the left edge should naturally connect with the right edge. Therefore, repeating the edges doesn't make much sense when dealing with a continuous, spherical frame. Intuitively, when determining what lies outside of the boundary, we should just start at the opposite boundary and move inwards. This is the logic behind the geometry padding proposed in [6]. For an equirectangular projection such as the one shown in Figure 8, it is a fairly straightforward calculation. If we have a point  $(x, y)$  that lies outside of the ERP frame, we will need to find a point  $(x', y')$  to act as the padded point. The calculation is as follows (directly from [6]),

$$x < 0 \text{ or } x \geq W \text{ and } 0 \leq y < H :$$

$$x' = x \% W$$

$$y' = y$$

$$y < 0 :$$

$$x' = (x + \frac{W}{2}) \% W$$

$$y' = -y - 1$$

$$y \geq H :$$

$$x' = (x + \frac{W}{2}) \% W$$

$$y' = 2H - y - 1$$

where  $W$  and  $H$  are the width and height of the equirectangular projection. Comparing Figure 8 with Figure 9, it's easy to see how much more accurate the padding will be with geometry padding.

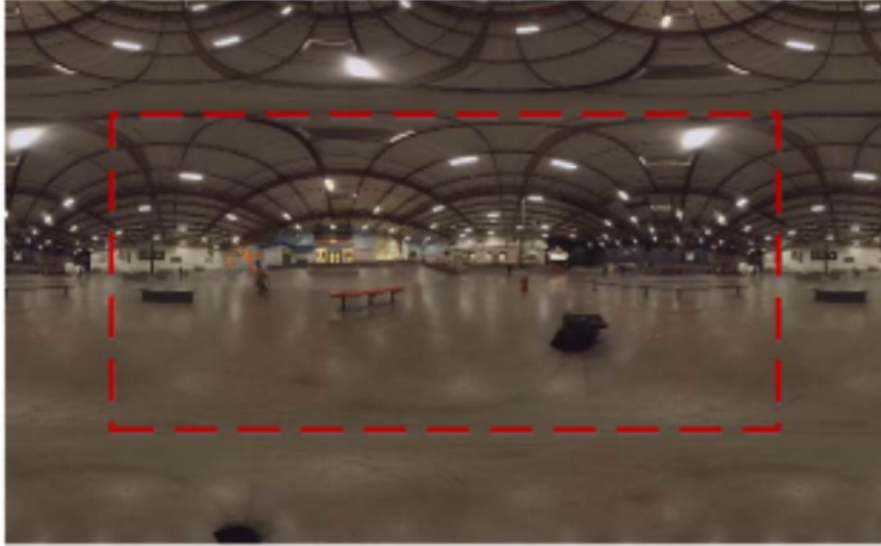


Figure 9: Same image from Figure 8, but this time it has been padded with geometry padding [6]

If the frame is in a cube map projection rather than equirectangular, it's not as intuitive or easy to deduce how geometry padding should work, but the proposed padding scheme's benefits are equivalent no matter the projection type.

Comparing Figure 8 and Figure 9, it's easy to see that geometry padding provides much more meaningful frame padding. This will result in much more accurate predictions by the coding algorithm. Predictions that are more accurate require less error or residual encodings which means less information needs to be stored and transmitted. The tests that were performed within the paper showed that the proposed geometry padding achieved a BD-rate reduction of 0.3% for equirectangular projections and 1.0% for cube map projections. The authors note that the gains were more significant for sequences that involved more movement (between 1.9% and 4.3%).

## Sampling Density Modifications

A naive and basic approach to encoding an image or video frame is to treat each pixel independently and equally. This would however, result in massive file sizes and a ton of redundant information. Modern codecs and compression algorithms employ many clever tricks to reduce the amount of redundant information and therefore file sizes. Some of these tricks leverage the fact that a human is generally the target audience for the image or video and therefore the limits and shortcomings of human perception can be exploited to drop information without affecting the human experience.

This isn't always appropriate, for example if the image or video is meant to be analyzed

by a machine learning algorithm such as Computer Vision, the algorithm may be able to utilize some of the information that wouldn't be important to a human. The panoramic or 360 degree video that we are discussing in this paper is generally created for human consumption so the next few topics will be exploring ways to discard information from a 360 degree video without affecting the human viewing experience too drastically.

## Downsampling and Smoothing

Downsampling is the process of reducing the sampling rate of a given signal. Simply put, in the context of images and video frames, downsampling can be thought of as reducing the resolution for a particular region of the frame. Two of the main reasons that this is done are if that region is not very important to the overall image or if that region doesn't have much detail. In these cases, these regions can be downsampled without the overall quality of the image or frame being affected too much.

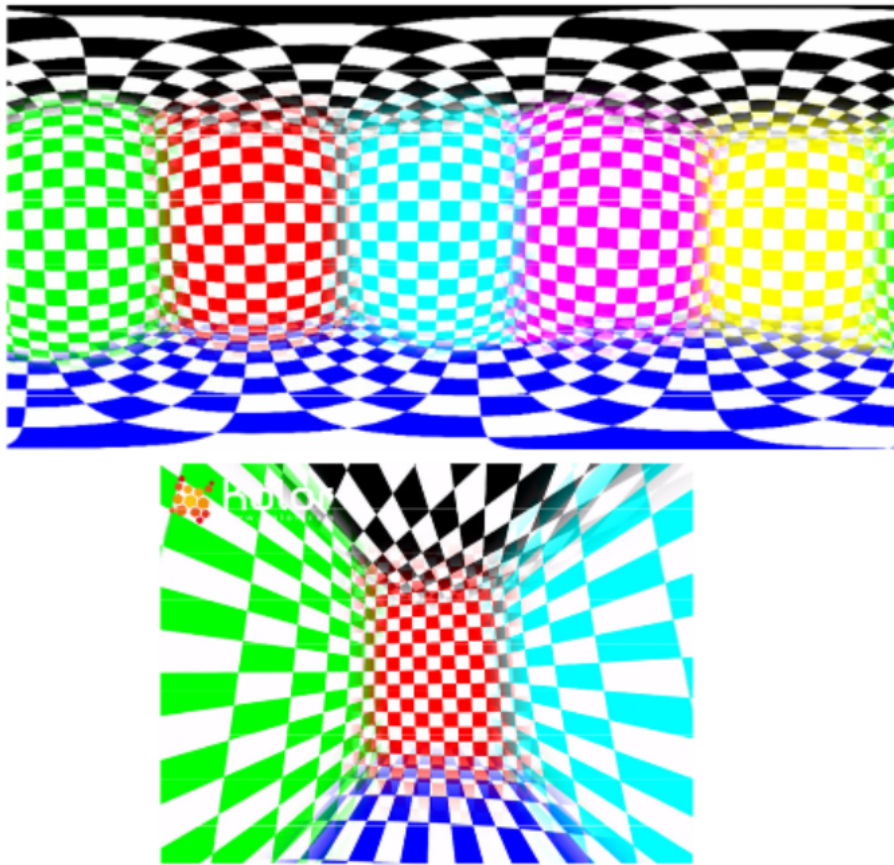


Figure 10: Top (black) and bottom (dark blue) of a frame are given much more resolution in the equirectangular 2D space (top image) than they require when rendered in the 360 degree video viewer (bottom image) [7]

Many of the compression techniques in modern codecs utilize downsampling in some capacity, but none of them are specific to 360 degree video. One intuitive idea for employing downsampling for 360 degree video is to vary the sampling rate of the video regions based on the region’s latitude. The user will rarely be looking straight up or straight down while viewing the video and if they are, the content of the video in those regions probably isn’t very important to the overall experience. Because of this, it’s natural to assume that we could have lower sampling rates for those regions. This would result in a blurrier, smoother, and/or less detailed image for those regions, however the user experience likely wouldn’t be affected much and this could result in a significant reduction in bitrate.

M. Budagavi et al. in [7] present a “region adaptive video smoothing technique that exploits the unique characteristics of 360 degrees video to provide up to 20% bitrate savings with minimal perceptual quality degradation while requiring no modifications to the video decoder.” This paper focuses less on the fact that the top and bottom of a frame are less important and more on the fact that due to equirectangular mapping, the top and bottom of a frame are given a disproportionate amount of the 2D resolution. Figure 10 is a good visual representation of this fact. Notice how in the bottom image (how the image is displayed by the VR display device) the black and dark blue regions are given a fairly similar amount of screen space as the other colour regions. However, in the top image (the image in equirectangular projection) the black and dark blue regions are given significantly more screen space than any other colour regions.

The technique used to apply the downsampling within [7] is Gaussian smoothing. A Gaussian smoothing filter is applied to the image at varying levels of smoothing based on the latitude (the closer to the top or bottom of the frame, the more smoothing that is applied). The authors note that for natural video sequences, the video quality starts to become noticeably degraded at around 20% smoothing whereas for computer generated images, the degradation becomes noticeable around 15%.

## Adaptive Quantization and Frame Rate Adaption

M. Tang et al. in the paper “Optimized video coding for omnidirectional videos” [8] proposed two different techniques for reducing the bitrate in 360 degree videos. One technique (adaptive quantization) is for equirectangular projections and the other (frame rate adaptation) is for cube map projections.

The adaptive quantization technique proposed stems from the same logic as the downsampling in the previous section. Because the top and bottom of each frame are disproportionately represented in the equirectangular projection, we do not need to spend as many bits encoding those regions. Their proposed formulas, in simplified form, involve providing finer grained quantization to regions closer to the frame equator and less fine grained quantization to the regions closer to the top and bottom of the frame.

The intuition behind the frame rate adaptation is that sequences with less movement can be given a lower frame rate without the viewer being able to tell the difference. This works

well with cube map projections because each cube face can be encoded with a separate frame rate. This means that even if there is significant movement within an entire frame, some of the cube faces may not see much movement. The cube faces with less or no movement can be encoded with a lower frame rate which will lower the overall bitrate. By allowing for specific faces within the cube map to have adaptively lower frame rates, those faces can either contribute to a lower overall bitrate or maintain their bitrate and benefit from increased visual quality. This does not work well with equirectangular projections due to the entire frame being “connected” so if any part of the frame within a sequence had significant movement, you couldn’t lower the frame rate of the rest of the frame without also lowering the frame rate of the region with the movement.

“Experiments implementing the proposed algorithms in the open source x265 encoder prove that the proposed algorithms can help save 14.69% and 13.01% of the bitrates for the cube map and equirectangular projected videos respectively while maintaining the visual quality after the compression. Furthermore, the proposed algorithms are compatible with and therefore can collaborate with the currently used adaptive spatial resolution scheme in the real-world VR processing system for further bitrate reduction.” [8]

## Spherical Wavelets

There is an additional technique we can apply to the spherical projections discussed above. All video data before being sent to the video feed can be represented as sums of functions, which can be viewed as the “frequencies” of the data. The functions are combined and compressed in a (usually) lossy fashion, then sent through the feed to display graphical elements to the user. In the case of  $360^\circ$  video, functions created on the 2D domain are projected onto a 3D sphere. A much more efficient way to represent functions are utilizing wavelets.

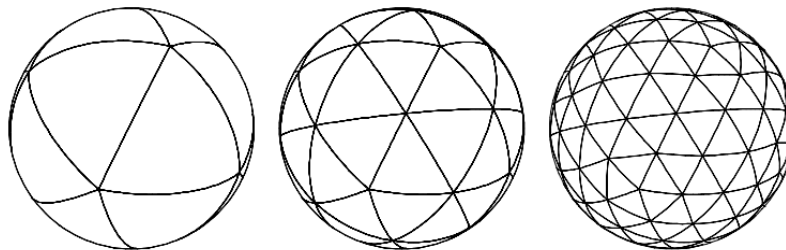


Figure 11: Wavelets on a Sphere [9]

Wavelets essentially cut a function into different frequency components, which are scaled to match a certain resolution. It oscillates with a positive amplitude, and can be visualized as a “brief oscillation”. Consequently, wavelets require a very few amount of coefficients to represent data. This means that wavelets are very compressible, much more so than their originating functions after conversion. “They offer both theoretical characterization of

smoothness, insights into the structure of functions and operators, and practical numerical tools which lead to faster computational algorithms”. [9]

What makes wavelets ideal here is that computer graphical data very accurately translates into wavelets. Specifically, functions representing surface edges, volume illumination, shadows, lighting, etc are efficiently represented by wavelets. For the purposes of video compression calculations for interactive mediums (such as VR), these elements make the bulk of the computations.

For 360° or panoramic video, as discussed earlier, 2D video functions are projected onto a 3D spherical plane. If we translate said functions to wavelets, we need to be able to efficiently do so. Luckily, wavelets are actually much more efficient to project as well. Functions must be bounded to finite domains before projection is possible; video graphical data cannot be represented by infinite functions. Wavelets are, more often than not, incredibly easy to bound as opposed to their native functions since the construction of a wavelet involves restricting the domain.

Construction of a wavelet also allows further scaling reductions if quality can be reduced. This can help 360° video for VR attain target framerates if video compression takes too long. Typically, quality on objects not in center focus or close by can reduce their wavelet functions (and thus graphical quality) efficiently if needed. Figure 12 shows how a simple function projected on a sphere can be scaled multiple times using wavelets.

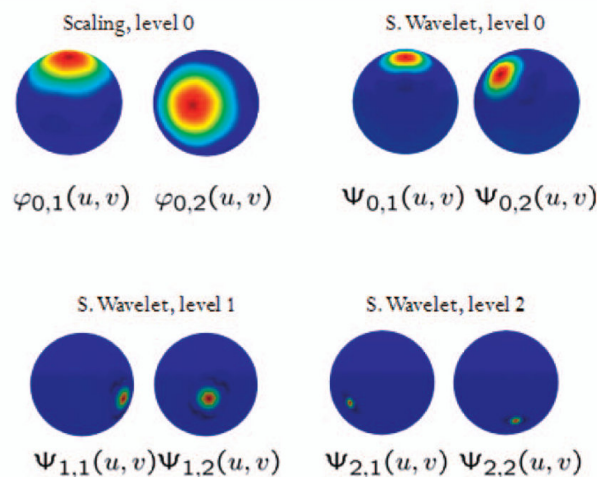


Figure 12: A function projected on a sphere (top left) with multiple wavelet scalings

## Still Images

As discussed previously, 360° video for VR demands very high resolutions, many multiples more than regular video. Because of this, low-end devices with VR support, such as smartphones, may not have the necessary processing power to achieve this. Typically smart-



phones already reduce video quality and tracking ability in order to attain target framerates. A recent solution to this is to incorporate still images into video compression.

The approach is to only process/render the focus of the frame as a video, and keep the rest as still images. The idea is that, typically, users in VR (and real life as well) turn their head to look instead of just move their eyes, meaning that moving objects of interest are at midheight of the frame. If more computing power is needed, the video can be split horizontally into three sections. The middle section would be rendered as video, the top and bottom would stay as still images. Figure 13 shows an example of how a video could be sectioned.

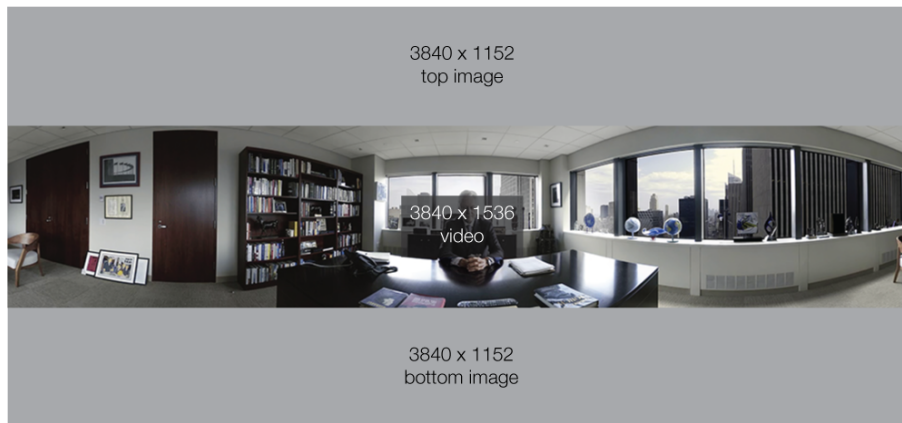


Figure 13: A frame where only the middle section is processed as video (headjack.io)

Note that this technique is not a video compression technique; it is done before compression as a way to lessen the load, resulting in higher possible framerates and resolutions. This is very noticeable to users if they specifically look for it (ie. up or down), but by processing less than half the frame, compression times drastically decrease without affecting apparent resolution. Due to this, still images are an efficient fix for low-end VR such as 360° video viewing, chatrooms, conferences, etc.

## Asynchronous Reprojection

Head motion responsiveness and framerate are by far the most imperative aspects of VR to keep stable for usability. Sometimes, even with usage of techniques listed above, complex graphical frames are not compressed in time for the video feed. When this occurs, high-end VR headsets have the necessary technology for a backup method.

Asynchronous Reprojection is an additional form of motion estimation developed specifically for high-end VR headsets. It takes the previously rendered frames, and uses motion info inputted from the headset sensors to warp said previous frames into an interpolated next frame. This method differs from the previously discussed motion estimation, which predicts based on motion specifically detected within the previous frames. Reprojection utilizes the headset hardware to track current motion force vectors and adds it to the previous frame.

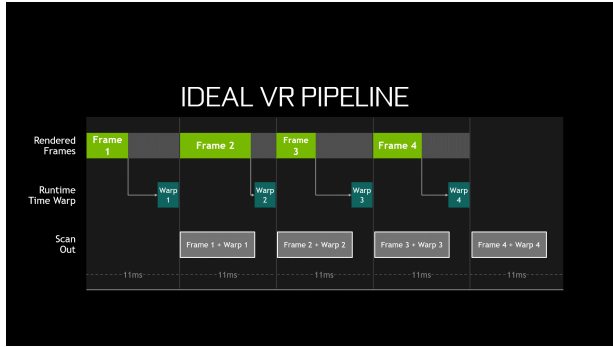


Figure 14: Ideal frame sequence with no frame drops

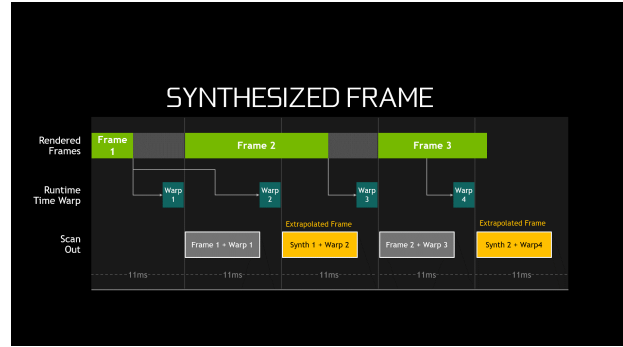


Figure 15: Synthesized frame extends second frame with warp if third frame unready

The reason it is labelled "Asynchronous" is because this operation is performed completely independent of video rendering, or in parallel to video compression. This reprojected frame is displayed without delay if the "real" frame is not rendered after compression in time for the display. This allows high-end VR to always maintain target framerates, resolutions and quick response times, regardless of rendering and compression complexity. The reprojected frames are usually indistinguishable from the regular frames to the user if they are interlaced at a reasonably fraction. Figures 14 and 15 above are a video timeline showing how an unready frame is dropped in place of the interpolated or "warped" frame.

## Conclusion

The video compression techniques covered in this report are some of the major methods used for Virtual Reality. These have come together to allow current VR to be 100% ready for consumer use by attaining satisfactory framerates and resolutions. The only limiting factor for the general public to experience immersive virtual environments is the need for a powerful PC and headset. Future developments and new, more efficient compression techniques will allow for less powerful (and more affordable) hardware to be able to run VR on a potentially more impressive scale than today.

## Team Member Contributions

Both team members contributed to the topic, report, and the presentation. We both searched for research papers and compression techniques; we were in constant contact with each other whenever we found something deemed useful for the report.

To be more specific on contributions, Patrick created and wrote much of the report, Nicolaus finished and formatted the report. Patrick created and voiceovered half the slides for the presentation, Nicolaus finished the slides and formatted the video.

However, again, we both researched VR video compression, wrote sections in the report, and made slides for the presentation. Both team members contributed to each part of the project.

## References

- [1] Y. Wang, L. Li, D. Liu, F. Wu, and W. Gao, "A new motion model for panoramic video coding," *IEEE International Conference on Image Processing*. IEEE, 2017, pp. 1407–1411.
- [2] R. Ghaznavi-Youvalari and A. Aminlou, "Geometry-based motion vector scaling for omnidirectional video coding," *IEEE International Symposium on Multimedia*. IEEE, 2018, pp. 127–130.
- [3] Y. He, Y. Ye, P. Hanhart, and X. Xiu, "Motion compensated prediction with geometry padding for 360 video coding," *IEEE Visual Communications and Image Processing*. IEEE, 2017, pp. 1–4.
- [4] C. Li, M. Xu, S. Zhang, and P. Le Callet, "State-of-the-art in 360 video/image processing: perception, assessment and compression," *IEEE Journal of Latex Class Files*, Vol. 14, No. 8. IEEE, 2019
- [5] L. Li, Z. Li, X. Ma, H. Yang, and H. Li, "Advanced spherical motion model and local padding for 360° video compression," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2342–2356, 2019.
- [6] Y. Wang, D. Liu, S. Ma, F. Wu, and W. Gao, "Spherical coordinates transform-based motion model for panoramic video coding," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 98–109, 2019.
- [7] M. Budagavi, J. Furton, G. Jin, A. Saxena, J. Wilkinson, and A. Dickerson, "360 degrees video coding using region adaptive smoothing," *IEEE International Conference on Image Processing*. IEEE, 2015, pp. 750–754.
- [8] M. Tang, Y. Zhang, J. Wen, and S. Yang, "Optimized video coding for omnidirectional videos," *IEEE International Conference on Multimedia and Expo*. IEEE, 2017, pp. 799–804.
- [9] Schröder, Peter, and Wim Sweldens. "Spherical wavelets: Efficiently representing functions on the sphere." *22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995.