Real-Time Chromostereopsis for Arbitrary Three-Dimensional Scenes

Vladislav Gaidoukevitch
Supervisor(s): Petr Kellnhofer, Elmar Eisemann
EEMCS, Delft University of Technology, The Netherlands
24-6-2022

A Dissertation Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Chromostereopsis is a visual illusion that can produce perceived 3D images through an effect caused by how humans see different wavelengths. ChromaDepth® glasses may be worn to exploit this phenomenon and amplify the effect. Relative to other forms of stereoscopy, chromostereoscopy is lesser-known and has seen fewer applications. This paper seeks to address this, by investigating possible solutions to augment any arbitrary 3D scenes for chromostereopsis. We present techniques to apply the effect in real-time to 3D scenes, while maintaining the scene's original shading, lighting, and optionally some degree of original colour. Additionally, we propose concepts for interfaces that allow user adjustment to applying the effect in areas where user studies have shown variance in preferences. These range from curves that determine varying depth precision, to creating custom chromostereoscopic colour maps, and modifying the speed at which the depth-colour mapping changes, for those sensitive to stereopsis.

1 Introduction

Nowadays, there are a number of methods to achieve a generated stereoscopic vision, to better visualise the third dimension for any variety of purposes, on two-dimensional surfaces. Whether anaglyph 3D, polarised 3D, or CrystalEyes™ 3D [6], multiple options already exist to achieve this effect; each with their own benefits, costs and limitations. One of such options that is perhaps lesser-known, is using chromostereopsis to achieve 3D. This is an observed illusion in human vision whereby different wavelengths of light present themselves at different depths to the viewer [8]. The company Chromatek has developed their ChromaDepth® glasses to further enhance this effect with prismatic lenses [2]. Figure 1 demonstrates this effect.

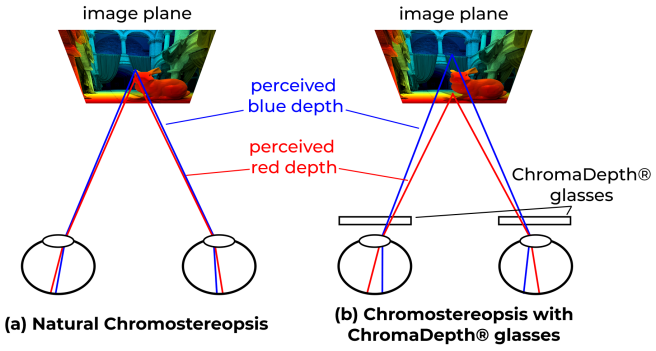

Figure 1: A demonstration of chromostereopsis with and without ChromaDepth® glasses.

Besides chromostereopsis, all other mentioned forms of achieving digital stereopsis have seen popular application in real-time 3D through, for example, injection into 3D games, retrofitting them with stereoscopic vision by major hardware vendors [9]. Existing research on chromostereopsis has explored some means of displaying this effect in real-time 3D scenes [2; 16; 4]. However, this paper aims to investigate further into what aspects would be necessary, interesting, and possibly of use in the arbitrary conversion of real-time 3D scenes to support this visual phenomenon. Potentially, this will provide real-time 3D graphics with insight into allowing a new method of presenting general stereopsis. More precisely, this research aims to answer the question: "What are effective ways to display chromostereoscopic depth in real-time 3D scenes?"

In this paper we propose a system to augment 3D scenes with functional chromostereopsis. This is done in a generalised manner, allowing for the augmenting of potentially arbitrary 3D scenes, with a focus on preserving the original appearance. This paper describes several techniques to create and assist with chromostereopsis, along with suggesting adjustable parameters for broader flexibility where necessary. A small user study was conducted to evaluate the efficacy of each technique and variation in user preferences for them.

2 Related Work

The study of chromostereoscopy began its roots with the first observation of chromostereopsis' effects in 1885 [8]. Since then, the effect has seen use in a variety of applications, such as topography [20], medicine[17], art [13], and paleontology [3].

In real-time 3D visualisations, there has been implementations of chromostereoscopy in Bailey's [2] and Schemali's [16] work. Bailey's work outlines the basis for generating a chromostereoscopic effect performantly with OpenGL. Schemali's work appended this knowledge with improved rendering for trichromatic displays, as well as exploring some depth techniques to enhance stereo perception.

Our implementation of real-time chromostereopsis aims to differ from the previous approaches in a few ways. Bailey [2] noted that the programmer must "place the near and far clipping planes very tightly around the scene" in order to optimise the depth-colour mapping, as otherwise the depth mapping may only use a small range of the total colour map and be inefficient. This is curation that cannot be applied to arbitrary 3D scenes, so our approach alleviates this by developing a generalised shader that maximises the depth mapping for any given image.

In Schemali's research [16], they observe the positive effects that Lambertian shading with "standard lighting from the right/left upper quadrant with respect to the view" provide to the image, giving additional detail and depth perception. We attempt to improve upon this idea by allowing arbitrary shading techniques to be combined with a chromostereoscopic depth-colour mapping. Specifically for our testing, we choose for the more modern physically-based rendering/shading (PBR) [14] approach, which has seen widespread adoption in recent years, and could therefore be a good candidate for testing arbitrary shading models in 3D scenes. Furthermore, its use of additional maps for roughness, normals, and added lighting detail through shadows, specular highlights, and spherical harmonics may all contribute to more detail that may assist with depth perception. Figure 2 demonstrates an example of this form of

real-time rendering with the scene used for this paper [12; 10]. Additionally, we aim to retain the original lighting position(s), out of an interest to preserve the original appearance of the scene.

Figure 2: A 3D scene rendered with modern PBR shaders and lighting; a common standard in current real-time graphics. This is the testing environment used for this paper.

Toutin and other researchers after him have also noted the use of linear mapping of depth to colour across the scene [2; 16; 20]. However, perceptually, a viewer typically guides more of their attention towards objects closer to themselves, and real-time computer graphics reflect this in how depth precision is typically calculated with an inverse relation to distance from the camera [15]. For this reason, we also investigate options for customisation of how colours are mapped to the depth of the image.

In motion, stereoscopic depth has some history in its ability to cause nausea and other unpleasant feelings due to confusing the human senses and our perception of depth [7; 11]. Our implementation of depth motion smoothing was created to try alleviate some of these.

3 Setup and Results

The setup for the application used in this research consists of first mapping the scene's depth to a chromostereoscopic colour map, in section 3.1, followed by adding additional lighting and shading details in section 3.2. Options for altering the scene's depth precision are then examined in section 3.3, followed by exploring ways to minimise unpleasant, sudden depth changes in section 3.4. Finally, in section 3.5 we add two interfaces to create custom colour maps in the program, and custom depth precision maps based on user-drawn curves.

This setup will assume an existing 3D scene with all objects, lighting and materials in place. The scene used for this setup is lit by static, precomputed global illumination with spherical harmonics. Multiple reflection probes are also present, although light probes are not, as the scene contains no dynamic geometry. A global post-processing is applied to the image for (softer) tone mapping and auto exposure in the scene.

To first create a chromostereoscopic image, the depth of a scene must be mapped to colours. The nearest objects are coloured with longer wavelengths and further objects with shorter wavelengths. This depth-colour mapping forms the basis for chromostereopsis.

3.1 Depth-Colour Mapping

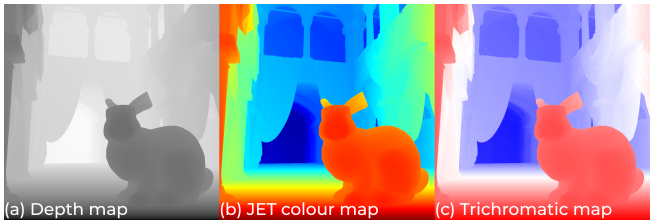

Figure 3: A depth map next to two depth-colour mappings.

The basic form of depth-colour mapping is the foundation of this implementation. A linear depth buffer/texture is obtained (linear from the near to far clipping planes, as generally is performed in 3D chromostereoscopy [2; 16; 20]). This is combined with a colour map, such as an HSV range from red to blue where each pixel p samples the colour map, and returns the position of the depth texture's intensity for that pixel. In other words,

$$p = tex(m_c, d)$$

where tex is a function that takes as input a colour map, m_c, and a depth value in [0, 1] to sample the texture's UV position, d. This produces a simple chromostereoscopic image that "colours in" a depth texture. Given that the function takes in an arbitrary texture as input, it is trivial to substitute the colour map for another, assuming only that its UV mapping is bound by range [0, 1], and the gradient(s) occur in a uniform direction. Figure 3 demonstrates two valid colour maps, one using MATLAB's Jet colour map [18], and the other using Schemali's trichromatic mapping [16].

From a simple depth-colour mapping with a linear depth buffer, an issue is presented that the full colour map may not be utilised if the minimum and maximum ranges of the depth are bound by the clipping planes. More usefully, maximising the depth range would be preferable if bound to the minimum and maximum depth of what is immediately visible on-screen. The minimum and maximum are possible to obtain through a linear search of all pixels per frame, but this is quite slow for a real-time application (approximately 180ms per frame at a 1024x1024 resolution). Alternatively (and with a 35x speedup in our testing scene), this is achievable through generation of mipmaps [23]. The idea is to create two custom mip chains, one where each mip level's pixels are the maximum of their local neighbours, and the other where the pixels are the minimum. After the full mip chain is generated, the highest level (a single 1x1 pixel texture) will present the respective furthest and nearest point visible. With these known, the depth texture's bounds can be remapped from a [0, 1] range to instead use the known minimum and maximum, using an inverse linear interpolation given by the equation

$$InvLerp = \frac{(x - min)}{(max - min)}$$

which, for an input x will remap to a range from min to max. This then allows the full use of the colour map to be visible across each frame, without adjusting clipping boundaries.

3.2 Shading

Figure 4: A 3D scene with greyscale shading displayed together with a depth-colour mapping. In (c), some of the scene's original colours are also reintroduced.

With a depth-colour mapping alone, a lot of a 3D scene's details may disappear, as texture detail among other factors are not included in a depth texture. It is possible to reintroduce these details by converting the original image to greyscale, and merging its output with the depth-colour result achieved previously. First, in converting the image to greyscale, for each pixel p we use the equation

$$p = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

where R, G, and B are the red, green, and blue channels respectively. This equation is the NTSC standard for converting colour to black and white[19], based on human perception of colour. With this, we can add the greyscale image together with the depth-colour image by multiplying the two together.

It is possible to reintroduce the scene's original colours (partially) as well. Using the depth-colour output with greyscale shading applied, we can merge it together with the original colour image based on a scaling factor of choice, such that

$$p = c \cdot f + s \cdot (1 - f)$$

where c is the scene's original colour output, s is the depth-colour shaded output, and f is a blending factor in [0, 1] that determines how much of the original colour to reintroduce to the image. This approach does not provide a guaranteed blend factor that will work for every scene, so this is a modifiable parameter that should be evaluated per scene, and the colours the scene has in relation to the colour map, if used. Figure 4 demonstrates an example of this with an f of 0.4 (40% of the scene's original colours blended in).

3.3 Depth Precision

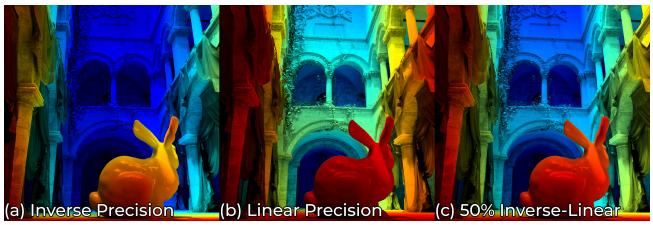

Figure 5: Depth precision in the colour mapping using (a) an inverse 1/z relation, (b) linear precision, (c) an average of the two.

With colours assigned to depth values in a scene, it is clear there are only finite points on a colour map to assign depth to. Alternative depth mapping techniques may provide more depth detail in areas that have more use for it. The first real-time implementation of chromostereopsis [2] states the use of a linear depth mapping, without a particular motivation for it. Toutin [20] elaborates by correlating the familiarity of linear depth with "taking advantage of the viewer's conceptual knowledge of the perspective phenomena." This is certainly useful in some applications, such as visualising geological features and topographic maps. However, when viewing a scene frontally from a virtual camera's perspective, this is not necessarily the case. People generally pay more attention to objects closer to them, and real-time computer graphics reflect this in how depth precision is typically calculated with an inverse relation to distance from the camera [15]. Leveraging this concept, we may want to use more of the depth-colour map for objects closer as well, to enhance the perception of depth there. We implement this along with a scalable factor f between the inverse and linear depth, similar to its implementation in the coloured shading. An inverse depth buffer is provided to us with the Unity engine, but its equation is performed as

$$d = a \cdot \frac{1}{z} + b$$

where d is depth (stored as a [0, 1] value), z is world-space depth, and a and b are the near and far plane's distances to the camera. Figure 5 illustrates how different methods of precision affect the depth-colour mapping.

3.4 Depth Motion Smoothing

During camera motion, the colours created by the depth-colour map may sometimes change rapidly as an effect of the adjusting minimum and maximum depth of the view. This effect may potentially be intense for some users, so we provide a method to slow down the adjustment of these changes. Per rendered frame, we store the minimum and maximum depth of the previous frame. With this, we may perform the equation

$$min_x = min_o \cdot f + min_x \cdot (1 - f)$$

where min_o is the recorded minimum distance from the previous frame, and f is a parameter in [0, 1] that determines how much the depth is adjusted per update/frame. With each update, the new minimum will gradually converge into the "true" minimum, until it is changed again. The same equation can be performed for the maximum depth, with the appropriate parameters exchanged. The result of this calculation is then forwarded to the shader which will use these values as its input for minimum and maximum, rather than the "true" minimum and maximum.

3.5 Additional Tools

To step beyond linear or inverse depth precision, there may be scenarios where a more complex precision scale may be necessary. To support a larger range of mappings for this, we have added the support for the user defining their own precision function using Bézier curves. In an interactive interface, the user may define a curve by hand or select presets, and observe the scene adjust the depth-colour mapping in real-time, as the adjustments are made. In order to make this operation performant, the curve is evaluated to a 1-dimensional gradient texture that is generated and sent to the shader during every adjustment. This texture is created by dividing the curve into 512 points along the X axis, and evaluating each into a greyscale value that remaps the [0, 1] range of the curve to [0, 255] for colour. In Figure 6, we demonstrate an exaggerated example with a custom curve function that places all emphasis on the bunny in the scene. The depth effect is greatly focused on the subject. Areas of the curve outside of the range are clamped within bounds.

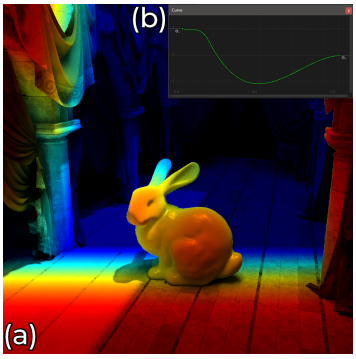

Figure 6: A custom depth precision curve (b), designed to emphasize the subject in scene (a).

Another feature added was a tool for the user to generate their own colour maps within the program. This would save the user relying on an external tool for the creation of these maps and enable customising colours, perhaps in an application-specific manner, where a colour map may be designed to interfere less with the scene's original colours. A demonstration of the interface is provided in Figure 7, which consists primarily of Unity's Gradient API. Upon opening the option to create a gradient, the user is presented with an editor that previews existing presets, and allows the user to create and save new ones. By selecting anywhere along the gradient, they can place and move anchor points that hold a colour of choice. Any amount of anchors can be placed, and colours will be blended between them. Alternatively, the mode above can be changed from "Blend" to "Fixed" to disable any colour blending between anchors. Selecting "Save Gradient" will save the gradient as an image file that can be slotted into the shader's colour map slot. An overview of all of the features can be viewed in video as well [22].

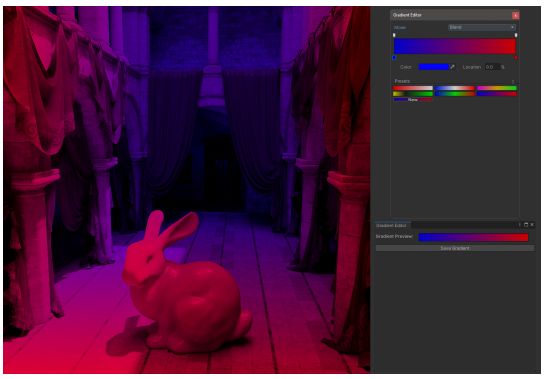

Figure 7: A custom colour map defined in the interface on the right and displayed in the scene.

4 Responsible Research

The research conducted in this paper evaluates methods of using a proprietary form of stereoscopic vision. Through this research, some conclusions are drawn about this technology that are subjective, but the conducting of the research expands the global knowledge of Chromatek's ChromaDepth® system. This paper was not sponsored, incentivised, or in any way guided by Chromatek, and was conducted out of own interest in the subject matter. As such, we believe our findings and conclusions are ethically sound for this research.

Similarly, we find a personal interest in the overall research field of stereoscopy, and this has been a partial motivation to create this paper. Stereoscopy itself is a very old concept that the world has used for a very broad variety of purposes since its inception, and we do not believe the technology alone can be perceived as unethical.

The Unity® game engine developed by Unity Technologies [21] and used for the creation of the program described in this paper, was chosen primarily out of familiarity with the engine from past experiences and the efficiency it would provide for this project. It is a free engine to use non-commercially, and much of its original code has been published openly. Furthermore, it is widely-used in the real-time rendering industry, and as such, there is value in conducting research on it that may find use in this field. These were our factors that we considered when we chose to use it for this research. Our program currently requires the use of this engine to run, but all aspects of our application can be translated to other software.

The test scene for this paper, Sponza Atrium [12], is a popular choice for all manner of graphics research due to its academic license permitting the use of such. The specific iteration we chose additionally contains updated, modern rendering features, such as the appropriate texture maps for PBR

rendering, and a level of detail that matches contemporary 3D environments. By using an environment that is commonly used for similar research, comparisons between other articles should hopefully be easier to do as well. The environment shaders in the scene, Filamented [10], were chosen out of compatibility with the scene and rendering engine, and provide newer features than the standard shaders from Unity.

We have done our best to include any specific implementation details that we believe would otherwise change the outcome of this research, and the rest is explained broadly in terminology that should allow software-agnostic rewriting elsewhere. Some aspects of the implementation were omitted for conciseness, where deemed irrelevant to the topic and implementation.

5 Evaluation

A small user study was conducted on six people to evaluate the efficacy of the techniques developed. These were conducted in various locations with differing lighting conditions, but all on a Surface Pro 6 laptop, with ChromaDepth® glasses. Before any evaluation of features, the users were first asked if they were able to perceive depth with the glasses on. All stated they were able to, except one which was only unable to with shading disabled. This was important to ask prior to further questions, to ascertain that they were susceptible to chromostereopsis at all.

The first feature to be evaluated was the shading technique, and users were asked with which they observed "more depth," where depth was defined as the discrete number of "slices" along the depth axis in the image that they could perceptually distinguish. They examined scenes with no shading (depth-colour map only), Lambertian shading lit from the right/left upper quadrant with respect to the view (as noted in Schemali's work [16]), and the scene's original PBR shading. An example of these is shown in Figure 8. Five out of six users noted a preference for the PBR shading's added detail, specifying that the detail largely contributed to the increase in depth perception. One user made the observation that with the PBR shading option, however, some angles would have less lighting contrast than the other two methods, which either had a fixed lighting with respect to the camera (Lambertian), or none. Therefore, those angles were perceived with less depth than Lambertian for them.

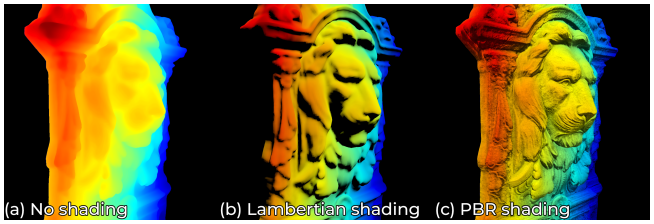

Figure 8: An example of the scenes users viewed to compare shading techniques.

Users were also asked to evaluate the use of different colour maps in the scene. For this test, PBR shading was exclusively used and the users were again asked with which option they perceived "more depth" as defined previously. The choices were given between a hue shift map (a fully saturated colour gradient from red to blue, with all colours in between), MATLAB's Jet map [18], Schemali's trichromatic map [16], and Google's Turbo map [1]. A comparison of the 4 options is demonstrated in Figure 9. These pre-existing maps were selected for their compatibility with chromostereopsis and history of use for other forms of data visualisation. Four out of six observed preference for the Turbo map, and two users preferred Jet. All users stated a close match between those two options, with reasons for Jet stated as its saturated colours, while Turbo was more pleasing to observe and had more individually discernible colours.

Figure 9: A demonstrative comparison between pre-existing colour maps.

After explaining the principles of depth precision in depth-colour mapping, users were asked for their preference here as well. When asked to adjust the scaling factor f, which determines the blend between inverse and linear depth, there was a wide variance in answers regarding which value users perceived most depth with. Answers ranged from one user choosing 0 (fully inverse mapping), and the rest up to 70% linear-inverse, with none preferring completely linear mapping. Most users preferred somewhere midway, but the variance in such a small sample of users may indicate that it is a good parameter to leave adjustable to the user.

The ability to blend the original scene's colours with that of the depth-colour map was evaluated more subjectively. It is clear that reducing the colours of the colour map to any extent will reduce the depth effect, as it is necessary for chromostereopsis, so the use of this option serves more for enhancing the scene with auxiliary information unrelated to depth. One question asked was how far the users could blend the original colours in, while maintaining a perception of chromostereoscopic depth. Here, all users answered an amount between 10-40% of the original colours reintroduced, combined with their colour map of preference. Visu-

ally, however, all users preferred this option completely off or preferably 10% or lower, with the addition of the scene's "true colours" being observed as disorienting/jarring with ChromaDepth® glasses.

Lastly, depth motion smoothing was evaluated by the users. With none present, no user reported any discomfort or motion sickness, but all preferred the setting enabled for less sudden colour changes.

6 Limitations

With the benefits that the techniques introduced in this paper bring, some limitations and constraints are present, and bear mentioning. One of such is in the depth-colour mapping, which samples the minimum and maximum indiscriminately. This means if situations arise where, for example, a very small object is present on-screen at a very far distance, the depth will be stretched to accommodate it, despite its (potential) unimportance to the viewer. Similarly, given that the sky is usually calculated to be at an infinite distance in many real-time renderers, this will exhibit the same effect whenever the sky is visible, and our implementation can be assisted by clamping the maximum distance to something closer.

Performance for this dynamic min/max depth mapping is a current bottleneck of this system. While it is enough for real-time (render time of $<$5ms per frame on an Nvidia RTX 3080 at 1024x1024 resolution), generating two full mip chains for every frame is too costly for smaller or portable hardware, together with its dependant synchronisation on the CPU. This could be somewhat improved by combining the two mip chains into one with two colour channels, where each channel represents the respective min/max. Our implementation also assumes square image aspect ratios to simplify mipmap generation, but this could be expanded upon with edge padding beforehand.

With shading enabled, one limitation of the retaining original lighting is the depth-colour map in darker scenes. Our current implementation shades these darker scenes at their correct intensity, which has the side effect of diminishing chromostereopsis if it is too dark.

7 Conclusions and Future Work

This paper has presented and explored a number of ways to introduce chromostereopsis into arbitrary 3D scenes. We have shown methods to allow full-range depth-colour maps to be rendered in real-time, together with arbitrary scene shading techniques, and modifiable depth precision. We have further explored that there are ways to smooth the presentation of chromostereoscopic depth if desired, and introduced novel interface ideas for adjusting a number of the parameters hereof. Evaluation of these techniques has shown that they are effective in displaying chromostereopsis to users wearing ChromaDepth® glasses, throughout the scene. The addition of modern shading to the image was appreciated, as well as the ability to adjust how depth was mapped, with depth preferences being mostly unique to each user. Similarly, the effectiveness of colour maps seemed to vary per user, while in motion, the smoothing of depth changes was unanimously preferred.

Beyond the scope of this paper and project, there are a number of related challenges that may be interesting to address in future work. Certain graphical features present in real-time 3D were not handled in this paper, such as handling depth for UI overlays, screen-space effects (e.g. bloom, ambient occlusion, post-effects that do not present at specific depths), and skyboxes. For user interfaces, a number of other ideas could also be explored, such as making the adjustment sliders for users perceptually linear, rather than the scale currently used. Additionally, more creative control over depth, for example through the use of manually selecting objects in the scene to emphasize, is an idea that has not been explored yet. Possibly once some of these features are addressed, it would be very interesting to see the techniques shown in this paper further applied to an injection tool that can hook into modern rendering APIs, such as ReShade [5].

We hope that through this paper and the ideas explored within, we have broadened the knowledge in the field of chromostereoscopy, and have brought some further interest into this type of stereoscopic vision.

References

[1] Anton Mikhailov. Turbo, an improved rainbow colormap for visualization, 2019. https://ai.googleblog.com/2019/08/turbo-improved-rainbow-colormap-for.html.

[2] Michael Bailey and Dru Clark. Using chromadepth to obtain inexpensive single-image stereovision for scientific visualization. *Journal of Graphics Tools*, 3(3):1–9, 1998.

[3] A Boczarowski. Chromo-stereoscopy as a tool in micropalaeontological investigations: Echinoderms as a case study. *Studia Geologica Polonica*, 124:21–35, 2005.

[4] Alan Chu, Wing-Yin Chan, Jixiang Guo, Wai-Man Pang, and Pheng-Ann Heng. Perception-aware depth cueing for illustrative vascular visualization. In *2008 International Conference on BioMedical Engineering and Informatics*, volume 1, pages 341–346. IEEE, 2008.

[5] crosire. Reshade, 2022. https://reshade.me/.

[6] Carolina Cruz-Neira, Daniel J Sandin, and Thomas A DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142, 1993.

[7] Simon Davis, Keith Nesbitt, and Eugene Nalivaiko. A systematic review of cybersickness. In *Proceedings of the 2014 conference on interactive entertainment*, pages 1–9, 2014.

[8] Willem Einthoven. Stereoscopie durch farbendifferenz. *Albrecht von Graefes Archiv für Ophthalmologie*, 31(3):211–238, 1885.

[9] Samuel Gateau. 3d vision technology-develop, design, play in 3d stereo. *Acm Siggraph 2009*, 2009.

[10] Romain Guy, Mathias Agopian, and Silent. Filamented standard shader, 2021. https://gitlab.com/silent/filamented.

[11] Joseph J LaViola Jr. A discussion of cybersickness in virtual environments. *ACM Sigchi Bulletin*, 32(1):47–56, 2000.

[12] Frank Meinl, Katica Putica, Cristiano Siqueria, Timothy Heath, Justin Prazen, Sebastian Herholz, Bruce Cherniak, and Anton Kaplanyan. Intel sample library, 2022. https://www.intel.com/content/www/us/en/developer/topic-technology/graphics-processing-research/samples.html.

[13] Jean-Paul Migneco. Reinterpreting colour-field and landscape artworks: an investigation in multi-disciplinary ways of creating and experiencing colour and sculpture. Master's thesis, University of Malta, 2017.

[14] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[15] Nathan Reed. Depth precision visualized, 2015. https://developer.nvidia.com/content/depth-precision-visualized.

[16] Leila Schemali and Elmar Eisemann. Chromostereoscopic rendering for trichromatic displays. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering*, pages 57–62, 2014.

[17] AmirAli Sharifi. Enhancing visual perception in interactive direct volume rendering of medical images. *University of Alberta*, 2016.

[18] The MathWorks, Inc. jet, 2022. https://www.mathworks.com/help/matlab/ref/jet.html.

[19] The MathWorks, Inc. rgb2gray, 2022. https://www.mathworks.com/help/matlab/ref/rgb2gray.html.

[20] Thierry Toutin. Qualitative aspects of chromostereoscopy for depth perception. *Photogrammetric Engineering and Remote Sensing*, 63(2):193–204, 1997.

[21] Unity Engine. Unity, 2022. https://unity.com/.

[22] Vladislav Gaidoukevitch. Features overview, 2022. https://youtu.be/ZADyPtfKbVE.

[23] Lance Williams. Pyramidal parametrics. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 1–11, 1983.