BRINGING 3D VISION TO THE WEB: ACQUIRING MOTION PARALLAX USING COMMODITY CAMERAS AND WEBGL

Patrik Goorts, Dimitri Scarlino, Steven Maesen, Philippe Bekaert

Hasselt University - tUL - iMinds
Expertise Centre for Digital Media
Wetenschapspark 2
3590 Diepenbeek, Belgium

ABSTRACT

In this paper, we present a system to acquire 3D vision by motion parallax on web-based platforms using head tracking. We employ the camshift algorithm to perform color-based head tracking. Using the position of the head, we can render a 3D scene from the viewpoint of the viewer, thus acquiring motion parallax, a strong cue for 3D vision. We employed web technologies to allow the adoption of our method to any modern device, including mobile devices. WebGL is used for rendering and head tracking, and WebRTC is used for camera input. No software installation or plugins are required. We demonstrated the effectiveness of our method on a variety of devices, such as desktop computers, laptops, and tablets.

***Index Terms*—** WebGL, Head Tracking, Motion Parallax, 3D Vision, WebRTC, Camshift

1. INTRODUCTION

3D vision is a well-known technology for richer immersion in digital imaging methods, such as virtual environments, movies and games. To provide 3D vision, different depth cues are used to create a perception of depth in the flat scene on the screen. Humans can use different cues to acquire a 3D perception, such as binocular disparity, motion parallax, accommodation, convergence, perspective, scene knowledge, and many more. While perspective and scene knowledge provide strong cues, depth perception is still limited due to the absence of the other cues.

In this paper, we present a method to bring motion parallax to virtual scenes and games on the web using inexpensive hardware. We use WebRTC to capture the images of a webcam, and use WebGL to process the images for tracking the face of the user. By using head location, the scene can be adapted to the viewing position of the user, thus introducing motion parallax and increasing the 3D perception. Furthermore, WebGL is used to render the virtual scene or game. By using web technologies, which can operate on any modern browser, a variety of multi platform, user friendly and easily reachable applications can be developed. The applications use the GPU directly through WebGL, allowing real-time image processing and scene rendering, thus avoiding the slower Javascript sandbox. Because the head tracking is only part of the complete system, it should be faster than real-time.

Attempts have been made to accomplish motion parallax using special devices, such as the Microsoft Kinect or the Nintendo Wii Remote [12]. Contrary to these 3D vision systems, our method requires no specialized hardware. Any commodity modern computer or mobile device has a webcam and a browser, thus allowing the use of WebGL and WebRTC, independent of operating system. Consequently, our method brings 3D vision to widely used devices.

We accomplish the face tracking, required for motion parallax, by using the Continuously Adaptive Mean Shift (camshift) algorithm [2]. The color of the face is sampled beforehand, which can be used to calculate the probability that a specific color is part of the face. By using an adaptation of the Mean Shift method [5], an area of each frame can be determined where the probability of containing the face is maximal, thus effectively tracking the position of the head. The algorithm is based on the color of the face, eliminating the requirement of calibrated and high quality camera equipment.

Using motion parallax by head tracking differs from more commonly used techniques, such as stereo vision. Here, each eye sees a different image, thus creating the illusion of being placed in a real 3D scene due to binocular disparity. The stereo screen can display the images of the two eyes simultaneously, whereby the images are distinguished by wearing glasses with color filtered or polarized glasses [9], or the system can close the two glasses in an alternating fashion. This method is commonly used in 3D cinematography. Alternatively, autostereoscopic methods, as used in commodity 3D televisions, removes the requirement to wear glasses. This can be accomplished by using parallax barriers or lenticular arrays [3].

By using stereo directly, there is a mismatch between accommodation and convergence, where the focus of the eyes is kept on the screen while the eyes converge to a point behind or in front of the screen, resulting in a blurred image. This

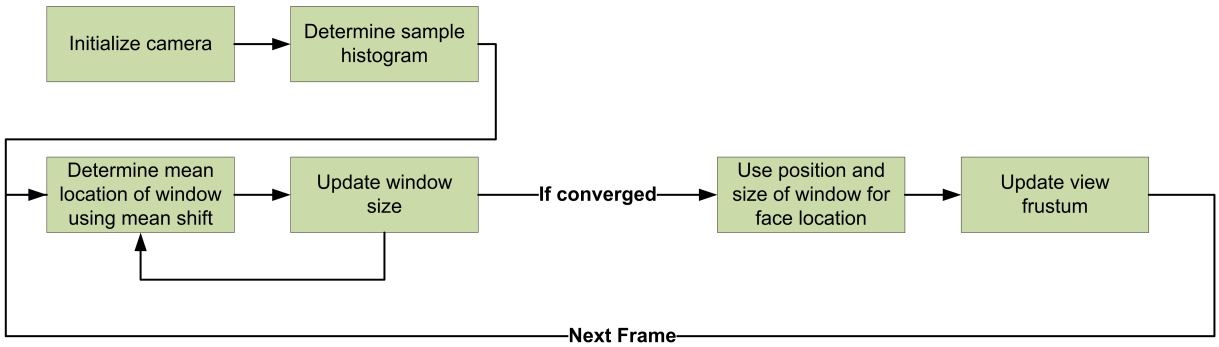

Fig. 1. Overview of our method. There are two phases, a preprocessing phase and a real-time tracking phase. The tracking phase uses an iterative process to determine the position of the head.

can cause fatigue for the users, which can result in unpleasant user experiences [11, 4, 10]. While there is a tolerance for visual discomfort [6, 19], special care might be required to reduce this discomfort. Numerous attempts have been made to cope with this issue, both in software [18] and in hardware [1, 13, 14, 16, 15], but heavy computational processing or specialized hardware is required. The accommodation-convergence mismatch is avoided by using motion parallax, where the focus of the eyes can be kept on the screen, together with the convergence of the eyes. Only one image is used, thus no blurring can occur.

Another problem with stereo vision is crosstalk, where the image for one eye is partly visible by the other eye, resulting in ghosting effects and blurry vision. This is induced by hardware limitations, and is avoided by using single-image motion parallax.

It has been demonstrated that motion parallax can be a better cue for depth perception [17, 20, 7]. Furthermore, no devices should be attached to or worn by the user, increasing the user experience. However, multi user environments, such as 3D cinematography, cannot adapt the images for every user, which limits motion parallax technologies to single user setups. Furthermore, movement of the head is a requirement for motion parallax to work. This can limit the applicability to smaller devices, where relative head movements are expected.

We demonstrated our method by running our implementation on different devices, including a normal desktop computer, a laptop, and a tablet, using separate and built-in cameras. The results show the effectiveness of our method.

2. FACE TRACKING USING CAMSHIFT

To generate motion parallax, the position of the head of the user must be known. We use the camshift method [2], adapted to WebRTC and WebGL to run on web environments. The camshift algorithm takes in a frame from the webcam, converts it to the HSV color space and iteratively searches for a window in the image where the probability for a face is maximal. It uses the color histogram of a head sample, acquired in the preprocessing phase. An overview of our method is depicted in Figure 1.

2.1. Preprocessing Phase

Before face tracking can be accomplished, some preprocessing should be done. This includes the determination of the color of the face using the chosen camera settings.

The user places his head in a predefined square in the image and the colors are sampled. The colors of the pixel in the sample are converted to the HSV color space [21] and the normalized histogram of the hue channel is stored. This histogram will be used during the tracking to look up the input color values, resulting in a probability of flesh color. By using the HSV color space, only a 1D histogram is required. It is known [22, 23] that the color of flesh is only determined by the hue channel; intensity and saturation only determine the darkness of the skin color.

Furthermore, the starting position of the face is now known, which will be used in consecutive phases of the algorithm.

2.2. Real-time Processing Phase

Once the color of the skin is sampled, tracking can be performed. Tracking per frame is performed in different steps. First, an initial search window size and location is chosen. Next, the mean location of the window is computed using the mean shift method and the search window is centered here. This repeats until the location converges. Lastly, the search window size is adapted and a new center of the search window is computed. This repeats until both the center and the size of the search window converges. The search window now contains the face. Using the size and location of the search window, X, Y, and Z position can be extracted.

The initial search window size and location is chosen as the result of the previous frame. Due to the temporal coherent

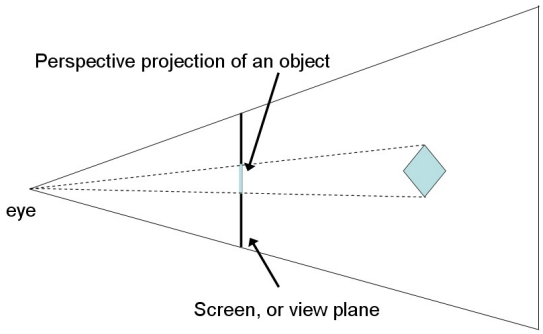

Fig. 2. Typical projection for 3D objects on the image plane. The eye is placed on the normal through the center of the image plane. [8]

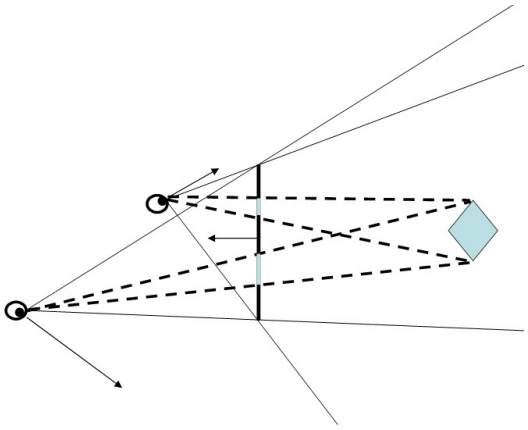

Fig. 3. Asymmetric projection for 3D objects on the image plane. The eye is no longer placed on the normal through the center of the image plane, thus a different viewpoint of the 3D scene is acquired. [8]

nature of the input frames, this initial assumptions are valid and allow a fast convergence of the method.

For a given color, we can calculate the probability of being skin color by looking up the hue in the histogram determined in the preprocessing step. The sum of all the values of the histogram add to 1, thus representing a normalized probability lookup table. Each color of the search window is converted to a probability, thus acquiring a confidence map. This confidence map can be considered as a discrete probability density function. We calculate the mode of this function using the mean shift algorithm. First, the zeroth and first moments of the probability density function are calculated:

$$\begin{aligned} M_{00} &= \textstyle\sum_x \sum_y P(x,y) \\ M_{10} &= \textstyle\sum_x \sum_y xP(x,y) \\ M_{01} &= \textstyle\sum_x \sum_y yP(x,y) \end{aligned} \tag{1}$$

where $P(x,y)$ is the acquired probability of being skin

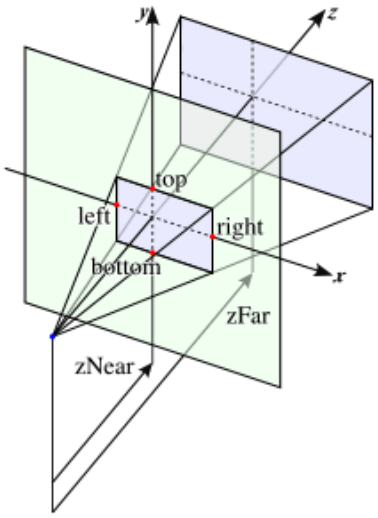

Fig. 4. The frustum is determined by 6 parameters: the coordinates of the topleft and bottomright corners of the image plane, and the depths of the near and far planes.

color of the pixel at position (x,y), and x and y range over the search window. Using these moments, a new center of the search window can be acquired:

$$x_c = \frac{M_{10}}{M_{00}} \text{ and } y_c = \frac{M_{01}}{M_{00}}$$

Once the new center of the search window is determined, a new search window width is acquired:

$$w = 2 * \sqrt{M_{00}}$$

The zeroth moment is the distribution area found in the search window, which can thus be used as a guideline to set the search window size. We increase the window size by two to allow the growing of the search area. When the window is too big, the zeroth moment stays the same as the previous iteration and shrinking occurs. Due to the elliptical characteristics of the face, the height h is set to $1.2 \times w$.

Once the window size is adapted, a new center is determined. This process is repeated until convergence occurs. After convergence, the head position is determined as follows:

$$X = c_x \tag{2}$$
$$Y = c_y \tag{3}$$
$$Z = w \times h = 1.2 \times w^2 \tag{4}$$

The depth of the head is based on the size of the found window, exploiting that objects further from the camera appear smaller.

3. USING TRACKING FOR MOTION PARALLAX

Once the position of the head is known, updates of the virtual camera can be performed to acquire 3D vision.

In a setup without motion parallax, representing a 3D scene by using perspective projection is straightforward. A position of the eye and a projection plane, or image plane, are chosen. All objects are projected onto the image plane. This is demonstrated in Figure 2. The screen is typically at the same position of the image plane, and the eye is placed symmetrically, i. e. the eye is projected on the middle of the image plane when projected along the normal of the plane.

When using motion parallax, the eye is moved, while the image plane stays fixed. This will result in a non-symmetrical projection. This is demonstrated in Figure 3. Simulating motion parallax can be accomplished by adapting the frustum of the render camera. The frustum (see Figure 4) is controlled by 6 parameters: the coordinates of the topleft and bottomright corners of the image plane, and the depths of the near and far planes. The X and Y coordinates control the first four coordinates directly, while the Z coordinate controls the depths of the planes.

To cope with noise in the tracker data, all coordinates of the head are smoothed using a running average. This will reduce jittering in the final result.

4. RESULTS

To allow the use of the previously mentioned method in web-based applications, we only used conventional web-based technologies. WebGL is used for rendering and WebRTC is used for capturing camera images. To test our setup, we implemented a 3D environment using basic geometric shapes to emphasize the 3D effect. We tested our setup on a desktop, a laptop, and a tablet.

Our desktop setup can be seen in Figure 5. A camera is placed above the screen to allow an optimal view of the face of the user. We used a Logitech Quickcam, demonstrating that no high-end cameras are required. For our laptop setup, we used the built-in camera. The camera is initialized such that the color images are consistent and not over- or underexposed. Exact color calibration is not required.

To demonstrate our method, another camera is placed close to the eyes of the user. This camera will effectively show the screen image, as seen from the viewpoint of the user. A few examples are shown in Figures 6, 7, and 8. As can be seen, the viewpoint is effectively adapted to the position of the viewer. Motion parallax is thus acquired. A frame rate of 49 Hz is achieved, allowing the allocation of more processing power to rendering, while retaining real-time tracking.

To demonstrate the portability of our method, we also applied the method on a mobile device. We used the Samsung GT-N8010, equipped with a $1.4GHz$ Quad Core CPU, an ARM Mali-400 GPU, and a 1.9 Megapixels front facing camera. We used Firefox 24.0. The setup is shown in Figure 9. As can be seen, the viewpoint can be effectively adapted using the front facing camera. However, due to the limited processing power, the framerate is limited to $16Hz$, which results in a reduced user experience.

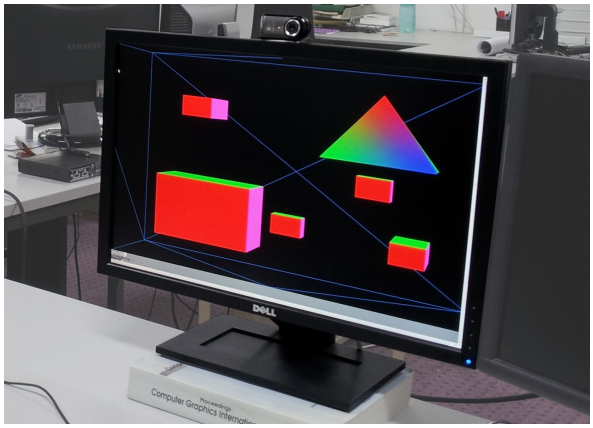

Fig. 5. The desktop setup. A low-end webcam is used to demonstrate the effectiveness of the method.

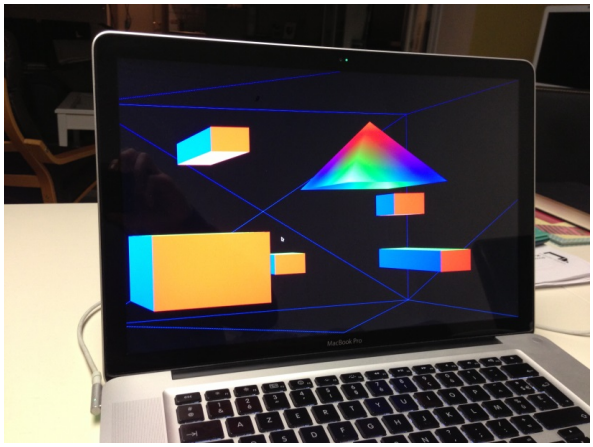

Fig. 6. Example of the scene as seen from the viewpoint of the user. The scene is adapted to accommodate motion parallax.

5. CONCLUSION

In this paper, we demonstrated a method to perform real-time head tracking using the camshift algorithm to acquire motion parallax. Motion parallax proved to be a good 3D cue, allowing 3D vision on a variety of devices, without specialized hardware. Tests performed on a number of devices demonstrated the portability and speed of our system.

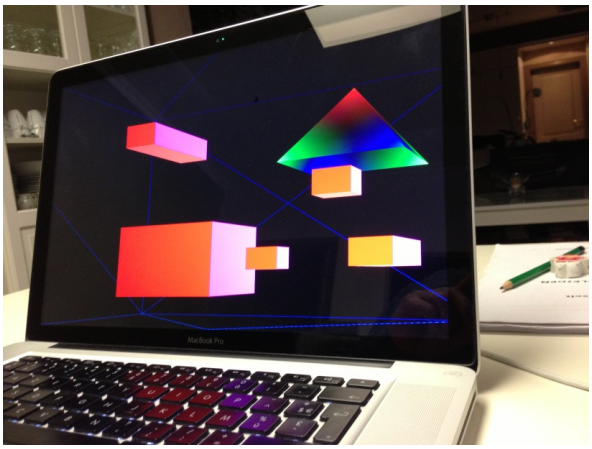

Fig. 7. Example of the scene as seen from the viewpoint of the user. The scene is adapted to accommodate motion parallax.

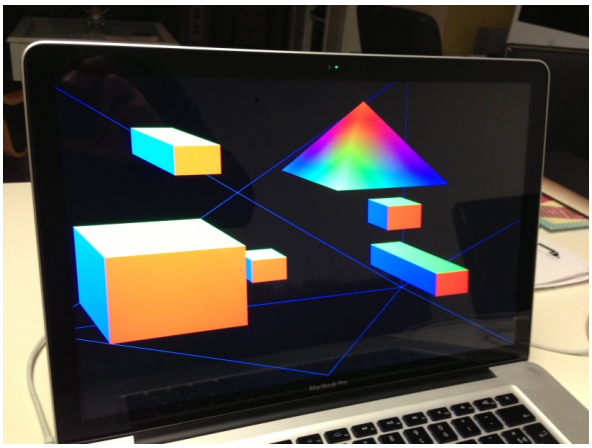

Fig. 8. Example of the scene as seen from the viewpoint of the user. The scene is adapted to accommodate motion parallax.

Fig. 9. Example of the mobile setup. Motion parallax is achieved on a tablet, using the front facing camera.

Acknowledgments

Patrik Goorts would like to thank the IWT for its PhD specialization bursary.

6. REFERENCES

[1] Kurt Akeley, Simon J Watt, Ahna Reza Girshick, and Martin S Banks. A stereo display prototype with multiple focal distances. In *ACM transactions on graphics (TOG)*, volume 23, pages 804–813. ACM, 2004.

[2] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.

[3] Neil A Dodgson. Optical devices: 3d without the glasses. *Nature*, 495(7441):316–317, 2013.

[4] Masaki Emoto, Takahiro Niida, and Fumio Okano. Repeated vergence adaptation causes the decline of visual functions in watching stereoscopic television. *Display Technology, Journal of*, 1(2):328–340, 2005.

[5] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Access Online via Elsevier, 1990.

[6] David M Hoffman, Ahna R Girshick, Kurt Akeley, and Martin S Banks. Vergence–accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of vision*, 8(3), 2008.

[7] Anshul Jain and Qasim Zaidi. Discerning nonrigid 3d shapes from motion cues. *Proceedings of the National Academy of Sciences*, 108(4):1663–1668, 2011.

[8] David E. Johnson. Projection and view frustums, 2013.

[9] Helmut Jorke and Markus Fritz. Infitec-a new stereoscopic visualisation tool by wavelength multiplex imaging. *Proceedings of Electronic Displays, September*, 2003.

[10] Frank L Kooi and Alexander Toet. Visual comfort of binocular and 3d displays. *Displays*, 25(2):99–108, 2004.

[11] Marc Lambooij, Marten Fortuin, Ingrid Heynderickx, and Wijnand IJsselsteijn. Visual discomfort and visual fatigue of stereoscopic displays: a review. *Journal of Imaging Science and Technology*, 53(3):30201–1, 2009.

[12] Johnny Chung Lee. Hacking the nintendo wii remote. *Pervasive Computing, IEEE*, 7(3):39–45, 2008.

[13] Sheng Liu and Hong Hua. Time-multiplexed dual-focal plane head-mounted display with a liquid lens. *Optics letters*, 34(11):1642–1644, 2009.

[14] Gordon D Love, David M Hoffman, Philip JW Hands, James Gao, Andrew K Kirby, and Martin S Banks. High-speed switchable lens enables the development of a volumetric stereoscopic display. *Optics express*, 17(18):15716, 2009.

[15] Kevin J MacKenzie, Ruth A Dickson, and Simon J Watt. Vergence and accommodation to multiple-image-plane stereoscopic displays:real world responses with practical image-plane separations? *Journal of Electronic Imaging*, 21(1):011002–1, 2012.

[16] Kevin J MacKenzie, David M Hoffman, and Simon J Watt. Accommodation to multiple-focal-plane displays: Implications for improving stereoscopic displays and for accommodation control. *Journal of Vision*, 10(8), 2010.

[17] Brian Rogers and Maureen Graham. Motion parallax as an independent cue for depth perception. *Perception*, 8(2):125–134, 1979.

[18] Sammy Rogmans, Maarten Dumont, Gauthier Lafruit, and Philippe Bekaert. Biological-aware stereoscopic rendering in free viewpoint technology using gpu computing. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pages 1–4. IEEE, 2010.

[19] Takashi Shibata, Joohwan Kim, David M Hoffman, and Martin S Banks. The zone of comfort: Predicting visual discomfort with stereo displays. *Journal of vision*, 11(8), 2011.

[20] Maurice HPH van Beurden, Andre Kuijsters, and Wijnand A IJsselsteijn. Performance of a path tracing task using stereoscopic and motion based depth cues. In *Proceedings of the Second International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 176–181. IEEE, 2010.

[21] Günter Wyszecki and Walter Stanley Stiles. *Color science*. Wiley New York, 1982.

[22] Jie Yang and Alex Waibel. A real-time face tracker. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, pages 142–147. IEEE, 1996.

[23] Ming-Hsuan Yang and Narendra Ahuja. Detecting human faces in color images. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, volume 1, pages 127–130. IEEE, 1998.