Temporal Reprojection Anti-Aliasing

Christian Alexander Oliveros Labrador*

Lund University
Sweden

Abstract

In this essay, I describe the implementation of Temporal Reprojection Anti-Aliasing (TRAA) I implemented for the EDAN35 High Performance Computer Graphics project. It is based on the Pedersen TRAA article for their game, Inside [Pedersen 2016], and on the Xu TAA article for their game, Uncharted 4 [XU 2016].

1 Introduction

The Aliasing problem that computer graphics experience comes from not being able to sample as required by the Nyquist-Shannon Sampling Theorem, creating ragged edges that appear in the rasterization process (Spatial Aliasing) and jumps between moving objects (Temporal Aliasing), according with Doggett and Wronski [Doggett 2017; Wronski 2014]. Many solutions have been proposed and used to solve it, e. g. the Super Sampling Anti-Aliasing (SSAA) family of solutions that work on higher frequencies than the required at the cost of more space requirements.

The motivation of this project is to implement a solution from the Temporal Reprojection Anti-Aliasing family, which is relatively new and promises to solve spatial and temporal aliasing problems in post processing [Pedersen 2016] using less memory requirements than the SSAA family while keeping the same quality. This is done using the variation between the current frame and past ones to refine the output image.

2 Algorithms

Camera Jitter; Velocity Buffer; Frame History Buffer; Clipping Color Box; Sharpen Filter; and Motion Blur are used to implement the TRAA.

2.1 Camera Jitter

Camera Jitter is applied every frame to preserve information from local regions of surface fragments. If the current frame is static relative to the past ones then the system is loosing information that could use to refine it. [Pedersen 2016; XU 2016]

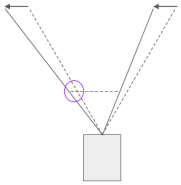

Figure 1: Jittering the Projection Matrix. Image taken from [Pedersen 2016]

The jittering is applied as a translation to the projection matrix using the Halton Sequence (2,3) as the translation deltas. This sequence is used because it is better to have an irregular pattern for the translations [Pedersen 2016; XU 2016].

*e-mail: christianol_01@hotmail.com

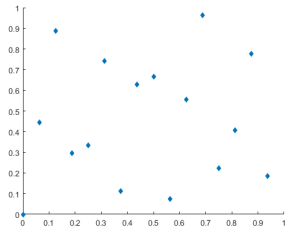

Figure 2: Values from the Halton Sequence (2,3) used

The Figure 2 is the representation of the 16 points used to jitter the projection in my implementation, as proposed by Pedersen [Pedersen 2016]. It was generated using MATLAB with the command haltonset(2) then scrambled using reverse-radix scrambling, scramble(p, 'RR2') and, finally, generated the 16 points used with net(p, 16).

2.2 Velocity Buffer

The Velocity Buffer algorithm used in this implementation is the one proposed by Chapman [Chapman 2012] which is calculated by subtracting in NDC space the current pixel position by its last frame position. This is possible by saving the MVP matrix of each object in the scene.

Also, as suggested by Xu [XU 2016], the jittering is not included as part of the motion.

2.3 Frame History Buffer

For each fragment in the current frame we look for the 3x3 neighborhood and plus (+) pattern neighborhood (See Figure 3). On both patterns we look for the minimum and maximum of colors of the current frame, later we average them and use them in the Clipping of History [Pedersen 2016].

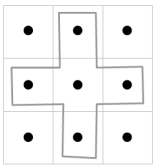

Figure 3: Sampling Pattern used. Image taken from [Pedersen 2016]

On the 3x3 neighborhood we look for the velocity of the pixel with the closest depth, this is to get better edges in motion for pixels that are occluded [Pedersen 2016]. We use this velocity to reproject the position of the current frame in the history [Pedersen 2016; XU 2016].

After we have the history we constrain it (See next subsection) and we mix it with the current frame. We linearly mix both of them using a feedback value thats calculated by the difference of luminance between colors. This feedback is clamped between values

closer to one to add some information of the current frame while keeping the history. This mix stabilizes the image, removing the jittering and smoothing the edges [Pedersen 2016; XU 2016]. Because history is accumulated, we get the effect that each frame weights less the more time the history is not rejected [Pedersen 2016].

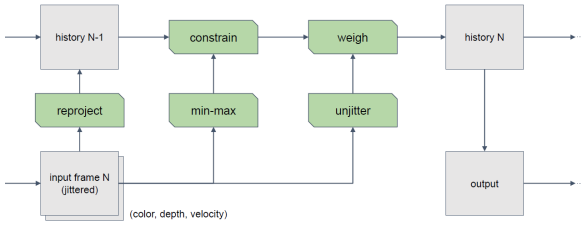

Figure 4: Temporal Reprojection Anti-Aliasing process. Image taken from [Pedersen 2016]

2.4 Clipping Color Box

To handle color rejection when history is too distant from current color a Clipping Color Box is used. This is a box built using the current pixel color as the center and the minimum and maximum color calculated in the last subsection as limits. The history color is taken as a position and projected against the limits of the box if it lies outside, else, it is left untouched. The use of the Clipping Color Box is to prevent color clustering that would happen if Clamp is applied (See Figure 5) [Pedersen 2016].

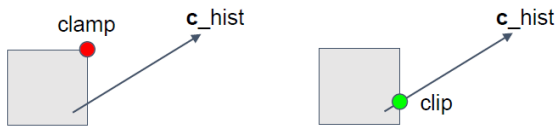

Figure 5: Color Clamping versus Color Clipping. Image taken from [Pedersen 2016]

2.5 Sharpen Filter

Because the Reprojection process and Color Clipping create blurriness, a Sharpen Filter is required. I used the one proposed by Xu [XU 2016].

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figure 6: Sharpen Filter Convolution Matrix. As used in [XU 2016]

2.6 Motion Blur

Because the nature of the History Buffer, ghosting is created by fragments from objects that move so fast that they are not rejected as quickly as necessary under special lighting and background conditions. The proposed solution by Pedersen and Xu [Pedersen 2016; XU 2016] is to use Motion Blur to hide this artifacts.

The Motion Blur used is the one proposed by Chapman [Chapman 2012]. It tries to behave like a real camera by scaling the velocity of each pixel by the division of the current FPS to the one wanted, thus, simulating the shutter speed. Then it mixes the colors of the pixels that are sampled while following the direction of the velocity buffer vector.

3 Results

Results are shown in the next figures. All of them are renders of the Sponza scene.

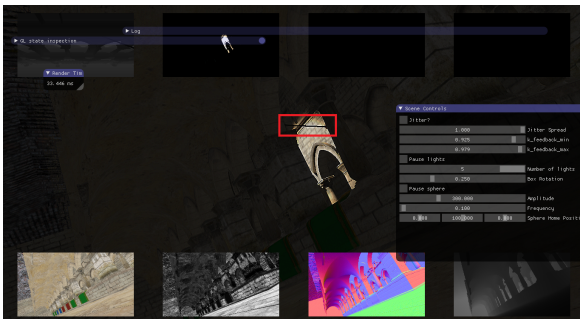

Figure 7: Bar Rendered without Anti-Aliasing

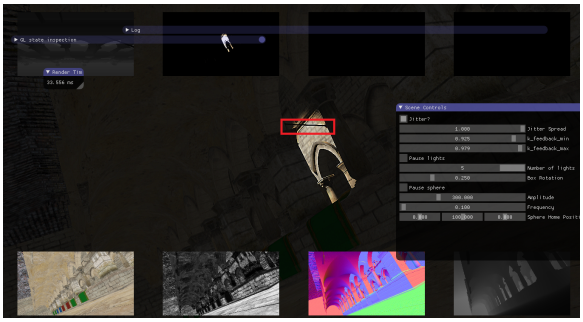

Figure 8: Bar Rendered with TRAA

Figure 9: Bar Rendered without Anti-Aliasing Zoomed

Figure 10: Bar Rendered with TRAA Zoomed

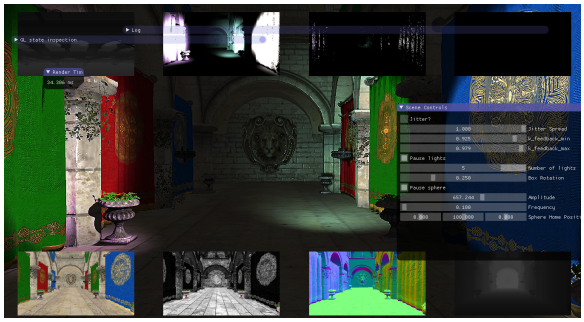

Figure 11: Sponza Render without Anti-Aliasing

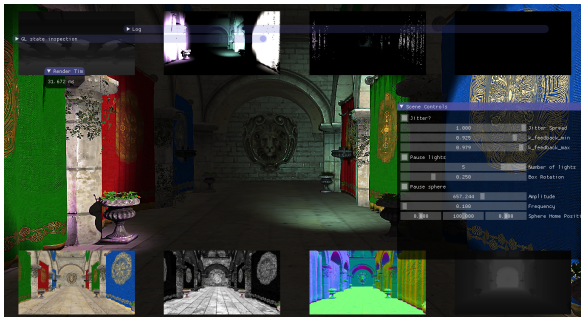

Figure 12: Sponza Render with TRAA

As we can see, the TRAA smoothes the edges of the images without a loss of quality or performance.

4 Discussion

Current advances in TRAA allows its use in real time computer graphics without a great loss of performance while achieving good quality rendering. But it contain artifacts that affect the quality of the render.

4.1 Blurriness

The current implementation of TRAA generates a very aggressive blur, needing a Sharpen Filter.

I tried the Sobel Filter to control the blend of the current frame and the history based on where the edges where, but it created too many jittering artifacts. Also, I tried with a 5x5 Sharpen Filter but it changed the brightness too much.

The Sharpen Filter that was finally used in the implementation is the one used by Xu [XU 2016], it solves the blurriness really well but it can not eliminate some artifacts. For future implementations I propose to try other Sharpen Filters, maybe one with a bigger kernel. Also, it might be possible to use the Sobel Filter to contrast edges.

4.2 Jittering

The current implementation of TRAA is fast to stabilize the image and cancel the jittering. But, it still has problems handling Specular Reflections that are close to the size of just one pixel. I propose to solve jittering by blurring a little the Specular Texture in the Deferred Resolve Pass to avoid pixel size Specular Reflections.

4.3 Ghosting

Some Ghosting is created when animation is done in particular special lighting and background conditions. This is partially corrected with motion blur but requires more processing to completely solve it completely. Xu proposes the use of Motion Blur and increase the size of everything by the use of an Stencil technique [XU 2016]. This is not used for the current implementation.

Pederson implementation allows the jitter in the Velocity Buffer calculations but it creates some unwanted blurriness [Pedersen 2016]. This could be another possible solution if the Sharpen Filter can remove that blurriness afterwards.

5 Conclusion

In conclusion Temporal Reprojection Anti-Aliasing is a good technique to solve the Aliasing problem in real time without incurring in heavy space or time requirements. Further research is needed to improve the image quality and reduce the ghosting, jittering and blurriness artifacts.

References

CHAPMAN, J., 2012. Per-Object Motion Blur. `http://john-chapman-graphics.blogspot.se/2013/01/per-object-motion-blur.html`, September. Accessed: 2017-11-30.

DOGGETT, M. 2017. Anti-aliasing. *EDAN35 High Performance Computer Graphics* (November).

PEDERSEN, L. J. F. 2016. Temporal Reprojection Anti-Aliasing in INSIDE. *GDC Vault*. Accessed: 2017-11-28.

WRONSKI, B., 2014. OpenGL GLSL - Sobel Edge Detection Filter. `https://bartwronski.com/2014/03/15/temporal-supersampling-and-antialiasing/`, March. Accessed: 2017-12-01.

XU, K. 2016. Temporal Antialiasing In Uncharted 4. *SIGGRAPH 2016*.