TIDOAA — A Temporal Anti-aliasing Method with Immutable Dynamic Objects

Lev Smirnov[1], Vladimir Frolov[1,2] and Vladimir Galaktionov[2]

[1]Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia
[2]Keldysh Institute of Applied Mathematics, Miusskaya sq., 4, Moscow, 125047, Russia

Abstract

The main purpose of this investigation is to use information from previous frames for partial synthesis of the new one. Algorithms utilizing already calculated information have become an integral part of many interactive applications in recent years, and using of such methods allows reduces the computational load on the GPU without significantly degrading the image quality. Despite its popularity, there are still serious unresolved issues that affect image quality in many scenarios. Especially acute are the issues of artifacts and frame distortions. This paper discuss ways of counting and using information from previous frames used in image rendering applications, and proposes a new algorithm which combine best traits of the previous ones.

Keywords

Temporal smoothing, real-time rendering, anti-aliasing

1. Introduction

Throughout the history of graphics, people have been striving to get images that are indistinguishable from what we see in real life. Rendering of this level is possible only if all the physical laws of light propagation in space are observed. This turns the task of creating an image into the task of completely modeling the space in front of the camera, that is, it leads to an increase in calculations for each pixel of the image. Even now, space modeling tasks are not always solvable on supercomputers, as they require a large number of calculations. Therefore, simplifications, heuristics, and space interpolation methods are used to render images. The growth of computing power makes it possible to use more complex and approximate to real physics techniques. So over the past 40 years, the realism of the render has improved significantly.

This improvement has been achieved not only due to the growing computing power, but also due to the complexity of the synthesis technique and post-processing of frames. One of these techniques is the method of generating a frame using additional information obtained from previous frames. If nothing has changed in the scene or has changed slightly, we do not need to completely synthesize the frame again, we can show an image partially created in the previous iteration. Also, the released calculations can be directed to improvement (refinement)

GraphiCon 2022: 32nd International Conference on Computer Graphics and Vision, September 19–22, 2022, Ryazan State Radio Engineering University named after V.F. Utkin, Ryazan, Russia
lev.smirnov@graphics.cs.msu.ru (L. Smirnov); vfrolov@graphics.cs.msu.ru (V. Frolov); vlgal@gin.keldysh.ru (V. Galaktionov)
0000-0001-5385-4841 (L. Smirnov); 0000-0001-8829-9884 (V. Frolov); 0000-0001-6460-7539 (V. Galaktionov)

an already existing frame, for example, with a small shift within a pixel. Since frame synthesis is the sampling of a function, a half—pixel shift is equivalent to the sampling of a function with a step of two times less, which improves the quality of the resulting image.

2. Related work

To date, there are many methods that use temporal information. Most of them are used for smoothing objects (also known as Temporal Anti Aliasing or TAA), i.e. they remove the gradation of borders (Fig. 1). Initially, the term ”temporal smoothing” was used for methods aimed at reducing the stroboscopic effect [1]. At the moment, the concept of spatial smoothing using time samples is clearly fixed for this term.

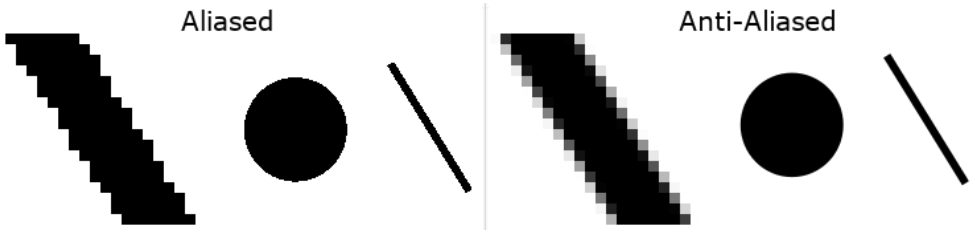


Figure 1: Example of a aliasing in a graphics

Before using temporary smoothing methods, hardware acceleration of MSAA (Multisample Antialiasing) [2] was the most popular smoothing method in real-time algorithms. Unfortunately, there are a number of problems when using it in combination with the methods of deferred shading [3, 4]. Nevertheless, the MSAA method is considered to be the closest to the qualitative smoothing results. Therefore, the images obtained using MSAA in the work will be used to verify the proposed method.

The first methods using temporal coherence of information (that is, the connection between neighboring frames) sought to reuse the results of calculations in several frames in order to save on expensive calculations. These methods took a certain number of previous frames separately and used the distortion of the geometry of the scene to reproject the data [5]. Another group of methods, often referred to as motion-compensated filtering, used space-time filters to reduce noise in image sequences [6].

Despite the fact that the theoretical idea of using data from several frames to refine pixels existed back in the 90s [7], this idea became possible for real-time rendering only after significant technological improvements in computing devices. In the works Daniel Scherzer and Diego Ne Hub [8, 9] describes a method for reprojecting pixel-by-pixel vectors during a direct rendering pass. The approach tracks the movement of the camera between frames and allows you to reproject each pixel from the previous frames into the current one. In both papers, it was proposed to use this technique to improve the construction of shadow maps. This idea underlies almost all existing TAA methods. In an article titled ”Amortized Supersampling” in 2009, [10] for the first time proposed using data projection for smoothing. The authors identified two main difficulties in TAA for obtaining high-quality images: excessive spatial-temporal blurring

due to oversampling error and slow adaptation to major signal changes. As a solution, it was proposed to store temporal information in a buffer of higher resolution, by means of theoretical analysis, the authors of the work proposed to store temporal data in a 2-fold target resolution compared to the original image.

An important idea that makes TAA more reliable in applications is to use current frame samples to correct redesigned historical data. A similar idea was first presented in the work of Lottom [11]. This approach has been improved in subsequent works by [12, 13]. The method prioritizes data from the current frame, and information from previous frames is taken with some weights. The approach proposed in the articles removes some artifacts caused by outdated data. This significantly reduces the need for other history checking heuristics and additional input data, making TAA more reliable.

In a 2009 paper, Lei Yang and Diego Ne hub[10] proposed an algorithm for accumulating details at the subpixel level in TAA. a combined spatially[14] was introduced, a combined spatio-temporal filter was introduced to perform upsampling in image space. Similar ideas have been implemented in game engines to meet the rapidly growing demand for high resolution, which is reflected in [15] and Unreal Engine [16].

An absolutely different approach to the processing of temporal information is presented by the article "Adaptive temporal smoothing" [17]. It divides the image according to the amount of time information for each area of the image, i.e. for each pixel of the new image, an affiliation class is selected. This separation allows you to understand exactly which areas of the frame require the greatest number of calculations.

Methods based on neural networks are also emerging. So in the work of Lei Xiao [18], two neural networks are used. One of them builds a map of projection vectors from one frame to another. The second one checks and rejects pixels transferred from the previous four frames. Another approach is the use of neural networks to fill pixels with indeterminate temporal information, such as the recently known DLSS algorithm.

2.1. Results of the review

Summing up all of the above, all existing methods can be divided according to two criteria. The first of them is a method of transferring information from previous frames to the current one. According to him , the methods are divided into:

- Counting the germination of information
- Neural network algorithms
- Calculation when drawing a scene

Methods for calculating the similarity of information appeared among the first, but at the same time they are the most inaccurate. Neural network algorithms have appeared relatively recently and show good projection quality, but they need to implement non-trivial validation algorithms and possibly retraining on a training sample of scenes. The most common method is to build a projection based on the information obtained when drawing the scene. It is this method that is used in the proposed work.

The second criterion is the method of storing temporal information

- Each frame is stored separately
- A single repository for all temporal information

When storing each frame separately, it is necessary to calculate the time coherence for all stored frames, which increases the necessary calculations. With a single storage, coherence needs to be calculated once, but there is a chance of losing information from old frames. When analyzing the pros and cons, a single buffer approach was chosen.

3. Proposed method

The proposed method is designed to use time data to calculate an image in a higher resolution to smooth the result. The various existing ideas discussed in paragraph 2 are taken as a basis, the best ones are selected experimentally, and new approaches are added to them for the correct projection of dynamic objects that do not change shape. The algorithm contains three main stages: projection of data from previous frames, addition of new information and calculation of the final pixel color. Which are repeated every frameы (Fig. 2).

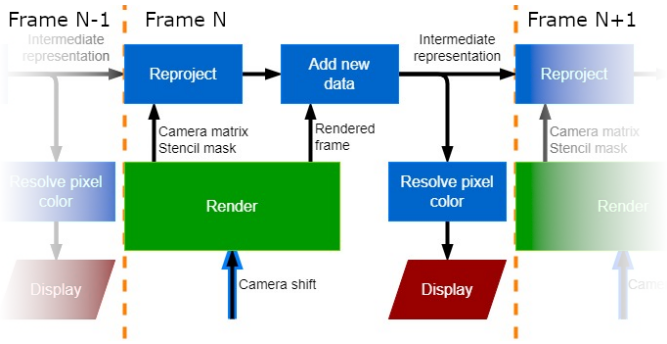


Figure 2: Scheme of the proposed method

3.1. Temporal data storage

To improve the quality of smoothing, it is proposed to store information in a buffer of higher resolution [10]. In this paper, it was decided to use a buffer in 2-fold resolution, i.e. each pixel of the original image corresponds to 4 pixels in the intermediate representation (Fig. 3). Each pixel belongs to the RGBA color space, where the RGB components store the color of temporary information, and the A component contains a number indicating the degree of information obsolescence. The smaller the number in A component, the less valid the time information, i.e. 1.0 means absolutely new information, 0.0 — not valid for use. Also, a depth map for each pixel and the map obtained from the template mask are stored in a separate buffer. This information is necessary for reprojection, described in more detail in paragraph 3.2.2.

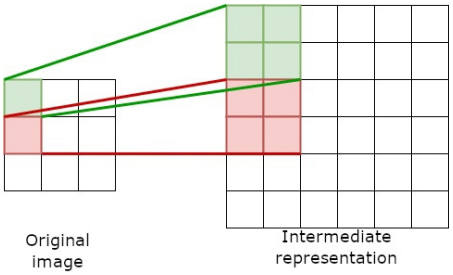


Figure 3: Scheme of data storage

3.2. Reprojection

When the scene moves between frames, motion vectors must be calculated for each pixel in order to obtain temporal information. These vectors determine the pixel location offsets between frames. A similar transformation must be calculated for each pixel of the image.

3.2.1. Theoretical basis

To calculate the transformation, it is necessary to understand how the transition from the local coordinates of the object to the screen space occurs. Where the local coordinates are the coordinates of the object measured relative to the reference point located where the object itself begins. The transitions are illustrated in Fig. 4:

1. Local coordinates are transformed into coordinates of the world space using the model matrix. The world coordinates are in the sense of coordinates of a larger world. These coordinates are measured relative to the global reference point, which is the same for all other objects located in the world space.
2. The world coordinates are transformed into the coordinates of the view space using the view matrix. Thus, each vertex is viewed as if it were viewed from a camera.
3. The coordinates are transformed into a clipping space using a projection matrix. The Clipping coordinates are valid in the range from -1.0 to 1.0 and determine which vertices will appear on the screen. Thus, we cut off unnecessary objects.
4. By transforming to the viewport, we convert the clipping coordinates to the normalized coordinates of the device by dividing by the fourth component of the vector and leave all values that are in the range from -1.0 to 1.0. This the range is called the area of the screen coordinates.

Let's describe all these transformations using mathematical formulas. To do this, a system of designations is introduced. M denotes a matrix of dimension 4 by 4 elements, V is a vector of four elements (x, y, z, w), with V_i taking the components of i vector, where $i = \{x, y, z, w\}$. The components x, y, z are responsible for the spatial position, and the component w is auxiliary in nature and is equal to the base vectors 1.

The mathematical formula for the transition from the space of local coordinates to the cutoff coordinates will look like this:

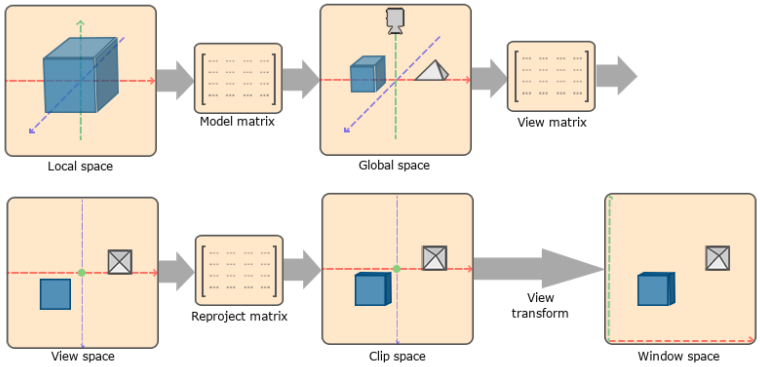


Figure 4: Diagram of the transition from the local coordinates of the object to the screen space

$$V_{clip} = M_{projection} M_{view} M_{model} V_{local} \tag{1}$$

Where $M_{projection}$, M_{view}, M_{model} are the projection, view and model matrices respectively. V_{clip} is a vector of coordinates in the clipping space, and V_{local} is a vector of local coordinates.

The formula for obtaining a vector in the normalized coordinates of the device

$$V_{NDC} = \frac{V_{clip}}{(V_{clip})_w} \tag{2}$$

From the formulas 1 and 2 we get their temporal analogues, where n means that the data belongs to n frame. Note that the local coordinates of the object V_{local} do not depend on the frame number:

$$V_{clip,n} = M_{projection,n} M_{view,n} M_{model,n} V_{local} \tag{3}$$

$$V_{NDC,n} = \frac{V_{clip,n}}{(V_{clip,n})_w} \tag{4}$$

To simplify the entries , we denote:

$$M_{pvm,n} = M_{projection,n} M_{view,n} M_{model,n}$$

Using the formula 3, we express V_{local}:

$$V_{local} = M^{-1}_{model,n} M^{-1}_{view,n} M^{-1}_{projection,n} V_{clip,n} = M^{-1}_{pvm,n} V_{clip,n} = M^{-1}_{pvm,n-1} V_{clip,n-1} \tag{5}$$

Based on 5, we can find a correlation between $V_{clip,n}$ and $V_{clip,n-1}$:

$$V_{clip,n} = M_{pvm,n} M^{-1}_{pvm,n-1} V_{clip,n-1} \tag{6}$$

Let's use the formula 4 and 6:

$$V_{NDC,n} = \frac{V_{clip,n}}{(V_{clip,n})_w} = \frac{M_{pvm,n} M^{-1}_{pvm,n-1} V_{clip,n-1}}{(M_{pvm,n} M^{-1}_{pvm,n-1} V_{clip,n-1})_w}$$

We found a correlation between the coordinates of the previous frame and the coordinates of the current one, but to reduce the size of the transmitted data, we simplify the resulting expression.Let's open $V_{clip,n-1}$ in the right part using the 4. Note that $(V_{clip,n-1})_w$ is a scalar, and we can take it out from under the brackets in the denominator, and then shorten it, we get:

$$V_{NDC,n} = \frac{M_{pvm,n}M^{-1}_{pvm,n-1}V_{NDC,n-1}(V_{clip,n-1})_w}{(M_{pvm,n}M^{-1}_{pvm,n-1}V_{NDC,n-1}(V_{clip,n-1})_w)_w} = \frac{M_{pvm,n}M^{-1}_{pvm,n-1}V_{NDC,n-1}}{(M_{pvm,n}M^{-1}_{pvm,n-1}V_{NDC,n-1})_w} \quad (7)$$

Using the formula 7 we can calculate where the normalized coordinates of the device have moved from the previous frame to the current one. Since the normalized coordinates are in the range from -1.0 to 1.0, and we only have a pixel with the cordinates x, y and the depth value d on the screen with a width of W and a height of H, then we need to bring them into the normalized coordinates of the device. The formula for the calculation looks like this:

$$V_{NDC} = (\frac{2x}{W} - 1, \frac{2y}{H} - 1, d, 1) \quad (8)$$

3.2.2. Types of transformations

During rendering, all objects are divided into types:

- Dynamic object with shape change
- A dynamic object without changing its shape
- Static object

For pixels belonging to dynamic objects with shape changes (such as skeletal objects), we will not apply projection. This is due to the fact that, although the object may not change its position in world coordinates, it changes. With such changes, the transformation we calculated is not correct, and this will generate artifacts in the image, for example, a ghostly trail behind the object. To avoid such an effect, information about such objects is used only from the current frame, that is, without taking into account the time frame. For pixels belonging to dynamic objects without changing the shape, we apply the standard transformation described in section 3.2.1 For the third type of objects, the transformation can be simplified. Note that for static objects it is true:

$$M_{model,n} = M_{model,n-1} \Rightarrow M^{-1}_{model,n} = M^{-1}_{model,n-1}$$

Then for static objects:

$$M_{pvm,n}M^{-1}_{pvm,n-1} = M_{projection,n}M_{view,n}M_{model,n}M^{-1}_{model,n-1}M^{-1}_{view,n-1}M^{-1}_{projection,n-1} =$$

$$M_{projection,n}M_{view,n}M^{-1}_{view,n-1}M^{-1}_{projection,n-1} \quad (9)$$

For simplicity of writing , we denote:

$$M_{pv,n} = M_{projection,n}M_{view,n}$$

From the formulas 9 and 7 we get:

$$V_{NDC,n} = \frac{M_{pv,n} M_{pv,n-1}^{-1} V_{NDC,n-1}}{(M_{pv,n} M_{pv,n-1}^{-1} V_{NDC,n-1})_w} \tag{10}$$

Since most of the screen is occupied by pixels belonging to static objects, we have greatly simplified the calculation process for the entire frame.

3.2.3. Attenuation of information

While reprojecting information, it is necessary to check its validity, since temporal information may become outdated. For these purposes, an intermediate representation uses A component for each pixel. It contains a number showing how much you can trust the information of this pixel. To reproject pixel A, it must be above the threshold value. In the proposed method, a threshold of 0.1 was selected during the experiments. If a pixel satisfies this condition, it is reprojected, and the value of the old one multiplied by the coefficient of forgetting information α is recorded in the A component of the new position. Experimentally, the best α turned out to be 0.5. Similarly, the depth buffer and the template mask are projected. There may be a situation when there are pixels from different objects in one block. In this case, only pixels with a large value in component A are subject to projection, since it is possible to move part of the second object to the wrong position. The problem of overlapping objects is solved in a similar way.

3.3. Add new data

After reprojecting, it is necessary to add new data from the current frame to the intermediate representation. This is necessary for validation and preservation of the temporal consistency of the information. This stage is divided into two sub-stages, since it is necessary to change the camera position and mix existing data with new ones.

3.3.1. Camera offset

In situations where the camera position is stationary relative to the scene, we sample the same points of space. This approach completely negates the idea of the temporal method, since the information from the previous frames is equal to the information received in the current one. The same problem occurs in some situations when the camera is rotated. As a solution, half-pixel camera offsets are used for each pixel. Although the pixel is the smallest element of the image, in fact, information of some function that falls into this area is discretized into it. With standard rendering, one sample per pixel is taken. It is the resulting value that will be displayed in the image. Ideally, it is necessary to average the entire function in this area. For example, many samples are used to approximate sampling, this is how MSAA[2] works. Several samples are taken from a pixel based on the selected pattern. Similarly, we will shift the camera to get new samples per pixel, even if the camera is not moving. There are many patterns of sampling overlap. In the proposed method, it was experimentally decided to focus on the 2 x 2 RGSS pattern [19], since it is one of the most common, while it gives full coverage of samples in a pixel. Based on this pattern, small shifts are made within the pixel for the camera and new samples are obtained. The shifts are bypassed in a certain order: left-upper, right-lower,

right-upper, left-lower. This is done in order to capture more information about the pixel in fewer samples, which ultimately increases the convergence rate.

3.3.2. Mixing data

Based on the known direction of the camera displacement, the filling of the intermediate representation is selected. This is more clearly demonstrated in Fig. 5. If there was valid information in the cell before the record, we replace it, thereby providing another way to update the information, in which newer data gets a higher priority. In a similar way, the buffer that stores the depth and the template mask used for dynamic objects is filled.

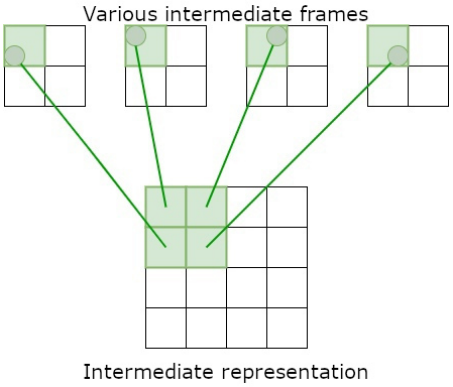


Figure 5: The process of adding new data. Depending on the offset during rendering, the value is written to various pixels of the intermediate representation

3.4. Counting the final color of pixels

Each pixel in the final image corresponds to a square of four pixels in the intermediate representation. The last step is to calculate the average color of these pixels, and it will be the final color of the pixel. In Fig. 6 shows how the color is obtained when averaging. On the left are four pixels of the intermediate representation corresponding to one pixel of the original image. When averaging them, a new color is obtained, which will be recorded in the final image, it is displayed on the right.

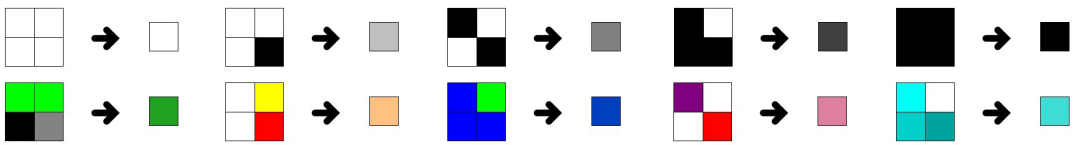


Figure 6: Examples of counting the final pixels of an image. The pixels of the intermediate representation are shown on the left, on the right which resulting color is obtained during averaging

When building an intermediate representation, we may get invalid or outdated data. We can check them again thanks to the value that lies in the component A. When receiving an

average color, such pixels with A component less than 0.1 are not used, since we believe that corrupted information is stored in them. At the same time, based on how the process of adding new data occurs, for any pixel of the final image, at least one of the four corresponding pixels in the intermediate representation has a value greater than 0.1 in the component. Therefore, we will always be able to calculate the average color of the final pixel.

4. Experiments

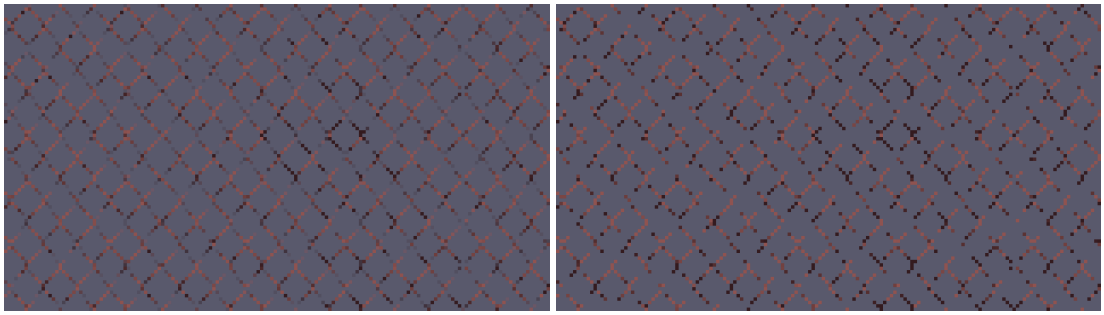
This section demonstrates the operation of the proposed algorithm. The comparison takes place according to several criteria. The effectiveness of the anti-aliasing method was chosen first for comparison. The figure 7 shows the comparison of the proposed method with a graphical pipeline without the use of TAA. In the experimental scene there is a model of a hut around which the camera rotates at a fixed speed.

(a) Using the proposed method (b) Without using the method

Figure 7: Comparison of the proposed method

The operation of the method is shown more clearly when displaying small details and fine lines. When sampling such objects, some of the samples may not get into them, this is expressed in the fact that some parts of the object disappear. This is especially evident in dynamics, as flickering objects are generated, which is very noticeable in the sequence of frames. Using temporal data during rendering restores some of the information about such objects. For experiments, a model of a fence was taken that approaches the camera, the camera itself is fixed on a static position. An example of restoring the bars of the grid is shown in the figure 8.

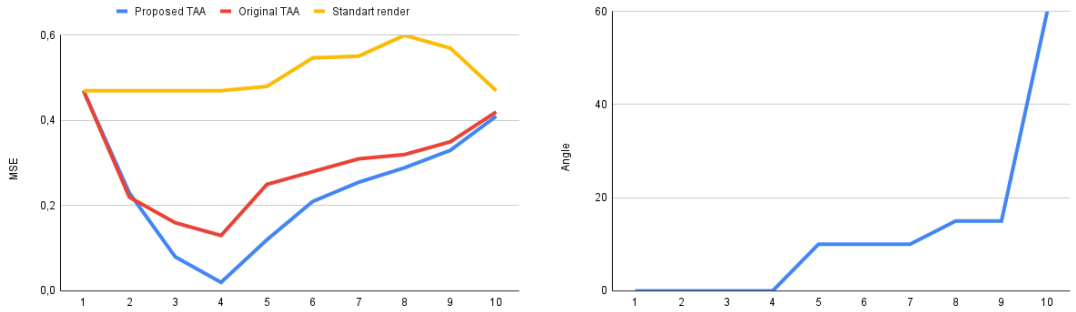
To objectively compare the work of the proposed method, the result of the work of MSAA[2] was used as a reference. That is, in this experiment, it is compared how different the images



(a) Using the proposed method (b) Without using the method

Figure 8: Comparison of dynamic object boundaries

obtained by different methods are relative to the images obtained using MSAA. The method without TAA, the original TAA and the proposed method were chosen for experiments. To demonstrate the quality of the method, the first scene with a hut was chosen, in which dynamic objects that do not change shapes, rotating around it, were added. The camera rotates between frames at fixed angles, creating areas of discovery about which time methods have no information. The result of measurements on 10 frames is shown on the graph 9.



(a) A graph of the similarity of methods to MSAA by the MSE metric. (b) The angle of rotation of the camera between frames.

Figure 9: Comparison of the dependence of the quality of the render methods on the speed of camera movement.

As can be seen from the graph, the proposed method works comparably better than the standard render in all situations. With an increase in the speed of rotation of the camera, i.e. with a large loss of time information, the quality of the resulting image tends to the quality of standard rendering, but does not become worse than it. This example is synthetic, necessary for demonstration, in most cases the angle of rotation of the camera between adjacent frames at 60 frames per second is from 1 to 5 degrees. Consequently, on average, the method is comparable in quality to MSAA.

The proposed method mathematically correctly projects dynamic objects that do not change shape. Other objects that are not static according to the algorithm are not reprojected. For

them, the color component calculated only on the current frame is taken. This improvement allows you to get rid of ghostly halos chasing moving objects.

The diagram 10 shows the speed of the various tested methods. It shows that the proposed method works much faster than MSAA, and at the same time is comparable in quality with it, which follows from the previous arguments. In speed, it is inferior to the usual render and the original TAA, but it gets a smoother image with small performance costs.

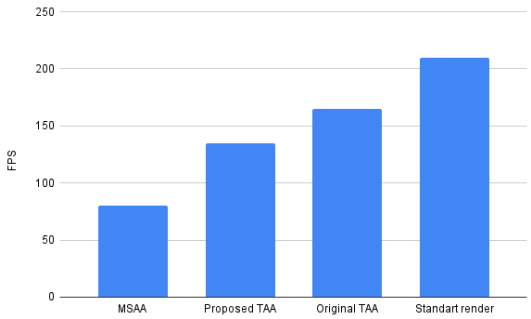


Figure 10: Diagram of the number of frames per second of each method.

The limitations of the proposed method include the problems that ordinary TAAS suffer from. The projection of reflected objects generates a ghostly trail, or creates an object where there is none. A similar problem occurs with transparent materials.

Based on experiments, the practical benefits of the proposed method are clearly shown, such a technique can be used not only for smoothing, but also for full frame generation. Images created in this way in some situations are obtained with noticeable artifacts, but almost no computational resources are spent on their rendering. Mobile devices are currently in dire need of such improvements for a variety of reasons: low computing power compared to stationary computers, strong dependence on power consumption and high heating of the device under heavy loads.

5. Conclusion

This paper presents a small overview of improvements in temporal methods, as well as a new algorithm for using temporal information, which improves the standard algorithm for dynamic objects that do not change their topology. As it was shown in the experimental section, the proposed algorithm demonstrates qualitative results comparable to hardware acceleration (MSAA) and surpasses other methods using time information, and its speed allows working in real time.

References

[1] J. Korein, N. Badler, Temporal anti-aliasing in computer generated animation, in: Proceedings of the 10th annual conference on Computer graphics and interactive techniques, 1983,

pp. 377–388.
[2] K. Akeley, Reality engine graphics, in: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, 1993, pp. 109–116.
[3] M. Pritchard, J. Brooks, R. Geldreich, Deferred lighting and shading, in: Game Developers Conference, 2004.
[4] S. Hargreaves, M. Harris, Deferred shading, in: Game Developers Conference, volume 2, 2004, p. 31.
[5] D. Scherzer, L. Yang, O. Mattausch, D. Nehab, P. V. Sander, M. Wimmer, E. Eisemann, Temporal coherence methods in real-time rendering, in: Computer Graphics Forum, volume 31, Wiley Online Library, 2012, pp. 2378–2408.
[6] J. C. Brailean, R. P. Kleihorst, S. Efstratiadis, A. K. Katsaggelos, R. L. Lagendijk, Noise reduction filters for dynamic image sequences: A review, Proceedings of the IEEE 83 (1995) 1272–1292.
[7] P. Haeberli, K. Akeley, The accumulation buffer: Hardware support for high-quality rendering, ACM SIGGRAPH computer graphics 24 (1990) 309–318.
[8] D. Scherzer, S. Jeschke, M. Wimmer, Pixel-correct shadow maps with temporal reprojection and shadow test confidence, in: Proceedings of the 18th Eurographics conference on Rendering Techniques, 2007, pp. 45–50.
[9] D. Nehab, P. V. Sander, J. Lawrence, N. Tatarchuk, J. R. Isidoro, Accelerating real-time shading with reverse reprojection caching, in: Graphics hardware, volume 41, 2007, pp. 61–62.
[10] L. Yang, D. Nehab, P. V. Sander, P. Sitthi-amorn, J. Lawrence, H. Hoppe, Amortized supersampling, ACM Transactions on Graphics (TOG) 28 (2009) 1–12.
[11] T. Lottes, TSSAA: Temporal supersamping AA, 2011. URL: http://web.archive.org/web/20120120082628/http://timothylottes.blogspot.com/2011_04_01_archive.html.
[12] B. Karis, High-quality temporal supersampling, Advances in Real-Time Rendering in Games, SIGGRAPH Courses 1 (2014) 2614028–2615455.
[13] M. Salvi, An excursion in temporal supersampling, in: Game Developers Conference, volume 3, 2016, p. 12.
[14] R. Herzog, E. Eisemann, K. Myszkowski, H.-P. Seidel, Spatio-temporal upsampling on the gpu, in: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, 2010, pp. 91–98.
[15] A. Tatu, Towards cinematic quality anti-aliasing in quantum break, in: Game Developers Conference Europe, 2016.
[16] E. Games, Unreal engine, Unreal engine 4 (2019).
[17] A. Marrs, J. Spjut, H. Gruen, R. Sathe, M. McGuire, Adaptive temporal antialiasing, in: Proceedings of the Conference on High-Performance Graphics, HPG '18, Association for Computing Machinery, New York, NY, USA, 2018. URL: https://doi.org/10.1145/3231578.3231579. doi:10.1145/3231578.3231579.
[18] L. Xiao, S. Nouri, M. Chapman, A. Fix, D. Lanman, A. Kaplanyan, Neural supersampling for real-time rendering, ACM Transactions on Graphics (TOG) 39 (2020) 142–1.
[19] K. Beets, D. Barron, Super-sampling anti-aliasing analyzed (2000).