RECONSTRUCTION OF IMAGE AFTER ANTI ALIASING

Prof. Sujogya Mishra [1], Mr. Pratyusabhanu Khuntia [2], Mr. Santosh Kumar Bisoi [3] and Mr. Chidananda Bhagabati [4]

[1] Department of CSE
Krupajal Engineering College
Bhubaneswar, Orissa, India

[2] Department of CSE
Krupajal Engineering College
Bhubaneswar, Orissa, India

[3] Department of CSE
Krupajal Engineering College
Bhubaneswar, Orissa, India

[4] Department of CSE
Krupajal Engineering College
Bhubaneswar, Orissa, India

Abstract: Anti aliasing is a process to reduce the aliasing effect. Our objective is to reduce the aliasing effect in the post processing stage. After applying anti aliasing techniques on a particular image, some aliasing effect is still present in the rendering image. Several methods and tools are available to reduce that effect. we propose an idea to reduce the aliasing effect in the post processing stage. We use clipping algorithms to locate the exact pixel position that is present on a particular region, after finding the exact position we apply flood fill scan line algorithms to verify whether the pixel position is present on a particular line or not , with definite region boundary. We apply illumination model to reduce aliasing such as we increase intensity values of the pixels which are found on the line of a particular region boundary and tries to reduce the intensity values of the pixels those are not present on the line.

Keywords: **Aliasing, post processing, rendering, splines, pixels, boundary,Bspline**

1.1. INTRODUCTION

Anti-aliasing means removing signal components that have a higher frequency than is able to be properly resolved by the recording (or sampling) device. This removal is done before re sampling at a lower resolution. When sampling is performed without removing this part of the signal, it causes undesirable artifacts such as the black-and-white noise.

In signal acquisition and audio, anti-aliasing is often done using an analog anti aliasing filter to remove the out of band components of the input signal before to sampling with analog to digital converter .In the context of Computer Graphics the topography we face named as aliasing. In general Aliasing is a signal processing term, aliasing occurs when a system being measured at an insufficient sampling rate. If a disk began to rotate at one revolution per minute, we could observe the angular velocity. If we open our eyes in every 15 sec we can still measure direction of rotation and speed. If we increase it from 15sec to 30 sec and then became difficult to determine the rotation (this is the NYQUIST FREQUENCY).

In computer graphics, anti aliasing improves the appearance of rendering polygon edges, so they are not "jagged" but are smoothed out on the screen.

The visual distortion that occurs when anti-aliasing is not used for example in case check board when render without applying anti aliasing appear the effect is clearly observed.

Anti-aliasing when used to blend the boundary pixels of a sample object; this reduced the aesthetically jagging effect of the sharp, step-like boundaries that appear in the aliased graphic at the left. Anti-aliasing is often applied in rendering text on a computer screen, to suggest smooth contours that better emulate the appearance of text produced by conventional ink-and-paper printing.

Particularly with fonts displayed on typical LCD screens, it is common to use sub pixel rendering techniques like Clear Type. Sub pixel rendering requires special color-balanced anti-aliasing filters to turn what would be severe color distortion into barely-noticeable color fringes. Equivalent results can be had by making individual sub pixels addressable as if they were full pixels, done in software or hardware. Anti-aliasing is a method of fooling the eye that a jagged edge is really smooth. Anti-aliasing is often referred in games and on graphics cards. In games especially the chance to smooth edges of the images goes a long way to creating a realistic 3D image on the screen. Remember though that Anti-aliasing does not actually smooth any edges of images it merely fools the eye. Like a lot of things they

are only designed to be good enough. If you can't tell the difference then that's fine. Let's take a look at the example given below demonstrate the effect of anti aliasing

Why do we get jagged edges on our graphics?

Jagged edges are caused by limitations in a computer screen, whether that be a CRT or TFT/LCD screen its all the same. Monitors are capable or producing nearly perfect

Straight lines either horizontally or vertically, but when it comes to diagonal lines of any angle our monitor is not capable of producing a line without some jagged edge. This is because our screen is made up of pixels in a grid formation. When we draw a diagonal line on a computer screen it has to cross several grid lines, because pixels create blocks of color diagonal lines displace these blocks slightly causing jagged edges. Informally speaking **a**nti-Aliasing is a method of fooling the eye that a jagged edge is really smooth. In games especially the chance to smooth edges of the images goes a long way to creating a realistic 3D image on the screen. Anti-aliasing does not actually smooth any edges of images it merely fools the eye. Another realistic example is watching a movie with a stage coach, the wagon wheels appear to move backwards once a certain speed is reached.. Aliasing occurs when sample of signal (anything which repeats a cycle over time) too slowly (at a frequency comparable to or smaller than the signal being measured), and obtain an incorrect frequency, amplitude as a result. For an example, let's say a ball that has three positions: 0 (middle), +1 (up), and -1 (down). Let's say that ball starts at 0 at time=0, goes to +1, then to 0, then to -1, then back to 0, and each of these 4 movements takes 1 second. Since it takes 4 seconds to complete this cycle, the frequency of this motion would be .25 Hz. Now let's say you want to measure this frequency, but you only look at the ball once every 5 seconds. At t=0, you see it at 0, at t=5 you see it at 1, at t=10 you see it at 0, at t=15 you see it at -1, and finally at t=20 you see it at zero after what appears to be a complete cycle. You will then say that the frequency appears to be (1 cycle)/ (20 seconds) = .05 Hz. But this is wrong! Because we weren't aware of the above said, so we missed out a bunch of movement, resulting in you 'measuring' a totally different frequency. In general anti aliasing is a term generally used in the field of digital signal processing. When an analog signal is digitized, any component of the signal that is above one-half the sampling or digitizing frequency will be 'aliased.' This frequency limit is known as the Nyquist frequency. When a digitized signal is analyzed, often by Fourier analysis, the power contained in the frequencies above the Nyquist frequency are added to lower frequency components. In fact, they are indistinguishable from those lower frequency components. The appearance of aliasing effects is due to the fact that lines, polygon edges, color boundaries, are continuous, where as raster device is discrete. On the raster display device, it must be sampled at discrete location. If two signals are sampled at the same rate, then the reconstructed signals are identical. In order to prevent aliasing, a signal must be sampled at a rate at least twice the highest frequency in the signal. This rate is called the Nyquist limit or frequency.

1.2 .Techniques for reducing aliasing

1.2.1. Signal processing approach to anti-aliasing

In this approach, the ideal image is regarded as a *signal*. The image displayed on the screen is taken as samples, at each (*x,y*) pixel position, of a filtered version of the signal. Ideally, one would understand how the human brain would process the original signal, and provide an on-screen image that will yield the most similar response by the brain.

The most widely accepted analytic tool for such problems is the Fourier transform; this decomposes a signal into basic functions of different frequencies, known as frequency components, and gives us the amplitude of each frequency component in the signal. The waves are of the form:

$$\cos(2j\pi x)\cos(2k\pi y)$$

Where *j* and *k* are arbitrary non-negative integers. There are also frequency components involving the sine functions in one or both dimensions, but for the purpose of this discussion, the cosine will suffice. The numbers *j* and *k* together are the *frequency* of the component: *j* is the frequency in the *x* direction, and *k* is the frequency in the *d*irection

1.3. Mipmapping

There is also an approach specialized for texture mapping called mipmapping which works by creating lower resolution, prefiltered versions of the texture map. When rendering the image, the appropriate-resolution mipmap is chosen and hence the texture pixels (texels) are already filtered when they arrive on the screen. Mipmapping is generally combined with various forms of texture filtering in order to improve the final result.

2.1. MORPHOLOGICAL ANTIALIASING

MLAA is designed to reduce aliasing artifacts in displayed images without casting any additional rays. It consists of three main steps:

1. Find discontinuities between pixels in a given image.
2. Identify predefined patterns.
3. Blend colors in the neighborhood of these patterns.

For the shake of simplicity, we first describe the MLAA technique for (binary) black-and-white images, for which these three steps are trivial, and generalize it for color images later on.

DISCONTINUITIES IN COLOR IMAGE

To find discontinuities in an image, one can rely either on difference in color between neighboring pixel or on some additional information, geometric or discrete (difference materials). The biggest advantage of the first approach (using color) is that this approach does not process pixel with similar color, thus avoiding extra work that would likely not change the color of the pixels in any meaningful way. On the downside, this approach also triggers superfluous processing of the pixels that get significant difference in the color only through texture mapping. MLAA as such can be used with any method that tells whether two pixels are different. For the remainder of this paper, we will use a threshold-based color differences without any additional information, because this approach is the most simple and universal one. Since MLAA does not cast any additional rays, it mitigates performance impediments caused by choosing low threshold value. We will also use an additional rule to decide when to actually blend pixels, as will be shown in the next section. In all results, presented in this paper, we use only the 4 most significant bits as a similarity measure for each 8-bit red, green, and blue channel (this is comparable with DXT1 texture compression, in which all texels that have the same 5 leading bits are considered equal). If there is a difference in these bits in either channel, two colors are assumed to be different. Using either 3,5, or 6 bits does not change results in any significant way, through for some images 3 bits are not sufficient to find all aliased pixels that are visible by a human eye.

2.2. PATTERN SEARCH

We will start deriving rules for searching patterns in color images by noticing that the color blending expression(2) does not depend on numerical values for black (0) and white(1) colors. It could be used for RGBA colors as well, separately for each channel. For black- and-white(B&W) images, color computed according to this equation are changing smoothly when different Z,U, or L shapes are adjacent to each other. It is a consequence of using middle points of the secondary edges as it defines continuous piecewise-linear approximation of silhouette edges. For color images, we will relax this restriction, but will still require continuous silhouette approximation.

MLAA LIMITATION

Arguably the biggest MLAA limitation is its inability to handle the pixel-size features. MLAA takes the aliased image shown on the top and produces the one in the middle. Despite being very narrow, quads, which are closer to the camera, span multiple pixels, For these quads, MLAA is working just fine, removing higher frequencies from the image.

2.3. Limitation of MLAA and our concept

Initially we find the pixel position by using clipping algorithm presents on a particular region[23] then we use flood fill [22] algorithms to find number of pixel on that region , then apply scan line algorithm [23] on it to find pixels position which are on the scan line ,we use edge coherence property [23] for the entire region . then to get the outline of a figure we use Bspline [22], once we get the outline we increase pixel values those are present on the scan lines and reduce the resolution of the background of those which are not present on the scan line, and expecting that these techniques will further reduced the aliasing effect to certain extent in the post processing stage. In this regard we present all the related algorithm

Algorithm-1

FINDING THE POSITION OF THE PIXELS WITHIN A WINDOW

Step-1 considering four pixel position as four corner of a window
Step-2 first position considered to be left lower corner as (umin, vmin), with respect to origin. Last position considered to be right upper corner as (umax, vmax), with respect to origin
Step-3 using these two positions we can find the rest two positions.
Step-4 all the four position looks like as an window.
Step-5 horizontal distance will be umax-umin.
Vertical distance will be vmax-vmin.
Step-6 continue with step1 to5 till the distance between all the corresponding pixels are found out in the visible zone.

We give the idea that both flood fill as well as piecewise interpolating polynomial for shader discontinuity detection. Flood fill scan line algorithm reduce the stack size so the pixel position is not repeated and piece wise interpolating polynomial for B Spline helps in the formation of regular shape at the aliasing position after post processing.

Algorithm-2: Procedure floodFill4(x, y, fillcolor, oldcolor:integer);

```
Begin
    If getPixel (x,y)=oldColor then
     Begin
        setPixel (x,y,fillColor);
        floodFill4(x+1,y,fillColor,oldColor);
          floodFill4(x-1,y,fillColor,oldColor);
        floodFill4(x,y+1,fillColor,oldColor);
        floodFill4(x,y-1,fillColor,oldColor);
     end
  end;
{floodFill}
```

B Spline curve passes through the nodal points as it has no local control over the curve so drawing the outline of the image using B Spline curve, then little adjustment at any nodal points does not affect the shape of the outline and produce better visualization and reduce the aliasing effect substantially.

Evaluate Bspline

```
Function y=BsplineS(k,x,n)
h=1/n;
m=length(x); %the points are k*h
for j=1:m
  if(x(j)<=(k-2)*h,  y(j)=0;
  else if ((k-2)*h < x(j) & x(j) <=(k-1)*h)
     y(j)=0.25/h^3*(x(j)-(k-2)*h)^3;
else if((k-1)*h < x(j) & x(j)<= k*h)
     y(j)=0.25*(1+(3/h)*(x(j)-(k-1)*h)+(3/h^2)*(x(j)-(k-
1)*h)^2…
                  -(3/h^3)*(x(j)-(k-1)*h)^3);
else if(k*h < x(j) & x(j) <= (k+1)*h)
       y(j)=0.25*(1+(3/h)*((k+1)*h-
x(j))+(3/h^2)*((k+1)*h-x(j))^2…
                -(3/h^3)*((k+1)*h-x(j))^3);
else if ((k+1)*h < x9j) & x(j) <= (k+2)*h)
    y(j)=0.25/h^3*((k+2)*h-x(j))^3;
else,         y(j)=0;            end
end
```

Our main intention is to find the nodal points along the axes of images. The point present along the x-axis with respect to the image called as nuts with respect to the nuts, we construct the B Spline. In accordance to the above said then looking to the pattern in which the B Spline covers the image we can easily judge the points which are responsible for anti aliasing. We can increase the intensities of these points to get anti aliasing effects considerably in the post processing stage.

APPLING ILLUMINATION MODEL

Once we have the points present on a line then we collect these points in low resolution and display it in a high resolution. Once these points are present on the curve, we study its nature. If the nature of the curve is uniform then the application is same as that of the straight line. If the nature of the curve is non uniform, then we go for piece wise continuity. To study and apply illumination model for non uniform curve on which the pixel position are present we take help of Bspline curve.

COMPUTING BLENDING FACTORS

To this point we have computed the discontinuity segments, their lengths and the relative position of any pixel along these lines. Therefore, for each pixel bordering a discontinuity segment, we observe the pattern to identify which of these pixels are of L shape. Each pixel can belong up to four L shapes and will be blended accordingly. For an identified L shape, the area of the trapeze A can be pre computed and depends from segment length L in pixels, and from the relative position p of the pixel in the segment.

- Trapeze area can be

0	1	1	1	1	1	0	1	1	1

0	1	1	1	1	1	0	1	1	1

- Approach to compute the length of the discontinuity lines. Example here
- Shows the computation of the relative distance to the left horizontal end line. In the first step, pixel A checks its three closest neighbors, each one making him “jump” to the left. Pixel b checks its closest left neighbor then its left pixel which does not belong to a discontinuity line (distance is zero then). So the computation will add 0 for the two last times. In the second step, pixel b has a distance lesser than a 4, so it is discarded as all pixels marked in red. Pixel A will check the fourth- its distance- left neighbor, add one, check the left neighbor and stop here.)

Conclusion:

Our intention is to reduce the aliasing effect after post processing stage. i.e. Some aliasing effect is still left after post processing stage and we are giving some techniques in this paper and expect that it will work to reduce the aliasing effect. In this context we refer several papers mentioned below in the reference section, we are also referring several text in this regards. The concept is that we change the intensity values of the back ground and use different curve drawing algorithms (Bezier, B-Spline) to locate the presence of the pixel positions in the outline of the rendering image. We conclude with this note that we are not able to simulate the concept, it is just a proposal and expect that it will work.

References:

[1] AMANATIDES J. 1984. Ray Tracing with Cones. In *Computer Graphics*, vol. 18(3), 129-135.

[2] ATI CrossFire™ Technology White Paper, http://ati.amd.com/technology/crossfire/CrossFire WhitePaperweb2.pdf

[3] BALA K., WALTER B., GREENBERG D. 2003. Combining Edges and Points for Interactive High-Quality Rendering. *ACM Transactions on Graphics* (Proceedings of ACM SIGGRAPH 2003), 22(3), 631-640.

[4] CHAN E. and DURAND F. 2003. Rendering Fake Soft Shadows With Smoothies. In *Proceedings of the 14th Eurographics workshop onRendering*, 208-218.

[5] CHAN E. and DURAND F. 2005. Fast Prefiltered Lines. In Matt Pharr,ed., *GPU Gems 2*. ISBN 0321335597. Addison-Wesley.

[6] DIPPE M. and WOLD E.H. 1985. Antialiasing Through Stochastic Sampling. *ACM Transactions on Graphics* (Proceedings of ACM SIGGRAPH 1985), 19(3), 69-78.

[7] EARLS I. 1987. Renaissance Art: A Topical Dictionary. ISBN0313246580. Greenwood Press. 263-263.

[8] GENETTI J., GORDON D., and WILLIAM G. 1998.

[9] Adaptive Supersampling in Object Space using Pyramidal Rays, *Computer Graphics Forum*, vol. 17, 29-54.

[10] HECKBERT P. and HANRAHAN P. 1984. Beam Tracing Polygonal Objects, In *Computer Graphics*, vol. 18(3), 119-127.

[11] HOFER H., CARROLL J., NEITZ J., NEITZ M., and WILLIAMS D.R.2005. Organization of the *The Journal of Neuroscience*, 25(42), 9669-79.

[12] SEN P. 2004. Silhouette Maps For Improved Texture Magnification. *Graphics Hardware* (Proceedings Of ACM SIGGRAPH /EUROGRAPHICS 2004), 65-73.

[13] SEILER L., CARMEAN D., SPRANGLE E., FORSYTH T., ABRASH M.,DUBEY P., JUNKINS S., LAKE A., SUGERMAN J., CAVIN R.,ESPASA R., GROCHOWSKI E., JUAN T., and HANRAHAN P. 2008.

[14] MARTIN W., REINHARD E., SHIRLEY P., PARKER P., and THOMPSON W. 2002.

[15] Larrabee: A Many-Core x86 Architecture For Visual Computing.In *ACM Transactions on Graphics* (Proceedings of ACM SIGGRAPH 2008), 27(3), 1-15.

[16] KELLER A. 1998. *Quasi-Monte Carlo Methods for Realistic Image Synthesis*. PhD thesis, University of Kaiserslautern.

[17] SEN P. and CAMMARANO M. 2003. Shadow Silhouette Maps. In *ACM Transactions on Graphics* (Proceedings of ACM SIGGRAPH 2003), 22(3), 521-526.

[18] SOILLE P. and RIVEST J.-F. 1993. Principles And Applications Of Morphological Image Analysis. ISBN 0646128450.

[19] THOMAS D., NETRAVALI A.N., and FOX D.S. 1989. Anti-aliased Ray Tracing with Covers, *Computer Graphics Forum*, vol. 8(4), 325-336.

[20] WALD I., MARK W., GÜNTHER J., BOULOS S., IZE T., HUNT W., PARKER S.G., and SHIRLEY P. 2007. State of the Art in Ray Tracing Animated Scenes. *Eurographics 2007 State of the Art Reports*, 89-116.

[21] WHITTED T. 1980. An Improved Illumination Model for Shaded Display, *Commun. ACM*, vol. 23(6), 343-349.

[22] ComputerGraphics,James D.Foley,Andries van Dam,Steven K.Feiner,John F.Hughes,pp-(156-166),(1003-1004),(515)

[23] Computer Graphics,C-version,,Donald Hearn, M.Pauline Baker,pp-46-247.

[24] Lynch, P.J. & Horton, S. (1999). *Web style guide,*Roberts, J. & Gross P. (1999).

Prof. Sujogya Mishra is an eminent researcher in the field of Computer Graphics, Data Base Management Systems, Algorithm Design and Analysis. At present he is working in the Department of Computer Science and Engineering, Krupajal Engineering College, Bhubaneswar. He has completed M.Tech in Computer Science and Engineering. Now he is pursuing Ph.D in Utkal University, Vani Vihar, Bhubaneswar, Odisha.

Mr. Pratyusabhanu Khuntia is a researcher in the field of Algorithm Analysis and Design, Computer Graphics, Data Base Management Systems. At present he is working in the Department of Computer Science and Engineering, Krupajal Engineering College, Bhubaneswar. He has completed M.Tech in Computer Science and Engineering.

Mr. Santosh Kumar Bisoi is a research fellow in the field of Computer Graphics, Algorithm Analysis and Design, Data Base Management Systems. At present he is working in the Department of Computer Science and Engineering, Krupajal Engineering College, Bhubaneswar. He has completed M.C.A from BPUT, Odisha. Now he is pursuing M.Tech in Computer Science and Engineering from BPUT, Odisha.

Mr. Chidananda Bhagabati is a research fellow in the field of Algorithm Analysis and Design, Computer Graphics, Data Base Management Systems. At present he is working in the Department of Computer Science and Engineering, Krupajal Engineering College, Bhubaneswar. He has completed M.C.A from BPUT, Odisha. Now he is pursuing M.Tech in Computer Science and Engineering from BPUT, Odisha.