CORRELATIVE ANALYSIS OF DIFFERENT IMAGE SCALING ALGORITHMS IN GRAPHICS

Zeeshan Asghar, Rizwan Naeem, Nabila Karamat Jarral

Abstract

This study paper demonstrates to find out what is basically scaling in terms of image? What are the basic functions used in image scaling, also we take a look on different scaling algorithms and try to find out which algorithms is best in which context and what are the flaws and limitations of these algorithms. We also try to give out complimentary solution that what should be needed in order to make the efficiency of the scaling algorithms more strong.

Keywords: Computer Graphics, Image Scaling, Scaling Algorithms,

1 Introduction

In the world of graphics, image scaling is the procedure of resizing a computerized picture. Scaling is a non-trivial process that includes an exchange off between productivity, smoothness and sharpness. With bitmap representation, as the extent of a picture is decreased or extended, the pixels that frame the picture turn out to be progressively obvious, making the picture seem "soft" if pixels are averaged, or jagged if not. With vector graphics the trade-off may be in processing power for re-rendering the image, which may be noticeable as slow re-rendering with still graphics, or slower frame rate and frame skipping in computer animation.

Apart from fitting a smaller display area, image size is most commonly decreased (or subsampled or downsampled) in order to produce thumbnails. Enlarging an image (upsampling or interpolating) is generally common for making smaller imagery fit a bigger screen. In “zooming” a bitmap image, it is not possible to discover any more information in the image than already exists, and image quality inevitably suffers. However, there are several methods of increasing the number of pixels that an image contains, which evens out the appearance of the original pixels.

2 Summary

In this section now we will take a look on the scaling algorithms from past to present in order to understand that what are previous needs, how they was achieved and in this era what sort of scaling algorithms are working in the universe of computer graphics.

Study[1] shows that Nearest-Neighbor Interpolation is one of the most old and widely used algorithm in the scaling of image. Basically functionality of this algorithm is to replace each pixel with number of pixels of the similar color. This means that id the image is zoomed or magnified as in most cases if there is no scaling algorithm then the image pixels should be black out but if this algorithm will implemented then the pixels which are empty to the larger resulting image will be filled by nearest neighbor color. Suppose the neighbor has the color of white so neighbor 2 should be filled by white color when image becomes large.

From the fig1.2 as the image is enlarged, the pixels box which was empty due to the change of size was filled by nearest neighbor color. This algorithm however will not work much as when image becomes too much large it will causes raster image and hence image cannot be identified properly, and sometime when colors are too much in single pixel it will causes the mixture of different colors, which is wrong.

In this stream a new Bilinear interpolation Algorithm[2] which shows Bilinear interpolation sounds like 'bilerp' as a nickname is process of filtering the surrounding texels, to smooth out any rasters or blurry zigzags occurring between pixels, and giving the screen a smoother look. Basically we can say that it is the enhanced power of NNI Algorithm.
As we can see from fig 1.4 that is done by Bilinear interpolation Algorithm and hence it is quite better as compared to the Nearest-Neighbor Interpolation in some extend and in some cases. As the matter of fact is problem is still same that image is still blurry and raster too. On the other hand we also came to the point that [3] Bicubic Interpolation Algorithm are basically mixture of 1st two algorithms which are works on neighbor side but as also do a surroundings textures to make the image more clear and more easy visible

![Fig 1.5 Bicubic](image)

![Fig 1.3 Original Image](image)

Figure 1.5 shows that image is looking like more clear and visible. As the matter of fact this algorithm is mixing up two other algorithms and hence it will takes time and its time complexity is more as compared the the 1st two algorithms but fact is the result of the image quality will be more nice as compared to the other 2 algorithms.

In this time a new algorithm [4]Fourier-based interpolation was introduced. We can say that it was introduced to wipe out demerits of the Bicubic Interpolation Algorithm. In these algorithm basically we try to find out the frequency domain with zero components, here zero components will be considered as those pixels are not filled up just because of image enlargement. Hence this algorithm also try to find out the value and if if crosses the section area it will causes the filling of color which will be widely used as in overall whole image. As the matter of fact this is basically the 4th step in the world of scaling algorithm

![Fig 1.6 Fourier Based](image)

![Fig 1.3 Original Image](image)

Fig 1.6 shows how much it is looking more clear and visible and every time to calculate a frequency domain frequency and then it analysis becomes hell of work for the CPU, thus this algorithm fails when we talk about the process management as it causes a lot of process for simple task.

One of the most wide used are [5]Edge-directed interpolation algorithms as the matter of fact is totally focus on the edges of the image. As image is enlarged the 1st thing will be done on edge to make it smooth and able to understand. Fact is this algorithm is best of edge detection section or forensic IT section where we have to concern with only shape but here in the universe of computer graphics the whole image should be be clear in such a way that it will be easy to understand that what a image represents. Edge-directed interpolation algorithms aim to preserve edges in the image after scaling, unlike other algorithms which can produce staircase artifacts around diagonal lines or curves.in it there is also concept of [6]Vectorization An entirely different approach is vector extraction or vectorization. Vectorization first creates a resolution independent vector representation of the graphic to be scaled. Then the resolution-independent version is rendered as a raster image at the desired resolution. This technique is used by Adobe Illustrator Live Trace, Inkscape, and several recent papers. Scalable Vector Graphics are well suited to simple geometric images, while photographs do not fare well with vectorization due to their complexity.

**Analysis & Findings**

After some experimentation, we came up with these rules of thumb:

- **When making an image smaller, use bicubic**, which has a natural sharpening effect. You want to emphasize the data that remains in the new, smaller image after discarding all that extra detail from the original image.

- **When making an image larger, use bilinear**, which has a natural smoothing effect. You want to blend over the interpolated fake detail in the new, larger image that never existed in the original image.
Of course, there are plenty of conditions that might make you want to choose one method over the other, but I think these are reasonable guidelines to start with. Compare this shot of Mario vs. Wario using pixel resizing and the same shot using 2xSAI resizing. It's a dramatic difference, especially since traditional bilinear and bicubic upsizing methods degenerate into a giant blur on pixel art.

Supposedly, one of the best image resizing algorithms on the market is Genuine Fractals. The web site boasts that you can use its fractal-based resizing algorithm to “enlarge your images over 1000% with no loss in image quality”. It’s probably pure marketing hyperbole, but I was still intrigued. Bilinear and Bicubic are decent, but there has to be room for improvement in there somewhere. I downloaded a trial version of the tool (which requires Photoshop Elements, or Photoshop CS) and gave it a shot.

From start to end on this document we have seen that what are the different algorithms which are used for scaling algorithms, their advantages and their disadvantages here we came to the point till to end that Vectorization method is the best method. Although it is takes time but gives maximum output. Hence vectorization algorithms are always be best when we talk about excellent result but in terms of time management nearest neighbor will be good as compared to other, it is because of course it just has to filled empty pixel so just checked the neighbor’s color and fill it. Remember never any algorithm will be best in all the case like time and accuracy but some are good in time some are good in time.

Among the more popular algorithms for enlarging a bitmapped image are pixel replication and bilinear interpolation; these algorithms are quick and simple to implement. The pixel replication algorithm simply copies the nearest pixel (nearest neighbour sampling). As a result, the algorithm is prone to blockiness, which is especially visible along the edges. Bilinear interpolation is at the other extreme, it takes the (linearly) weighted average of the four nearest pixels around the destination pixel. This causes gradients to be fluently interpolated, and edges to be blurred. Several other algorithms use different weightings and/or more pixels to derive a destination pixel from a surrounding set of source pixels.

The reasoning behind these algorithms is that the value of a destination pixel must be approximated from the values of the surrounding source pixels. But you can do better than just to take the distances to the surrounding pixels and a set of fixed weight factors stored in a matrix. Directional interpolation is a method that tries to interpolate along edges in a picture, not across them.

My interest in directional interpolation was spurred by two papers. In [6] the authors use polynomial interpolation with gradient-dependent weights. To calculate the gradients, the paper applies a pair of Sobel masks on every source pixel taking part in the interpolation. The algorithm in [7] splits a source pixel into four target pixels in one pass, considering eight additional surrounding pixels (nine source pixels in total). By applying a matrix with pre-computed weights for every relation between the nine pixels, the value of each of the four target pixels is calculated.

The algorithm in [6] has fairly simple operations on every source pixel that it considers for each target pixel, but it considers a big set of source pixels. This is a major performance bottleneck. The other paper ([7]) uses a few surrounding pixels, and calculates four destination pixels in one pass, but the amount of arithmetic required for the calculation makes it unsuitable for real-time scaling. Neither algorithm is easily adapted to colour images; a colour image is regarded as a 3-channel image where each channel must be resized individually.

The idea, then, is to combine concepts from the two papers, but at a lower cost: simple operations on few source pixels per target pixel.

A directionally interpolation algorithm requires that you find the edges first, or rather, that you determine the direction of the gradient at every pixel. While the paper [6] uses only the magnitude of the gradient, [7] also considers its direction, to its advantage. Simple edge detectors that are suitable to find the gradient (magnitude and direction), are pairs of Roberts masks or Sobel masks. In the figure below, the Roberts masks are put in 3×3 matrices, but, as you can see, calculating the value of one mask involves just one subtraction. Either Roberts mask detects an edge in a diagonal direction. From the two values, you can calculate both the direction of the gradient and the ”steepness” of the gradient. The other figure below shows a pair of Sobel masks; one Sobel mask detects an edge in the horizontal or vertical direction and, like the Roberts masks, both Sobel masks are combined for the complete gradient information. Given the direction of the gradient, I could then choose an appropriate directional interpolator. These interpolators are just linear, not bilinear.

Bicubic wouldn’t normally be my choice here, but I chose it because it’s technically the most advanced method, and it produces the results closest to the effect that the fractal resizing delivers. Still, the fractal algorithm comes out way ahead: you can’t see any pixel resize artifacts in the enlarged image, and the edges are sharp and well defined. It does start to bear an unfortunate resemblance to a watercolor drawing filter, but arbitrarily resizing images to 5 times their original size will always involve tradeoffs of some kind.
Conclusion

Reducing images is a completely safe and rational operation. You're simply reducing precision and resolution by discarding information. Make the image as small as you want, and you have complete fidelity - within the bounds of the number of pixels you've allowed. You'll get good results no matter which algorithm you pick. (Well, unless you pick the naive Pixel Resize or Nearest Neighbor algorithms.)

Enlarging images is risky. Beyond a certain point, enlarging images is a fool's errand; you can't magically synthesize an infinite number of new pixels out of thin air. And interpolated pixels are never as good as real pixels. That's why it's more than a little artificial to upsize the 512x512 Lena image by 500%. It'd be smarter to find a higher resolution scan or picture of whatever you need* than it would be to upsize it in software.

But when you can't avoid enlarging an image, that's when it pays to know the tradeoffs between bicubic, bilinear, and more advanced resizing algorithms. At least arm yourself with enough knowledge to pick the best of the bad options you have.
3 References


[8] An algorithm to split a source pixel in 4 target pixels by looking at eight surrounding pixels. The algorithm is "trained" on a set of specific input images where the desired output for that image is known.

4 Bibliography