AUTOMATIC IMAGE RESAMPLING FILTER GENERATION

Costin-Anton Boiangiu [1*]
Ionut-Adrian Nitu [2]

ABSTRACT

The digital world, that we're in constant contact with, is filled with images of all sorts and shapes. However, the resolution of some of those is just not good enough in several scenarios: feature recognition, OCR, document processing, machine vision, and so on, thus making the resampling of the images a very important step towards a correct processing,
For the purpose of this work, several existing filters have been analyzed using a comprehensive set of test images. The main goal is to create new filters by using different approaches: genetic algorithms - by generating populations of filters and keeping the best individuals based on a mean error criteria, brute-force solution searching– by selecting the most efficient filters.

KEYWORDS: *Image resampling, filtered resampling, downsampling, upsampling, Box, Hermite, Triangle, Cubic, Lanczos, Mitchell, Bell, B-Spline, genetic algorithms, brute force.*

INTRODUCTION

With the current integration of multimedia into almost every aspect of daily lives, consumers (people) are viewing visual data like images, video, on a wide variety of products, ranging from computer screens to a large selection of handheld devices. Image resizing [1, 2] is the process where an image is converted from one resolution/dimension to another resolution/dimension with as low as possible detail loss, thus a certain mechanism of interpolation is usually required.

There are two main categories of image interpolation algorithms: adaptive and non-adaptive. In the latter category the computational logic is fixed regardless of the input image features, whilst in adaptive algorithms computational logic is dependent upon the intrinsic image features and content of the input image [2]. Interpolating from lower to higher resolution is termed as upsampling and from higher to lower resolution is termed as downsampling.

The paper at hand presents the students' efforts, carried under the supervision of the first author for the project of Document Image Analysis, aimed at finding, by any available means, the most suitable filter functions for downsampling/upsampling purposes, filters that are different from the "well-established" (classical) ones.

[1]* corresponding author, Professor PhD Eng., "Politehnica" University of Bucharest, 060042 Bucharest, Romania, costin.boiangiu@cs.pub.ro
[2] Engineer, "Politehnica" University of Bucharest, 060042 Bucharest, Romania, ionut.nitu@cti.pub.ro

RELATED WORK

Image resizing typically involves fractional resampling and can lead to prohibitively large implementations resulting in compromises in color range and resolution.

Numerous digital image scaling techniques have been developed: nearest neighbor, linear pixel interpolation, kernel-based B-Spline, Triangle, Hermite, Lanczos, Mitchell, and so on. For example, Nearest Neighbor Interpolation [4] it is one of the fastest and simplest forms of interpolation technique. During enlarging (up-scaling), the empty spaces will be replaced with the nearest neighboring pixel. Shrinking, on the other hand, involves reduction of pixels.

A notable work includes the solution proposed by Kopf et al. [5]. It is first calculated at a low resolution, which is then upsampled using joint bilateral filtering. In Avidan and Shamir [6], a content-aware image resizing algorithm was proposed. Rather than resizing an image by scaling, this method carves out or inserts content using the image seams, the pixels chains regarded as being of little importance.

In Hegde, Tuzel and Porikli [7] is presented an algorithm that comprises of two main stages of processing in two layers: edge and detail. Firstly, for the edge layer, it is used a nonparametric approach by constructing a dictionary of patches from a given image, and synthesize edge regions in a higher-resolution version of the image. For the detail layer, a global parametric texture enhancement approach serves to synthesize detail regions across the image.

Despite an extensive research in this area, image scaling remains a computationally expensive operation. Its cost is dominated by convolution, which is necessary to control undesirable reconstruction and aliasing artifacts. Convolution may prove to be prohibitively expensive, especially when large (high-quality) filter kernels or large scale factors are applied. [8]

EXPERIMENTAL METHODOLOGY

Peak Signal to Noise Ratio (PSNR), is the ratio between the corrupting noise that affects the fidelity of image representation and the maximum possible power of a signal [9]. PSNR is usually expressed in terms of the logarithmic decibel scale due to a very wide dynamic range of signals. In this case, original data represents the signal, and the noise is the error introduced by the succession of downsampling-upsampling operations. Highest value of PSNR indicates the highest quality of image reconstruction, thus better resampling quality.

Mean Square Error (MSE) quantifies the difference between values generated by an estimate and the true quality being certificated. Lowest value of PSNR indicates the highest quality of image.

Present testing is realized using downsampling followed by upsampling, measuring the differences using PSNR.

GENERATION OF UP/DOWNSAMPLING FILTERS

Evaluation of "well-established" filters

The first step was to evaluate some of the classical filters, such as the Box, Hermite, Triangle, Bell, Spline, Lanczos3 and Mitchell filter. The point of this phase was to determine whether there is a best filter for upsampling and a different best filter for downsampling.

Table 1. Evaluation of "well-established" filters in upsampling/downsampling operations. Legend: the better the filter performance, the darker the markup color

Down/Up	Box	Hermite	Triangle	Bell	Spline	Lanczos3	Mitchell
Box	287.037	260.667	265.642	282.34	298.357	245.71	261.839
Hermite	291.979	267.795	272.776	289.727	305.037	248.155	268.671
Triangle	296.641	274.967	279.559	295.749	310.37	254.127	275.287
Bell	308.552	292.317	296.452	310.785	323.774	271.008	291.89
Spline	320.687	307.241	310.911	323.809	335.553	286.723	306.38
Lanczos3	296.279	254.116	256.4	271.358	286.918	236.285	251.832
Mitchell	296.138	272.089	276.207	291.779	306.243	251.102	271.794

A genetic algorithm-based approach

Following the desire of obtaining a better filter, the first proposed approach is geared towards running genetic algorithms to generate populations of new filters and test their behavior in a manner similar to natural selection.

Rather than representing the filters as mathematical functions, which does not permit combining two filters in an easy fashion, filters can be represented in a discrete manner, by storing the value of the filter in certain points and interpolating the values in between.

If the distance between the coordinates of the points is constant, then the range in which one needs to search to obtain the value in the desired point can be determined in constant time.

An individual fitness function is used in the method of evaluation, and so it promotes individuals which are better and should be kept in the further generations. For the proposed solution, the fitness is represented by the average MSE for that filter when running downsampling followed by upsampling on a certain set of images at different scales.

Two individuals are selected from the current population and combined to obtain new ones. The strategies used are:

- Strategy 1: average each point.
- Strategy 2: average with a 3x3 window. This helps smoothing the curves.
- Strategy 3: select each point either from one individual or the other. This is run twice (2 new individuals are created).

In order to obtain a more diverse population, mutations occur for a random number of the new individuals. A mutation simply implies that some of the points of the discrete filter are altered.

When a new era starts and a generation is created, the better half of the individuals (sorted by MSE) randomly chose a partner to recombine with. Each pair creates four new individuals. The entire population is sorted by its fitness and only the best individuals survive to the next generation by either adding or subtracting a random number.

Table 2. MSE results on different scales and filters. Legend: The best filter performances are written with bold font.

Filter Type	Scale 1.5	Scale 2.0	Scale 2.5	Scale 3.0	Scale 3.5	Scale 4.0	Scale 4.5	Scale 5.0
Box	4023.126	232.315	1736.314	242.555	1108.519	360.432	917.524	460.8
Hermite	55.26	95.035	145.789	199.712	254.844	306.536	361.743	413.003
Triangle	158.377	106.276	189.241	227.301	289.715	343.308	401.47	450.88
Bell	91.759	155.344	225.446	296.803	365.813	429.552	490.957	546.759
B-Spline	118.437	196.452	278.102	358.141	432.735	500.304	563.161	619.307
Lanczos	**23.595**	**50.353**	**82.121**	**120.844**	**164.746**	**210.007**	**258.763**	**307.383**
Mitchell	51.137	94.925	146.335	202.671	260.673	316.784	374.123	427.731
Proposed	451.941	363.693	461.467	701.974	608.837	877.197	863.227	718.165

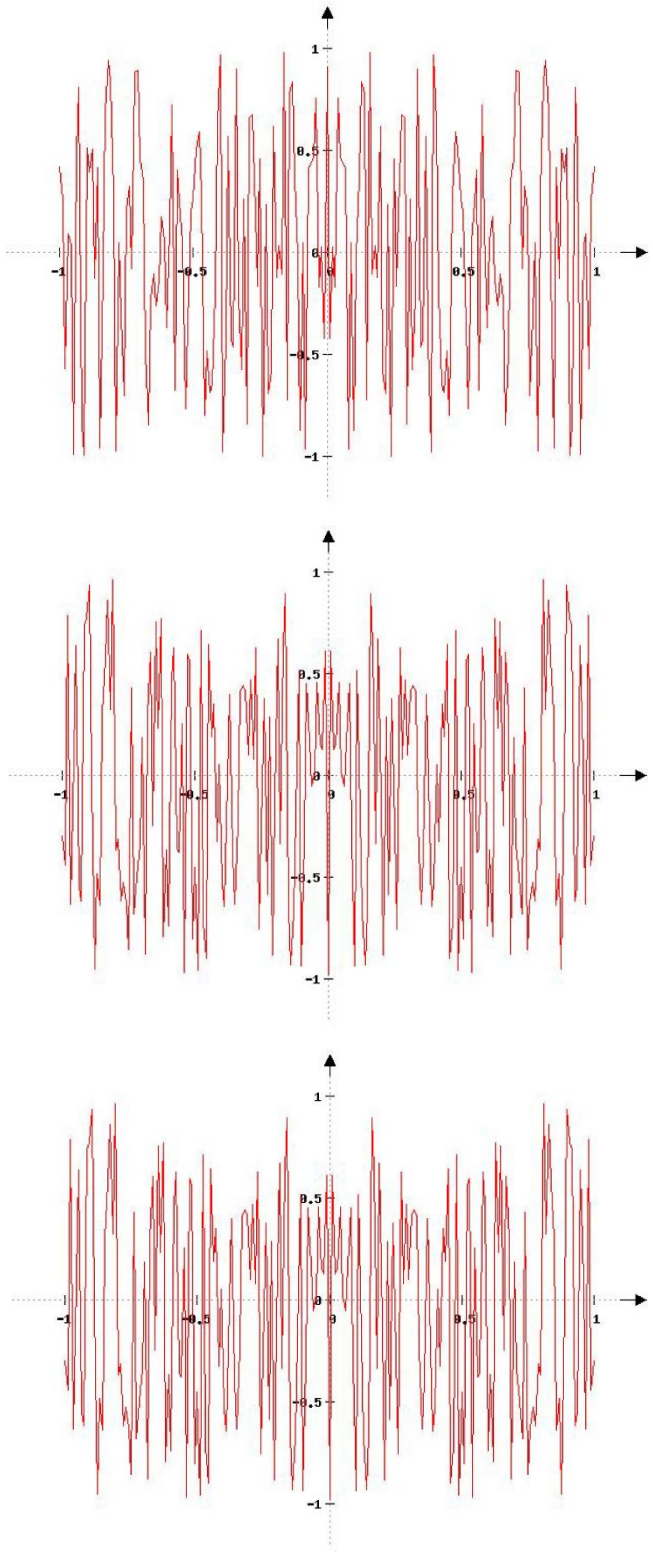

Figure 1. The evolution of filters – 10, 50, 100 individuals

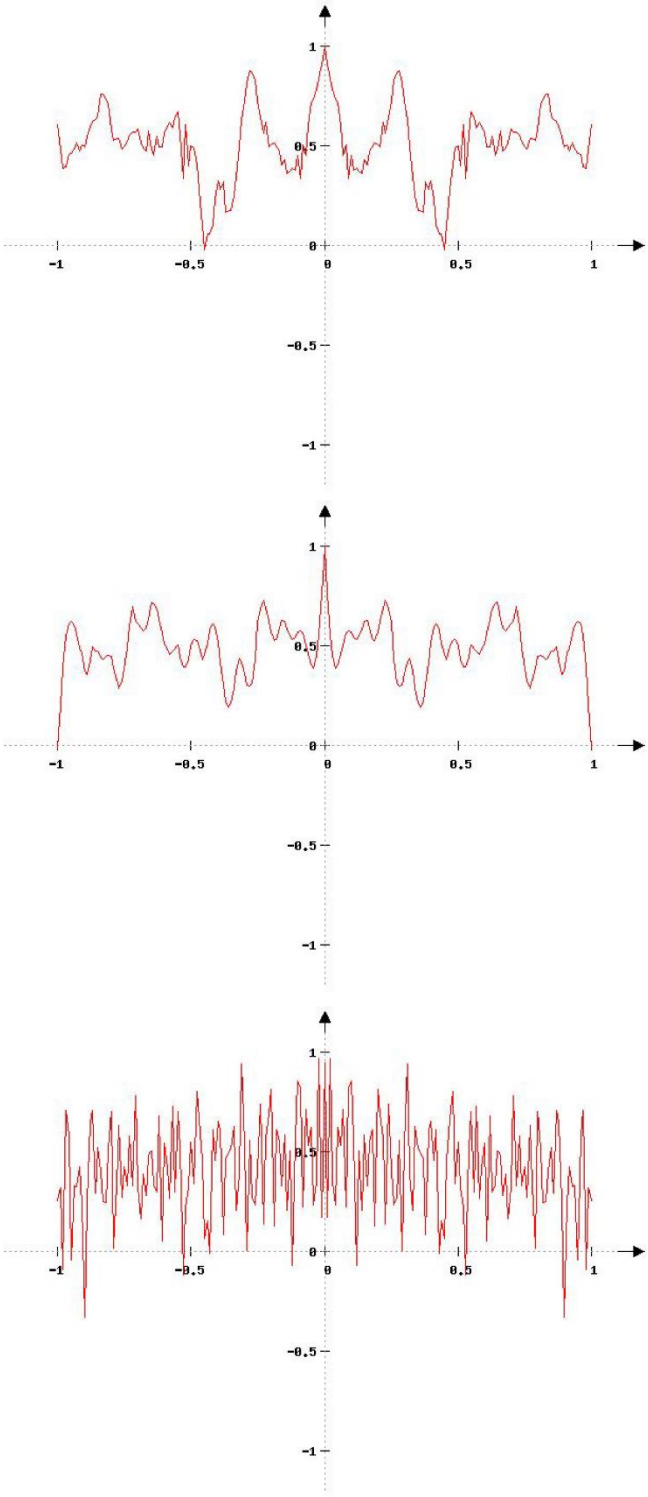

Figure 2. Best generations – 10, 50, 100 individuals

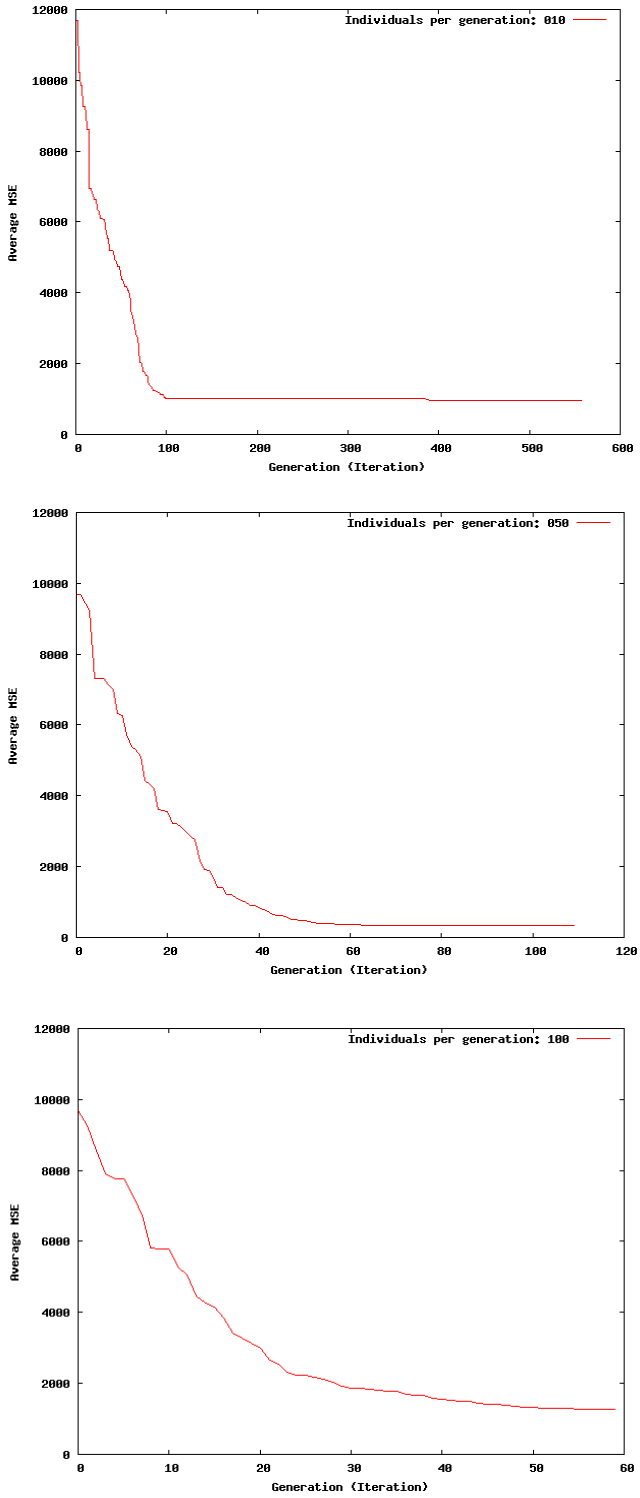

Figure 3. Convergence: MSE per generation for a population of 10, 50, 100 individuals

Another genetic algorithm-based approach

Among the best of classical filters is the Lanczos3 filter, which is defined on the interval [-3, 3] and is based on the "sinc" function. The question then becomes: is there another function, similar to the Lanczos3 – very close in the solutions space – that performs better than it?

There are multiple ways in which one may search for a neighbor it the Lanczos3 function. A very quick and simple way would be to modify the function slightly – there are many ways in which this can be performed – and check if the newly-generated function is better; by better we mean that the mean squared error (MSE) of the new function is lower than that of the original Lanczos3 function.

One key observation here is that this is basically an optimization problem: we have a very clearly defined way to analyze and compare our new functions and at least one way in which to generate new functions. Therefore we can use a genetic algorithm to find the best solution. Unlike most optimization strategies and algorithms – such as hill climbing or A* - genetic algorithms have the ability to preserve a seemingly bad solution, in the hope that it may spawn a very good solution down the line – which may also avoid local maximums.

The method follows the basic selection-breeding-mutation pattern of any such algorithm, with implementations specific to the problem at hand. The fitness function used is the MSE value: a lower MSE means a fitter individual.

Given that a functions has a relatively complex structure and that it is recommended to use a binary encoding for the individuals, the function is divided into position-based intervals, which makes the breeding and mutation operations easier to implements.

The first generations of individuals is composed of identical specimens, which are all a discretized version of the Lanczos3 function, with a relatively high number of samples (upward of 10,000).

At each iteration, the best individuals are selected to breed. Breeding is an interpolations of two equivalent intervals – same interval – from two selected individuals. In order to distance ourselves from the original Lanczos3 function as much as possible, the intervals chosen are those which are as dissimilar as possible from the original. To determine this, each interval is tested, and the absolute differences of both individuals are added up, one sum per interval. The interval with the highest sum, most different from the original, is chosen for interpolation.

The result is that we are taking the interval from one individual and merging it into the other. The small contribution of the original Lanczos3 function is there so that we don't obtain completely bogus results and we ensure that a neighboring function is generated.

After the breeding, we also keep a number of the best individuals from this generation, into the next generation. This ensure that best solution found so far is not lost.

The last step is to mutate some of the individuals. Here, we select from a wider spectrum, which includes some of the least fit individuals as well. The mutation itself is done by either lowering or raising all the values in a randomly chosen interval using a randomly generated value, linearly around the center of the interval, as shown in figure 4.

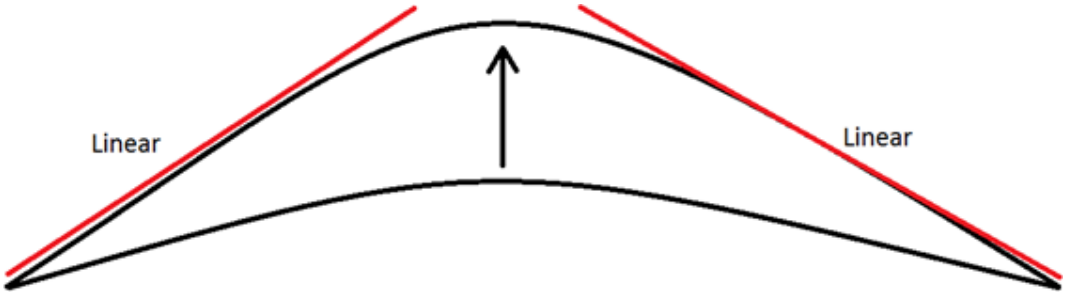

Figure 4. Linear raising of the values inside an interval

All of the above steps are done for a fixed number of iterations and, at the end, the best individual is selected as the final solution.

After a new individual is created – either by breeding or mutation – its fitness value must be computed. This is a very costly operation, as the filter needs to be used in multiple downscaling/upscaling calculations, performed on multiple images in order to determine its MSE. Given that the breeding and mutation steps are exactly the same, no matter which individuals are selected, and that they require a calling of the fitness function, they have been parallelized using OpenMP technology, in order to reduce the time need to run the entire algorithm – by increasing the throughput.

To give a sense of the runtime costs, on an Intel Core 2 Quad processor, clocked at 3.2 GHz with 4 GB or RAM, one iterations requires approximately 2 minutes to complete.

The filter found by the algorithm produced better results – there were no negative improvements, with a mean improvement of 3.5 %. Lower scales provided the best results, the maximum being an increase of 13.32 % over the original Lanczos3, at scale 1.5. As the scale was raised, the difference between the functions became smaller, almost insignificant in some cases, leading to the overall small 3.5% improvement.

The joint Bezier curves approach

The basic idea comes from the study of Lanczos (“sinc”-based) functions. The proposed algorithm creates quadratic Bezier curves on some specific markups and interpolates the distances between them. By using Bezier curves only on some specific spots, we reduce by a large amount the searching space. The values between curves are obtained by linear interpolation.

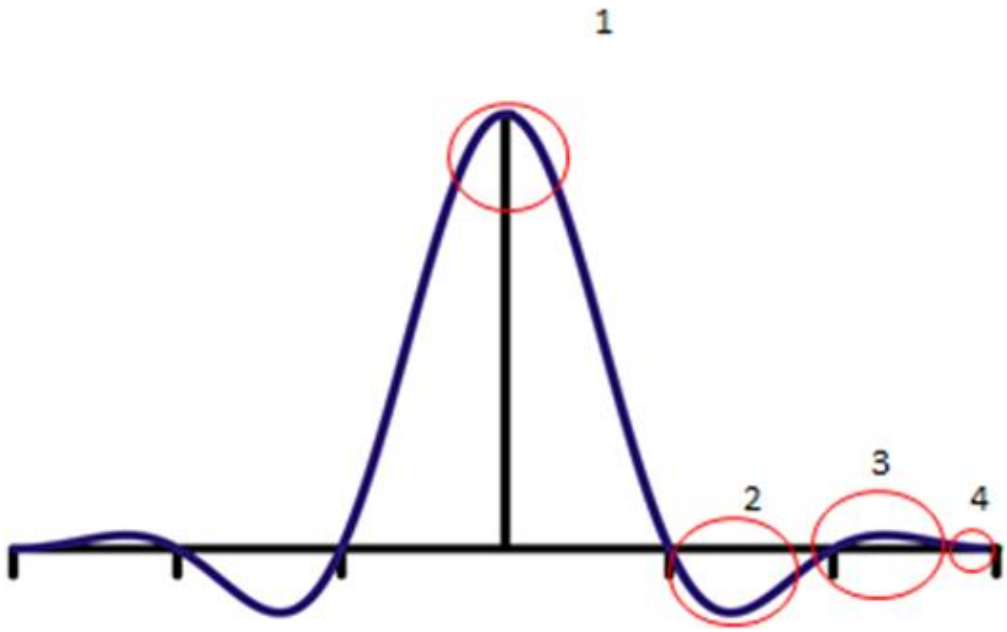

Figure 2. Placement of the Bezier curves

The Bezier curve are adjusted based on the previous results. For example, we start by choosing the Bezier curve which provided the best signal-convolution values when the input value was between [0, 1] (which is the case of the first Bezier curve). Processing each next Bezier was done based on the results of the previous one.

The primary impediment was selecting the exact ROI (Region Of Interest). The starting point is Lanczos3 because it provided the best results on tests. After setting the location of Bezier curves, the next problem was the large spectrum of possibilities that a curve parameters may have.

Table 3. MSE results on different scales and filters. Legend: The best filter performances are written with bold font.

Filter Type	Scale 1.5	Scale 2.0	Scale 2.5	Scale 3.0	Scale 3.5	Scale 4.0	Scale 4.5	Scale 5.0
Lanczos64	**32.867**	**83.738**	**124.140**	**153.505**	**176.009**	**194.806**	**210.997**	**225.246**
Lanczos3	49.846	98.752	138.652	167.539	191.277	209.758	224.687	239.294
Proposed	56.567	110.183	151.862	174.874	196.613	219.325	231.672	250.859
Mitchell	87.059	134.002	169.426	196.414	218.783	238.004	255.768	271.794

The brute-force windowing approach

We will start with the following assumptions:

- A good filter function will need to contain the "sinc" function along with a suitable windowed function. The problem reduces itself at finding the most suitable windowed function.
- A suitable windowed function can be a cosine series function: $$\sum_{i=0}^{M} a_i * \cos\left(\pi * i * \frac{t}{N}\right);$$

  where: $\sum_{i=0}^{M} a_i = 1; M = number\ of\ terms; N = window\ size$

Trying out all the possible combinations for the 'a' terms, we conclude that the best results are indeed for a window that has the largest ripple in the first position and smaller ripples following. Thus, for testing purposes, a0 will vary between 0.4 and 0.6 regardless of the number of terms.

An application was created that backtracks the 'a' terms such that their sum is equal to 1, and tested all possible combinations on several images and for various scales. Various numbers of terms were used. Besides this, the window of the filter was varied.

Because of memory management and efficiency needs, the algorithm pseudocode is the following one:

```
Foreach term_count:

➔ Backtrack over all possible terms combinations and store then in a
global array
➔ When reaching a limit L, process the gathered combinations (because
of limited memory):
    ▪ Foreach image,
        • Foreach scale
            ○ Foreach window
                ▪ Foreach term_combination
                    • Compute MSE and store result
    ▪ Cycle through the results, pick the best for each pair
      <image , scale>, remember just those and discard the rest
➔ Continue the backtrack writing the new combinations over the old ones
in the global array (so no allocation time is wasted).
```

The larger the filter window, the better the PSNR. However, testing a window of 3 versus a window of 70 just gave a means of a 10% increase, but a huge increase in processing time. Thus, for testing a larger number of terms, we chose a fixed window of 4.

The best results were obtained for an a0 term between 0.4 and 0.6, meaning a strong main ripple followed by small secondary ripples. The best general results of fully testing 2 and 3 terms filters are the following: [0.49, 0.49, 0.02] closely followed by [0.51, 0.49].

The brute-force by backtracking approach

The proposed approach consists in generating all possible filters using backtracking. The relevant values belong to the filtering function graph. Other values are obtained by linear interpolation.

There are available two methods of generating filters. In the first method, generated functions can have values from the interval [-1, 1]. The disadvantage is that in this way there are generated many inaccurate filters. Using the other method, generated functions will have closer values to a stable filter.

To be able to perform upsampling and downsampling of images, the resulting filters are normalized. The values are close to Lanczos filter. For searching functions there were considered 13 points on horizontal axis and 5 on vertical axis. This is, of course, a strongly sub-optimal searching space. It is perhaps the approach that may lead to the best solution for a set of input images but also, impossible to implement using a fine granulation, due to the huge brute-force complexity.

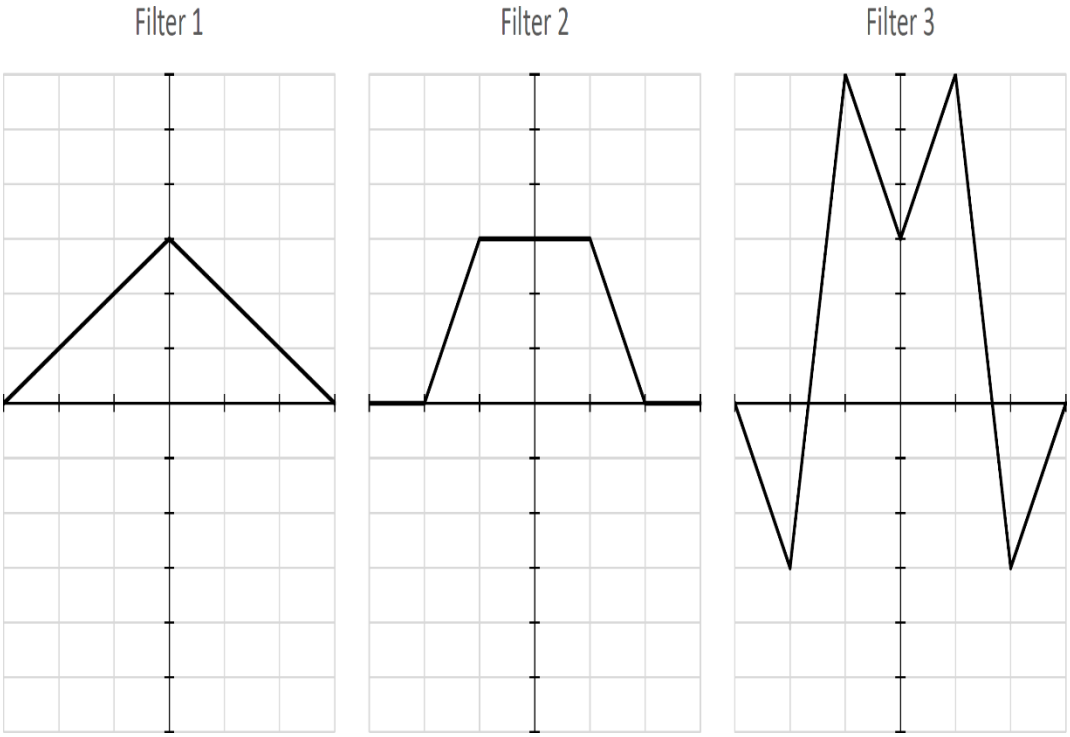

Figure 3. The brute-force by backtracking approach evolution (3 milestones)

Table 4. MSE results on different scales and filters. Legend: The best filter performances are written with bold font.

Filter Type	Scale 1.5	Scale 2.0	Scale 2.5	Scale 3.0	Scale 3.5	Scale 4.0	Scale 4.5	Scale 5.0
Box	4668.875	254.158	1609.515	183.285	1146.723	304.669	918.518	351.562
Hermite	43.740	71.198	105.484	135.434	167.569	196.987	224.749	255.110
Triangle	124.510	71.198	126.687	149.781	185.393	208.648	244.758	275.346
Bell	68.778	109.490	149.429	188.006	226.622	262.047	295.006	329.829
B-Spline	84.815	131.984	177.501	220.681	262.969	301.332	337.074	373.629
Lanczos	**25.641**	50.033	**74.504**	**99.347**	**125.318**	**150.386**	**173.411**	202.016
Mitchell	42.701	74.266	107.098	138.517	171.129	201.685	230.190	262.046
Proposed	35.851	**46.292**	83.306	99.884	133.766	151.899	175.897	**200.004**

Brute-force Bezier generator approach

Implementing filtering on GPU has efficient results, easily obtaining 50 times speed-ups, comparing to the classic implementation on CPU. A brute-force method is used to detect other classes of filters as efficient as the ones in literature.

Filters are obtained by interpolation of a Bezier Cubic Curve with 4 control points. First point is always fixed, otherwise leading to redundancy.

From the Bezier Curve resulted after running an optimal search algorithm of Brute-Force Greedy type, or after user's manual parametrization, there are extracted N points {x,y} that represents filter discretization. The number of points must be chosen in such a way that the resulting Bezier Curve is not be affected by consecutive interpolation of values. For the actual carried tests, N is 101.

A binary search approach is used for getting the filter value in a point. The value that is returned will be the result of linear interpolation (Y-axis) of the 2 points resulted after running the binary search.

Table 5. MSE results on different scales and filters. Legend: The best filter performances are written with bold font.

Filter Type	Scale 1.5	Scale 2.0	Scale 2.5	Scale 3.0	Scale 3.5	Scale 4.0	Scale 4.5	Scale 5.0
Box	131.56	151.727	288.279	336.639	376.511	329.401	545.951	517.142
Hermite	102.391	135.55	223.602	274.268	291.813	281.613	420.985	333.657
Triangle	95.9473	135.55	235.093	243.197	281.054	293.657	437.536	409.742
Bell	132.036	183.865	227.969	266.085	301.987	331.66	390.504	394.206
B-Spline	155.828	209.092	256.181	296.13	332.428	364.734	399.56	426.853
Lanczos	59.8015	**104.73**	168.13	**175.591**	**204.913**	**232.962**	**274.732**	312.508
Mitchell	94.951	140.487	184.127	219.389	251.364	278.261	310.606	333.657
Proposed	**59.6421**	109.884	**144.303**	178.847	208.466	233.698	287.02	**289.303**

CONCLUSIONS AND FUTURE WORK

Although the results are good, they are not extraordinary. This leads to the natural question of whether we can significantly improve already great filters like those in the Lanczos family on every input image set. However, we have found that on fixed image sets and with fixed resampling scales, with enough time to let the filter function search operation work, valuable results different from the classical ones may be found.

The best filters found were of type “sinc” with cosine series windows. As the number of terms increases for the window, the results improved, but so the time spent on brute-force searching of the result. If this algorithm will be run on a cluster of GPUs, the computational time would be decreased dramatically and better results shall be found.

Even though, the resulted filters may obtain a better MSE than the classical, mathematically-defined ones. The results clearly show that this approach is viable and that it has potential for obtaining filters specialized for particular classes of images.

In the genetic approach department, it is visible in the evolution of the filters that they converge to a lower average MSE and that for a larger population there is more diversity among the individuals. Considering this observation, a higher mutation rate could be used to obtain a more diverse population and hence avoid being stuck in a local minimum. Furthermore, other recombination or mutation techniques could be used to generate new individuals.

The advantages of the Brute-Force Bezier Generator is that it restraints the search region of each Bezier curve to a specific zone. The next logical step is to use cubic Bezier curves and to try to find more exactly what the best values for each resampling scale are. An algorithm that could function in this case and reduce the computational time is the one that advances only in the direction which gives best result.

The proposed methods could be used to find new filters that could be even better than those available. For the Genetic-Based approach there are many changes that could potentially enhance the method. We are in disadvantage because this is in essence a blind search – meaning that we don't know what the perfect filter should look like – so it becomes a case of trial and error.

The way individuals are bread and mutated may have a tremendous effect on the results. For example, when breeding individuals, we could merge two intervals instead of just one; the same with mutation. The best strategy would be to generate more output data for each step of the algorithm and then develop a tool that can analyze and find out which change has the greatest effect and where it is best to invest the most effort.

Finding a good, solid resampling filter may lead to immediate benefits in other related research areas like image storage using pyramidal approaches [11][12] or even super-resolution [13] or image deblurring [14].

ACKNOWLEDGEMENTS

The authors would like to thank students Sorina Sandu, Sabina Batranu, Ciprian Apetrei, Gabriel Ivanica, Victoria Sima, Alexandru Anghelache, Dorian Dogaru and Dragos Dumitrescu for their great support and assistance with this paper.

REFERENCES

[1] E. Ethan, S. Agaian and A. Panetta, "Algorithms for the resizing of binary and grayscale images using a logical transform", SPIE Proceedings Vol. 6497:

[2] Image Processing: Algorithms and Systems V, 64970Z, 27 February 2007.

[3] S. Saffinaz ,"An Efficient Algorithm for Image Scaling with High Boost Filtering", International Journal of Scientific and Research Publications, Vol. 4, Issue 5, May 2014, p. 1-5, ISSN 2250-3153.

[4] R.D. Turney and C.H. Dick, "Real Time Image Rotation and Resizing, Algorithms and Implementations", 1999, [online] Available: http:// www.xilinx.com / products/ logicore/ dsp/ rotation_resize.pdf , Accessed at: 11 May 2016

[5] J.W. Hwang and S. Lee, "Adaptive Image Interpolation Based on Local Gradient Features", IEEE Signal Processing Letters, Vol. 11, Issue 3, March 2004, p. 359 – 362, ISSN :1070-9908.

[6] J. Kopf, M. Cohen and D. Lischinksi, "Joint bilateral upsampling", ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007), Vol. 26, Issue 3, 2007.

[7] S. Avidan, A. Shamir, "Seam carving for content-aware image resizing", 2007, [online] Available: http:// perso.crans.org/ frenoy/ matlab2012/ seamcarving.pdf, Accessed at: 12 May 2016

[8] C. Hegde, O. Tuzel and F. Porikli , „Efficient Upsampling of Natural Images", 28 February 2015 , [online] Available: https:// www.researchgate.net/ publication/ 273067650_Efficient_Upsampling_of_Natural_Images, Accesed at: 16 May 2016.

[9] Y. HaCohen, R.Fattal, D. Lischinski, „Image Upsampling via Texture Hallucination", Computational Photography (ICCP), 2010 IEEE International Conference on, 29-30 March 2010, p.1-8, Print ISBN: 978-1-4244-7022-8.

[10] M. Markandeshwa, „Comparison Of Different Image Enhancement Techniques Based Upon PSNR & MSE", International Journal of Applied Engineering Research, 2012, Vol.7, Issue 11, ISSN 0973-4562

[11] Emil Calofir, Radu Ionut Dan, Vlad Lionte, Ion Bucur, "Image Reconstruction after A Succession of 2:1 Downsampling – Upsampling", Journal of Information Systems & Operations Management (JISOM), the Proceedings of Journal ISOM Vol. 8 No. 2, pp. 252-261, December 2014

[12] Mihai Cristian Tănase, Mihai Zaharescu, Ion Bucur, "2:1 Upsampling-Downsampling Image Reconstruction System", Journal of Information Systems & Operations Management (JISOM), the Proceedings of Journal ISOM Vol. 7 No. 2, pp. 294-299, December 2013

[13] Florin Manaila, Costin-Anton Boiangiu, Ion Bucur – "Super Resolution From Multiple Low Resolution Images", Journal of Information Systems & Operations Management (JISOM), the Proceedings of Journal ISOM, Vol. 8 No. 2, pp. 316-322, December 2014

[14] Alexandra Ghecenco, "Principles of Image Deblurring", Journal of Information Systems & Operations Management (JISOM), the Proceedings of Journal ISOM, Vol. 8 No. 2, pp. 488-497, December 2014.