Optimal Image Upscaling using Pixel Classification

G. Banupriya, C . R.Jerinsajeev

Abstract*— Image magnification generally results in loss of image quality. Therefore image magnification requires interpolation to read between the pixels. Generally the enlarged images suffer from imperfect reconstructions, pixelization and jagged contours. The proposed system provides error-free high resolution for real images. The basic idea behind the system comprises two basic steps: Fast Curvature Based Interpolation (FCBI) which involves the filling of missing values after zooming and Iterative Curvature Based Interpolation (ICBI) which involves the modification of the filled values. The results obtained from the simulation shows that the proposed interpolation algorithm improves the quality of the image both subjectively and objectively compared to the previous conventional techniques.

***Keywords*— Image enhancement, image processing, Image magnification, interpolation, jagged contours, NEDI, FCBI, ICBI, nvidia CUDA**

I. INTRODUCTION

Image upscaling (and more generally image interpolation) methods are implemented in a variety of computer tools like printers, digital TV, media players, image processing packages, graphics renderers and so on. The problem is quite simple to be described: we need to obtain a digital image to be represented on a large bitmap from original data sampled in a smaller grid, and this image should look like it had been acquired with a sensor having the resolution of the upscaled image or, at least, present a ‖natural‖ texture. Methods that are commonly applied to solve the problem (i.e. pixel replication, bilinear or bicubic interpolation) do not fulfill these requirements, creating images that are affected by visual artifacts like pixelization, jagged contours, oversmoothing. They obviously rely on the assumption that, in natural images, high frequency components are not equally probable if low frequency components are known and a good algorithm is able to guess the image pattern that would have been created by a higher resolution sensor better than other methods.

More effective non-iterative edge adaptive methods like NEDI (New Edge Directed Interpolation or iNEDI (improved NEDI), present a relevant computational complexity, even higher than that of many learning based methods. In this paper we propose a new image upscaling method able to obtain artifact-free enlarged images preserving relevant image features and natural texture. The method, as several edges directed ones, approximately doubles the image size every time is applied by putting original pixels in an enlarged grid then filling holes. The hole filling is done in two steps, linearly interpolating closest points in the direction along which the second order derivative of the image brightness is lower. After each hole filling step an iterative refinement is performed, updating the values of the newly inserted pixels by minimizing the local variations of the second order derivatives of the image intensity while trying to preserve strong discontinuities. The main contributions of our paper can be summarized in the following items:

- A review of constant covariance constraint used in the NEDI method with the proof of the relationship of that constraint with the second order derivatives smoothness used in our algorithm.
- A new algorithm for image upscaling based on the iterative smoothing of second order derivatives (ICBI, Iterative Curvature-Based Interpolation). The algorithm is initialized using a simple filling rule based on second order derivatives (FCBI, Fast Curvature-Based Interpolation) that can be considered an edge directed interpolation algorithm too.
- A GPU implementation of the ICBI method able to enlarge images at interactive frame rates.

The paper is organized as follows: Section II. A. gives the basic description of the particular class of image upscaling methods based on grid doubling and hole filling, B. describes the NEDI algorithm, showing that some of its drawbacks can be removed by changing the constant covariance constraint with a more restrictive one, then C. demonstrates the relationship between this constraint and the hypothesis of second order derivatives continuity used in our new ICBI method. Section D. describes the new method in detail and the experimental tests showing its advantages and the GPU implementation realized using the CUDA technology is described in Section III.

(a) Related Works

In **2001**, Morse et al. **[10]** proposed ***level set reconstruction technique*** which presented a method for constrained smoothing of artifacts attempting to produce smooth reconstructions of the image’s level curves while maintaining the image fidelity. The disadvantage was that the results were marginally improved but not significantly. Again in **2001**, Li et al**. [9**] suggested ***New Edge Directed Interpolation*** **(NEDI)** that filled the missing values with the weighted averages of valued neighbors using local edge analysis but this methodology had a limitation of high computational cost. In **2002**, Freeman et.al **[5]** suggested three ***example-based*** approaches for zooming an image. First by a change in the spatial frequency amplitude spectrum associated with image sharpening then by aggregating from multiple frames and then estimating the missing new values by comparing the high – low resolution pairs. Here cubic spline method is used to interpolate the low resolution image. . But the performance is affected by the training set and these suits only for images because for moving objects multiple observations of same pixel need to be used as more input data is involved.

Manuscript received on January, 2013.

G.Banupriya, M.E.II Year / Department of ECE/ Einstein college of Engineering,Tirunelveli,India.

C.R.Jerinsajeev, Assistant professor/Department of ECE/Einstein college of Engineering,Tirunelveli,India.

In **2002**, Battiato et.al **[2]** addressed the problem of enlarging the image by considering the information about discontinuities or sharp luminance variations.The limitation in this methodology is that this operation cannot be performed if the resources available are limited and also the storage space required are high.In **2004**, Philip et.al **[11]** suggested image interpolation by Pixel ***Level Data Dependent Triangulation***.Though this method captures small and local features it is unable to report the jagged artifacts related to the orientation of the edges resulting in degradation of quality. Then in **2005**, Chen etal**. [3]** proposed ***Fast Edge Oriented Interpolation*** that interpolated the image by analyzing the local structures of the image where the image quality promotion was only for very low bit-rate videos. Next by **2007**, Fattal **[4]** suggested a method through ***edge statistics*** involving the promotion of predicted intensity differences in the upsampled images from the edge parameters obtained at low resolution input. In **2008** ,Jian et.al **[12]** proposed a novel generic image prior - ***gradient profile prior*** for the gradient field of the natural image. But the limitation is that for noisy input low resolution image, estimation of the gradient profile might be inaccurate due to noise. Then in **2008**, Kim et al**. [8]** adopted ***example based learning*** for single image super resolution involving a map based on example pairs of input and output images. But here the performance was unpredictable. So we go for the new approach that involves FCBI and ICBI. In **2009**, Daniel et.al **[7]** suggested a unified framework for combining classical multi-image resolution and example-based super resolution. The method is fast and efficient however it is not capable of reproducing fine detailed cluttered region. So we go for a new algorithm that consists of Fast Curvature Based Interpolation and Iterative Curvature Based Interpolation.

II. PROPOSED METHOD

(a) Interpolation From Four Neighbors: Fast Methods And The Nedi Algorithm

The “edge-directed” interpolation algorithms that, each time they are applied, approximately duplicate the image size by copying original pixels (indexed by *i* and *j*) into an enlarged grid (indexed by 2*i* and 2*j*), and then filling the gaps with *ad hoc* rules obtaining the missing values as weighted averages of valued neighbors, with weights derived by a local edge analysis. Algorithms of this kind are the well-known data-dependent triangulation and NEDI .

In these methods, the higher resolution grid is usually filled in two steps: in the first one, pixels indexed by two odd values [e.g., darker pixel in Fig. 1(a)] are computed as a weighted average of the four diagonal neighbors (corresponding to pixels of the original image); and in the second, the remaining holes [e.g., black pixel in Fig. 1(b)] are filled with the same rule, as a weighted average of the four nearest neighbors (in horizontal and vertical directions).

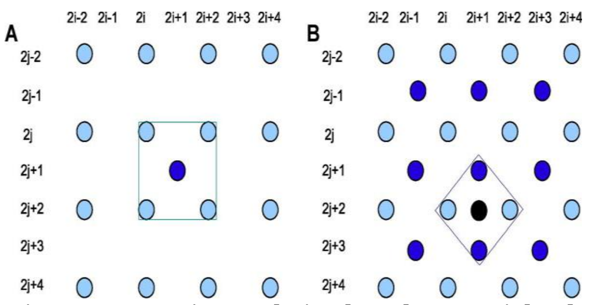

Fig.1. Two-step interpolation based on a weighted average of four neighbors

For example, for the first step, the interpolated value is usually computed as

$$I_{2i+1,2j+1} = \alpha.(I_{2i,2j,}I_{2i,2j+2,}I_{2i+2,2j,}I_{2i+2,2j+2}) \quad (1)$$

and specific algorithms of this kind differ for the way they estimate the coefficients vector $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ from the neighboring valued pixels in the grid.

In the data-dependent triangulation, the weighted average is computed setting to zero the weights of the two diagonally opposite pixels that differ more among themselves, and to 0.5 those of the other two. In the NEDI method the weights are computed by assuming the local image covariance (i.e., the vector) constant in a large window and at different scales. With this constraint, an over constrained system of equations can be obtained and solved for the coefficients. Images upscaled with this method are visually better than those obtained with the previously described methods, especially if some tricks are used to adapt window size and handle matrix conditioning . However, even applying the rule only in non uniform regions and using instead a simple linear interpolation elsewhere, the computational cost of the procedure is quite high.

(B) Constant Covariance Condition Revised: A Modified Well-Conditioned Nedi

If the locally constant covariance assumption used in NEDI, it is not ideal to model a classical step edge profile. In this case, the brightness changes only perpendicular to the edge and it means that the over constrained system solved to obtain the parameters is badly conditioned due to the rank deficiency of the problem (the expected rank of the matrix to be inverted is 2 and not 4). The simple solution we applied is to avoid computational problems consists of finding the minimum norm solution using the pseudoinverse. Finding a different constraint leading to a well-conditioned problem would be, however, more satisfactory, as in the ill-conditioned case, it would be possible to have a completely absurd pattern satisfying exactly the condition imposed to the local intensity.

We can obtain easily a better constraint by assuming that coefficients in multiplying opposite neighbors are equal. In this case, we can write

$$I_{2i+1,2j+1} = \beta.(I_{2i,2j,}I_{2i+2,2j+2,}I_{2i,2j+2,}I_{2i+2,2j}) \quad (2)$$

and assuming that this relationship is true with the same coefficients in a neighborhood of the point and also at the coarser scale as in the NEDI algorithm, write an over constrained system. In this case, the inverted matrix is full ranked. The solution is clearly faster (about 35% in this experiments) and most important, the quality of the interpolation is the same obtained with the NEDI method .

(c) Nedi Constraint and the Iterative Curvature Based Interpolation

If the condition 2 holds in a neighborhood and across scales, it is reasonable to think that an algorithm iteratively refining interpolated pixels by locally minimizing a function that should be zero when the constraint is valid would be effective in obtaining a good result. From 2, we have:

$$\beta(I2i,2j - 2I2i{+}1,2j{+}1 + I2i{+}2,2j{+}2) + \beta(I2i,2j{+}2 - 2I2i{+}1,2j{+}1 + I2i{+}2,2j) = (1 - 2(1\beta + 2\beta))\, I2i{+}1,2j{+}1 \quad (3)$$

The idea of ICBI is rather simple: after the computation of the new pixel values with a simple rule (in our case we take the average of the two neighbors in the direction of lowest second order derivative, an algorithm we called FCBI, Fast Curvature Based Interpolation), we define an energy component at each new pixel location that is locally minimized when the second order derivatives are constant. We then modify the interpolated pixel values in an iterative greedy procedure trying to minimize the global energy. The same procedure is repeated after the second interpolation step. Images obtained with this method do not present the evident artifacts; adding additional terms to reduce the image smoothing and heuristics to deal with sudden discontinuities, we obtained results that compare favorably with other "edge Based" techniques, with a computational cost that is compatible with real time applications.

(d) ICBI in Details

The two filling steps, as written before, are performed by first initializing the new values with the FCBI algorithm, i.e., for the first step, computing local approximations of the second-order derivatives $I_{11}(2i+1,2j+1)$ *and* $I_{22}(2i+1,2j+1$ along the two diagonal directions using eight-valued neighboring pixels.

$$\begin{aligned}I_{11}(2i+1,2j+1) = {} & I(2i-2,2j+2)+I(2i,2j)\\ & -3I(2i,2j+2)-3I(2i+2,2j)\\ & +I(2i,2j+4)+I(2i+2,2j+2)\\ & +I(2i+4,2j)\end{aligned}$$

$$\begin{aligned}I_{22}(2i+1,2j+1) = {} & I(2i,2j-2)+I(2\,i+2,2j)\\ & +I(2i+4,2j+2)-3I(2i,2j)\\ & -3I(2i+2,2j+2)+I(2i-2,2j)\\ & +I(2i,2j+2)+I(2i+2,2j+4)\end{aligned} \quad (4)$$

and then, assigning to the point $(2i+1,2j+1)$ the average of the two neighbors in the direction where the derivative is lower

$$I(2i,2j)+\frac{I(2i+2,2j+2)}{2}, if\ I_{11}(2i+1,2j+1) < (2i+1,2j+1)$$
$$I(2i+2,2j)+\frac{I(2i,2j+2)}{2}, otherwise$$

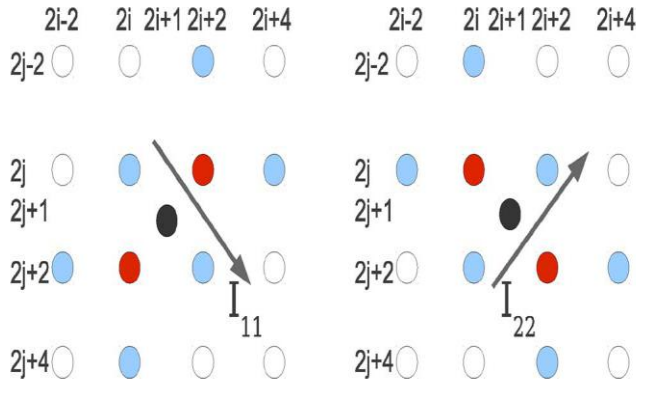

Fig . 2. At each step (here, it is shown the first), the FCBI algorithm fills the central pixel (black) with the average of the two neighbors in the direction of the lowest second-order derivative are estimated using for each one the eight-valued neighboring pixels (evidentiated with different colors).

Interpolated values are then modified in an iterative procedure trying to minimize an ‖energy‖ function. This function is obtained by adding a contribution for each interpolated pixel, depending on the local continuity of the second order derivatives and on other quantities that are minima when desired image properties are reached. The sum of these pixel components should be minimized globally by varying the interpolated pixel values. It is clear that the computational cost of the procedure could be high. We apply, however, a greedy strategy just iterating the local minimization of each pixel term. Being the initial pixel value guess obtained with FCBI reasonable, the procedure leads quickly to a local minimum that appears to be reasonable for our task. We said that the main energy term defined for each interpolated pixel should be minimized by small changes in second order derivatives. For the first interpolation step (filling gaps in the enlarged grid at locations (2i + 1, 2j + 1)), we defined this term as:

$$\begin{aligned}\mathrm{Uc}\ (2i+1,2j+1) = {} & w1\ (\,|(I11(2i,2j)-I11(2i+1,2j+1))|+|(I22(2i,2j)-I22(2i+1,2j+1))|)\ +\\ & w2\ (|(I11(2i,2j)-I11(2i+1,2j-1))|+|(I22(2i,2j)-I22(2i+1,2j-1))|)\ w3\\ & (|(I11(2i,2j)-I11(2i-1,2j+1))|+|(I22(2i,2j)-I22(2i-1,2j+1))|)+w4\ (|(I11(2i,2j)-I11(2i-1,2j-1))|+\\ & |(I22(2i,2j)-I22(2i-1,2j-1))|)\end{aligned} \quad (5)$$

where I11, I22 are local approximations of second order directional derivatives, computed as:

$$\begin{aligned}I11(2i+1,2j+1) = {} & I(2i-1,2j-1)+I(2i+3,2j+3)\\ & -2I(2i+1,2j+1)\end{aligned} \quad (6)$$

$$\begin{aligned}I22(2i+1,2j+1) = {} & I(2i-1,2j+3)+I(2i+3,2j-1)\\ & -2I(2i+1,2j+1)\end{aligned} \quad (7)$$

This energy term sums local directional changes of second order derivatives. Weights wi are set to 1 when the first order derivative in the corresponding direction is not larger than a threshold T and to 0 otherwise. In this way smoothing is avoided when there is a strong discontinuity in the image intensity. Assuming that the local variation of the gray level is small, second order derivatives can also be considered an approximation of the intensity profiles curvature. This is why we call this term a"curvature smoothing" term, and defined the algorithm "Iterative Curvature Based Interpolation" (ICBI). The optimization procedure minimizing the sum of the curvature smoothing terms is really effective in removing artifacts, but tends to create oversmoothed image. The smoothing effect can be only slightly reduced by replacing the second order derivative estimation with the actual directional curvature.

In our experiments we found more effective the addition of another energy term enhancing the absolute value of the second order derivatives:

$$\mathrm{Ue}(2i+1,2j+1) = -|I11(2i+1,2j+1)|+|I22(2i+1,2j+1)| \quad (8)$$

This term creates sharper images, but can introduce artifacts, so its weight should be limited. Another term we tested to reduce artifacts is related to isophotes (i.e. isolevel curves) smoothing. This is derived ,where an iterative isophote smoothing method is presented, based on a local force defined as

$$f(I) = \frac{-I_1(i,j)^2 I_{22}(i,j) - 2I_1(i,j)I_2(i,j)I_{12}(i,j) + I_{22}(i,j)^2 I_1(i,j)}{I_1(i,j)^2 + I_2(i,j)^2}$$

with I11, I22, I12, I1, I2 being local approximations of first and second order directional derivatives. The related energy term we applied is:

$$\mathrm{Ui}(2i+1, 2j+1) = f(I)|2i{+}1{,}2j{+}1 I(2i+1, 2j+1) \quad (9)$$

with I11, I22 computed as before and

$$\mathrm{I12}(2i+1, 2j+1) = 0.5(I(2i+1, 2j-1) + I(2i+1, 2j+3) - I(2i-1, 2j+1) - I(2i+3, 2j+1)) \quad (10)$$

$$\mathrm{I1}(2i+1, 2j+1) = 0.5(I(2i, 2j) - I(2i+2, 2j+2)) \quad (11)$$

$$\mathrm{I2}(2i+1, 2j+1) = 0.5(I(2i, 2j+2) - I(2i+2, 2j)) \quad (12)$$

Actually this term has a very small influence in improving the perceived and measured image quality. The complete energy function for each pixel location (2i + 1, 2j + 1), sum of the "curvature continuity", "curvature enhancement" and "isophote smoothing" terms becomes therefore:

$$U(2i+1, 2j+1) = a\mathrm{Uc}(2i+1, 2j+1) + b\mathrm{Ue}(2i+1, 2j+1) + c\mathrm{Ui}(2i+1, 2j+1) \quad (13)$$

Using this pixel energy, the first step of the iterative interpolation correction (adjusting pixel values with two odd indexes) is finally implemented as a simple greedy minimization as follows: after the placement of the original pixels at locations (2i, 2j) and the insertion of rough interpolated ones at locations (2i+1, 2j +1), we compute, for each new pixel, the energy function U(2i + 1, 2j + 1) and the two modified energies U+(2i + 1, 2j + 1) and U−(2i + 1, 2j + 1), i.e. the energy values obtained by adding or subtracting a small value δ to the local image value I(2i + 1, 2j + 1). The intensity value corresponding to the lower energy is then assigned to the pixel. This procedure is iteratively repeated until the sum of the modified pixels at the current iteration is lower than a fixed threshold, or the maximum number of iterations has been reached. The number of iterations can be also fixed in order to adapt the computational complexity to timing constraints. In our implementation we change the value of δ from an initial value of 4 to the unit value during the iteration cycle in order to speed up the convergence. a, b and c and T were chosen by trial and error in order to maximize the perceived and measured image quality. Note that the value of c and T are not critical (if T = Imax and c = 0) results are only slightly worse. If too large, the isophote smoothing term can introduce a bit of false contouring, flattening texture. The ratio between a and b determines a tradeoff between edge sharpness and artifacts removal. Actually, it may be also a reasonable option to use only the derivative-based constraint and to enhance contrast in post processing. After the second hole-filling step (assigning values to all the remaining empty pixels), the iterative procedure is repeated in a similar way, just replacing the diagonal derivatives in the energy terms with horizontal and vertical ones and iteratively modifying only the values of the newly added pixels.

III. EXPERIMENTAL RESULTS

For our experimental needs, we used images representing various objects, animals, flowers and buildings. These categories were chosen because they provide a wide range of colors and natural textures. Selected files were RGB color images with a depth of eight bits per channel. In all the previous equations we considered grayscale images; color images can be enlarged in the same way by repeating the procedures independently on each color channel or by computing interpolation coefficients on the image brightness and using them also for the other channels, reducing the computational cost and avoiding color artifacts. The high quality of the images obtained with the new method can be clearly seen comparing the images upscaled of the same factor with NEDI(see above snapshots).However, we also performed both subjective and objective tests in order to compare quantitatively the quality of the images created with different methods and the related computational.

(a) Objective Test

The objective test compares images obtained by downsampling the original images and then enlarging them with different methods, with reference images obtained just downsampling the original ones to the corresponding size. We performed this test on images converted to 8 bit grayscale, being the use of all three color channel not relevant to this test. Do not enlarge exactly the images by 2×/4× factors, being the exact enlargement at each step equal to (2width − 1)/width horizontally and (2height − 1)/height vertically. Finally we measured the differences between the upscaled images and the reference ones by evaluating the Peak Signal to Noise Ratio, defined as:

$$\mathrm{PSNR}{=}20\log 10(\mathrm{MAXPIX} / \textstyle\sum_{j=1}^{W}\sum_{j=1}^{H}(\mathrm{upscale(i,j)}\text{-}original(i,j))^2/\mathrm{W{*}H})$$

where $I_{upscale}(i,j)$ is the upscaled subsampled image, $I_{original}$ the original one, W and H the image dimensions and MAXPIX the end scale value of the pixel intensity.

I.Comparison of PSNR And Computation Time Of ICBI And NEDI

	ICBI	NEDI
PSNR Value	29.7638	21.533
Computation time	14.3906	61.875

(b) Subjective Test

In order to compare perceived image quality, we have taken some color images and enlarged them by a 2× factor with two different algorithms.

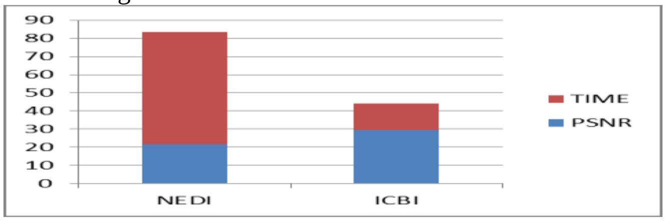

Fig 3: Comparison of Time and PSNR between NEDI and ICBI

(c) Image Sharpness And Arifacts

Subjective tests reveals that quality scores should be analyzed with care, being the perception of image quality related to image contents and to different factors that may be weighted differently according to the user's needs. The decrease in the perceived ‖image quality‖ is related to a linear combination of blurriness and artifacts, with higher weight given to blurriness (most people seem to prefer an increase in sharpness rather than a similarly noticeable artifact removal). This is probably one of the reasons why, for the enlargement of high resolution images for printing enhancement, photographers often use software that does not create natural detail or maximize similarity between high resolution patches and low resolution upsampled ones. Default options of professional photo zooming software usually strongly enhance contrast and straight lines, locally flattening texture. Of course, this is not necessarily a good choice if the enlarged bitmap should preserve detail recognition, realism and a correct depth perception from defocus. Learning based methods are also able to reconstruct sharp detail at the risk of creating ‖hallucinated‖ objects, and the perceived quality may be good or bad according to the fact that the detail is realistic or not in that position. It seems, therefore, a reasonable statement to say that there is not an interpolation method that is ideal in any condition: the choice of the algorithm is largely dependent on the application. The ICBI method proposed here is, in our opinion extremely effective in removing sampling artifacts, even if it does not enhance strongly lines and contrasted edges and results appear a bit oversmoothed. If the user wants to obtain images with less defocusing and enhanced contours, the final result can be, however, post-processed with sharpening filters to obtain a more contrasted image or clearer lines, without creating texture appearing too artificial or "painted".The other good feature of the method here proposed is the low computational complexity that allowed us to obtain real time performances with a GPU implementation.

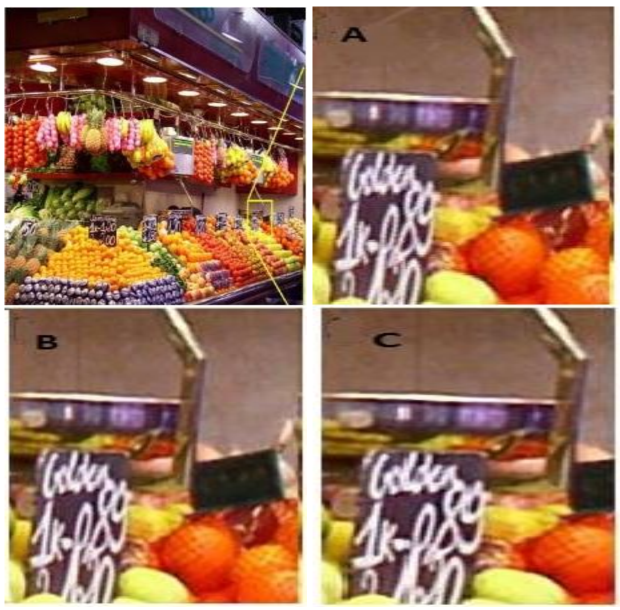

Fig 4. 4× upscaling of a 4 Megapixel image (not downsampled). A: Nearest neighbor enlargement showing a small detail at the original resolution. B: Same detail enlarged with ICBI: pixelization is removed without creating evident jaggies or artifacts, but the image appears oversmoothed. C: The same upscaled detail in B after a simple post-processing (selective smoothing and sharpening) enhancing the perceived quality of the printed image.

In the most recent generations of Graphic Processing Units (GPUs), the capacities of per-pixel and texturing operations have greatly increased. Millions of these GPUs are already present in the computers of consumers worldwide. Today you can easily apply those texturing and pixel engines, originally designed for 3D modeling and rendering, to many classic image-processing problems to provide tremendous speed increases over CPU-only implementations—and without any compromise in final image quality. This short introduction describes the basic methods of GPU usage for image processing and provides useful pointers to documentation, demo programs, and other developer tools.

In GPU each pixel in the rendered image can have image-based texturing applied (up to 16 simultaneous input images per pass can be accessed), and each pixel can run one or more small programs, called *pixel shaders*, to generate the final output color at each individual pixel. The GPU executes these shaders for many pixels at a time in parallel. Multiple passes of rendering may be executed, and the GPU provides additional image-blending hardware to permit images to be built-up in composited layers of arbitrary complexity. The results of each rendering pass, or any disk image, can likewise be passed back into the GPU pixel shader engine as another texture. This means that arbitrarily complex compositing operations can also be expressed as pixel shader operations. Image pixels can even be used as address indices into other images.

Once in the GPU, data flows in parallel for vertices and fragments. One can think of rendering as a SIMD problem where each instruction is executed on parallel streams of *vertices* or *fragments*. These streams only interact in the pipeline, either through the texture buffers or through accessing information in the frame buffer via multiple pipeline passes. In this way, one can think of the texture and frame buffers as pipeline registers. Since each stage is executed in lockstep, RAW hazards are avoided. In addition, by using two buffers, we can avoid RAW hazards between the original image to be processed and intermediate results. In our application we traverse the rendering loop on the GPU for each orientation we are searching.

In vertex processor the first stage through the pipeline for our application is determining the coordinates for the texels (texture primitives), and the color at each texel location. In this stage of the pipeline, the texel color can be altered by shading or blending with the underlying polygon. However, in our application, the vertex program execution is a compulsory part of the pipeline, where no actual computations are performed. Fragment programs are used to implement these per-pixel operations.

(d) Cuda Implementation And Real Time Interpolation

CUDA is a technology developed by nVidia allowing programmers to write code that can be uploaded and executed in recent nVidia graphics cards, exploiting their massively parallel architecture in order to obtain a relevant reduction of the computing time. C++ developers can write particular functions called ‖kernels‖ that can be called from the host and executed on the CUDA device simultaneously by many threads in parallel.

Using this technology, we implemented the ICBI algorithm by creating several CUDA kernels corresponding to the different steps of the algorithm. In this way computation performed in different blocks of the image can be executed in parallel, while the execution of the different steps is synchronized (see Figure 5). A first kernel creates the high resolution image from the low resolution one, a second fills odd pixels with the FCBI method, then two kernels computing derivatives and correcting the interpolated values are executed repeatedly. The second interpolation step is implemented in the same way, with a first kernel inserting new pixel values, and the iterative call of the two kernels computing derivatives and locally changing the interpolated values optimizing the energy function. With this implementation, we obtained the 4× enlargement of 128×128 color images in 16.2 ms on average, corresponding to a ideal frame rate of 62 frames per second and the 2× enlargement of 256 × 256 images in 12.3ms on average using a nVidia GeForce GTX280 graphic card (240 cores) and obtaining the same image quality of the Matlab and C version of the code. This example implementation clearly shows the possibility of applying ICBI for real time applications.

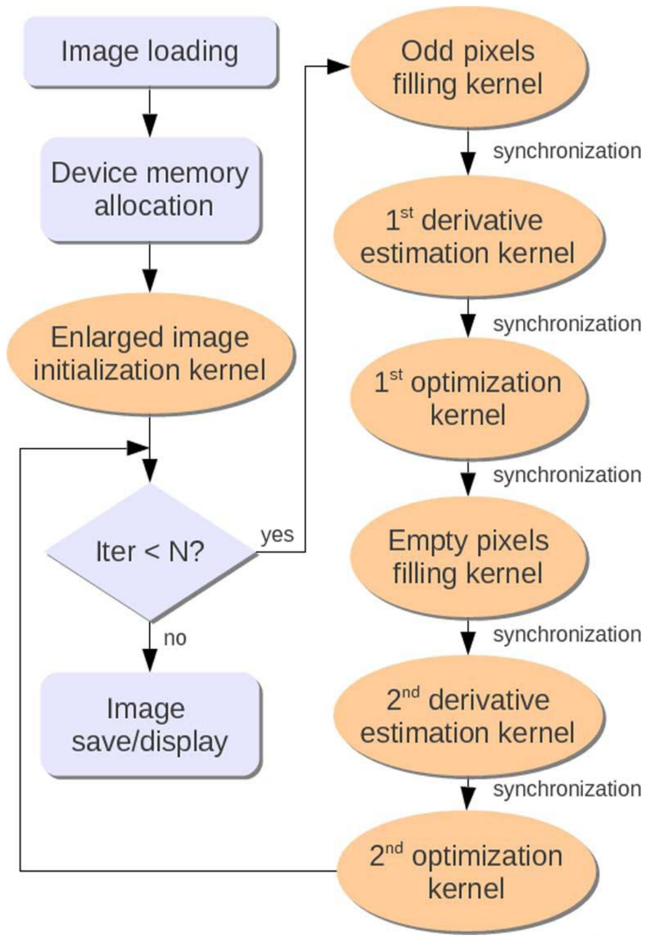

Fig.5. Flow chart representing the execution of the CUDA ICBI implementation. Ellipses represent kernels where matrices are processed in parallel creating multiple threads each one processing a separate block.

(e) Snapshots And Experimental Results

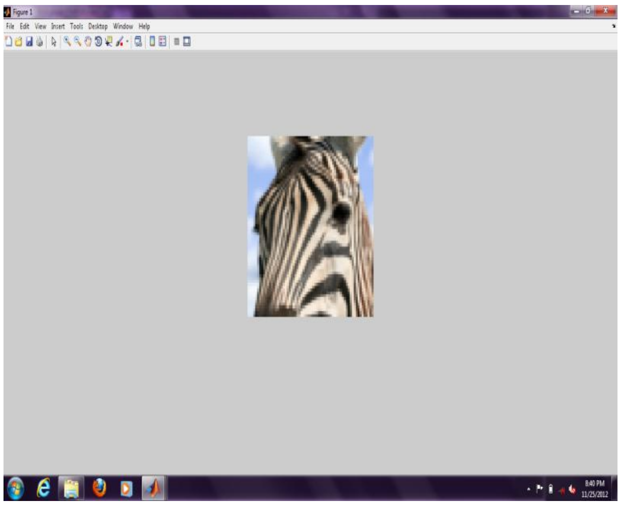

Fig 6.Original low resolution image

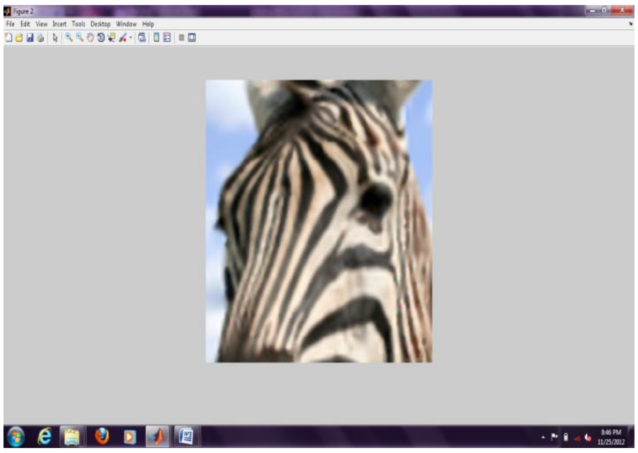

Fig.7. upscaled image by using icbi technique

This PSNR value and Required Cpu Time For Execution For Above Images

peak signal to noise ratio in db = 22.5061 required CpuTime in sec = 19.4063

IV. CONCLUSION

In this project we discussed several issues related to the problem of creating high quality upscaled images from low resolution original data. By using New Edge Directed Interpolation (NEDI) method can be leads to creating more time for program execution due to ill conditioned over constrained systems of equations and obtaining the high image quality compared to previous methods like Nearest Neighbour, Bicubic, and Bilinear Interpolation Techniques but we need to slightly reduce computation time. Then we showed used in our new ICBI (Iterative Curvature Based Interpolation) technique. This technique uses mainly the assumption that the second order derivatives of the image brightness are continuous along the interpolation directions and is able to obtain very good results, especially for its ability of removing artifacts without creating artificial detail, as proved by our objective and subjective tests. The new technique, based on a greedy minimization of an energy function that includes Curvature continuity, Curvature Enhancement And Isophote Smoothing defined at the interpolated pixel locations, is not computationally expensive like example based methods or the NEDI procedure and it is easily parallelizable.

REFERENCES

1. Andrea Giachetti and Nicola Asuni, "Real-Time Artifact-Free Image Upscaling".IEEE Transactions on Image Processing, Vol. 20, No. 10, October 2011
2. Battiato. S, Gallo. G, and Stanco. F, "A locally-adaptive zooming algorithm for digital images,"Image Visual Computing. , vol. 20, 2002, pp. 805–812.
3. Chen. M. J, Huang. C. H, and Lee. W. L, "A fast edge-oriented algorithm for imageinterpolation," Image Visual Computing. , vol. 23, 2005, pp.791–798.
4. Fattal.,R,"Image upsampling via imposed edge statistics",ACMTransactionsonGraph.,vol.26,no.3,2007,pp.95-1-95 8.
5. Freeman. W. T, T. R. Jones, and E. C. Pasztor, "Example-basedsuper-resolution,"IEEE computer Graphic application,vol.22,no.2,Mar./Apr. 2002,pp.56-65.
6. Gilad Freeman and Raanan Fattal, "Image and video upscaling from Local self-Examples",in proceedings of 12th International conference computer vision,2009,pp.349-356.
7. Glasner.D,Bagon. S, and Irani. M,"Super-resolution from a single image," in proceedings of 12th International conference on computer vision.,2009,pp.349-356.
8. kim. K.I and Kwon. Y,"Example-based learning for single - image super- resolution,"in proceedings of 30th DAGM SymposiumonPatternRecognition,Berlin,Heidelberg,2008,pp.456-465
9. Li. X and orchard. M. T,"New edge-directed interpolation," IEEE Transaction on Image processing, vol. 10, no. 10,oct. 2001,pp. 1521-1527.
10. Morse. B. S and schwartzwald.D,"Image magnification using level set reconstruction", in proceedings of IEEE conference on Computer vision and pattern recognition ,2001,vol. 3,pp.333-340.
11. Su.D and willis. P,"Image interpolation by pixel level data-dependent triangulation," computer Graphics Forum,vol.23,2004,pp.189-201.
12. Sun Jian, Z.B.Xu, and H.Y .shum,"Image super-resolution using gradient profile prior,"in proceedings of IEEE on computer vision and pattern recognition(CVPR),2008,pp. 1-8

AUTHORS PROFILE

G.Banupriya, currently pursuing M.E in Applied Electronics stream at Einstein college of Engineering, Tirunelveli (Dt). Completed B.Tech in ECE at Kalasalingam University in the year 2011.

C. R. Jerinsajeev, Currently working as an Asst. Professor at Einstein College of Engineering, Tirunelveli (Dt). Completed B.E in ECE from the C.S.I Institute of technology, Nagarkoil and M.E (Communication systems) from K.L.N college of Engineering , Madurai. He has five years of teaching experience.