REAL-TIME SINGLE FRAME SUPERRESOLUTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CEM TARHAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2014

Approval of the thesis:

REAL TIME SINGLE FRAME SUPER RESOLUTION

submitted by **CEM TARHAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** ____________

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** ____________

Prof. Dr. Gözde Bozdağı Akar
Supervisor, **Electrical and Electronics Engineering Dept., METU** ____________

Examining Committee Members:

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering Dept., METU ____________________

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Dept., METU ____________________

Assoc. Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Dept., METU ____________________

Assist. Prof. Dr. Fatih Kamışlı
Electrical and Electronics Engineering Dept., METU ____________________

Cahit Uğur Ungan, M.Sc
SST, ASELSAN Inc. ____________________

Date: 05/06/2014

ABSTRACT

REAL-TIME SINGLE FRAME SUPERRESOLUTION

Tarhan, Cem
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Gözde Bozdağı Akar

May 2014, 94 pages

A demand in real-time applications for superresolution increases as the surveillance and low resolution camera usage is spread for cost optimization. In this thesis two real-time superresolution algorithms have been proposed. The first algorithm (NDUID) is composed of non-dyadic upsampling cascade for smooth interpolation and an edge enhancement block that uses a non-blind deconvolution. Second algorithm (EDAT) starts with Total Variation decomposition. The structure component is interpolated with a fast edge adaptive interpolation and the edges are enhanced using a shock filter. Texture channel only upsampled by bicubic interpolation. The EDAT is also implemented on a Spartan 3A DSP FPGA to demonstrate the results on a real application. Experimental results show that implemented algorithm is capable of upsampling 320x240 resolution images by a factor of two at 52 fps with better quality than other real-time algorithms.

Keywords: Super Resolution, Total Variation, Bilateral Filter, Edge Adaptive Interpolation, Non-Dyadic Upsampling, Non-Blind Deconvolution, FPGA, Real time algorithm.

ÖZ

GERÇEK ZAMANLI TEK RESİM SÜPERÇÖZÜNÜRLÜK

Tarhan, Cem
Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Mayıs 2014, 94 sayfa

Gerçek zamanlı süperçözünürlük uygulamaları, artan izleme kameraları ve yaygınlaşan düşük çözünürlüklü kamera kullanımı ile artmaktadır. Bu tezde gerçek zamanlı iki superçözünürlük algoritması önerilmiştir. İlk algoritmada (NDUID) çift-olmayan çözünürlük arttırma metodu ile düzgün kenarlı yüksek çözünürlüklü resim elde edildikten sonra akıllı ters evrişim ile yüksek frekanslar belirginleştirilmiştir. İkinci algoritmada (EDAT) toplam değişim bazlı ayrışım metodu ile resim iki kanala ayrılmıştır. Yüksek gürültü içeriği olan doku kanalı bicubic yükseltme metodu ile yükseltilirken, resmin görsel olarak önemli kısmı olan yapı kanalı kenar uyumsal eklenti metodu ile büyütülmüş ve şok filtre ile kenarlar belirgin hale getirilmiştir. Sonuç bu iki kanalın toplanması ile elde edilmiştir. EDAT, gerçek zamanlı bir uygulamada ispatlanmak amacıyla alan programlanabilir kapı dizisi (APKD,FPGA) üzerinde uygulanmıştır.

Anahtar Kelimeler: Süper Çözünürlük, Toplam Değişim, Çift-yönlü filtre, Kenar Uyumsal Eklenti, Çift-Olmayan Yükseltme, Akıllı Ters Evrişim, FPGA, Gerçek zamanlı uygulama.

To my family...

ACKNOWLEDGEMENTS

First and foremost I would like to express my gratitude to my supervisor Prof. Dr. Gözde Bozdağı Akar for her endless help, guidance, encouragement and support during this study.

I would like to thank to my employer, ASELSAN for supporting me and aiding me with the FPGA board that we have implemented our algorithms on.

Finally, I would like to express my love for my family, my mother (Ece Tarhan) my father (Bülent Tarhan). They have raised me, taught me, cared for me, loved me and they have made me who I am. I could not be without you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The surveillance becomes widespread with concerns of security. Although using a high resolution camera might provide the best accuracy, it is not cost effective for mass surveillance applications. To optimize the cost of a system, the cost for video acquisition must be reduced. Even though using a low resolution camera is the only method for cost optimization, it reduces the received image quality and increases complexity of machine vision algorithms for security. For that reason superresolution algorithms start to draw attention. Increasing the size of captured video with state of the art algorithms might compensate for the loss of information for the sake of cost optimization. Since surveillance is a real-time application, the superresolution algorithm is needed to be implemented in real-time. Using a low resolution camera also reduces the complexities and cost for transmission by reducing the bandwidth therefore its usefulness is doublefold.

In image acquisition a 3D image is projected on 2D senor arrays that capture the image with limited number of pixels. Superresolution algorithms use given low resolution pixels to estimate unknown high resolution pixels. There are many methods to consider such as Bayesian approaches, dictionary based learning methods, multiframe, single frame methods etc. As multiframe applications are arguably the best methods to estimate high resolution pixel information, they require internal storage of multiple frames and large amounts of data to be processed simultaneously. Similarly learning based methods also require a large dictionary of low-high resolution image patches to be used as a method to interpolate missing pixels.

To implement an algorithm in real-time with large amounts of data to be processed there are limited number of options such as field programmable gate arrays (FPGA) and digital signal processors (DSPs) that are generally used together with FPGAs. Today’s technology of FPGAs allows users to store up to 100MB of data to be processed simultaneously with up to 500MHz of clock rate although an FPGA with such properties costs over 2000$. To create an affordable system one needs to accept the tradeoffs. Therefore for a real-time application single frame superresolution is the most suitable choice.

1.2 Scope of Thesis

In this thesis, single frame approaches for superresolution have been analyzed. The aim is to come up with a method suitable for real time applications. Since the computational efficiency and parallel implementation requirements are necessary and useful for FPGA applications we have filtered the researched literature.

We have searched the literature for all possible methods for superresolution such as spatial and transform domain methods, learning based and statistical methods. We have found the fastest state-of-the-art algorithms.

We have implemented all of the analyzed algorithms on Matlab for comparison. The weaknesses of all algorithms are discovered and two new methods are proposed to improve available algorithms in the literature.

The proposed methods are tested against the literature and according to the numerical results we have chosen one algorithm that exceeds the quality of state of the art algorithms to be implemented on a Field Programmable Gate Array (FPGA).

The implementation results are collected and compared against frequently used algorithms such as bicubic. Our proposed method exceeded the quality of other real-time algorithms.

1.3 Outline of Thesis

In chapter 2 literature research about super resolution is going to be given. Concepts that are used inside the thesis are going to be clarified.

In chapter 3 four algorithms that are analyzed in detail are going to be formulized and elaborated with results of our own.

In chapter 4 two proposed methods are going to be described and compared against used algorithms and mainstream algorithms such as bicubic interpolation.

In chapter 5 FPGA implementation details, resource utilization and speed limitations are going to be given.

In chapter 6 FPGA results are going to be compared against theoretical results.

In chapter 7 future works will be described and conclusions are going to be made.

CHAPTER 2

LITERATURE RESEARCH

2.1 Image Observation Model

The image capturing systems are not perfect due to discrete elements used inside. Finite aperture size causes optical blur since it allows only an allowed distance to be captured without blur. The blur is modeled by a point spread function. Non-zero aperture time causes motion blur, discrete sensor elements cause sensor blur and aliasing limits the resolution of an image. Figure- 2-1 shows the diagram of image observation model.

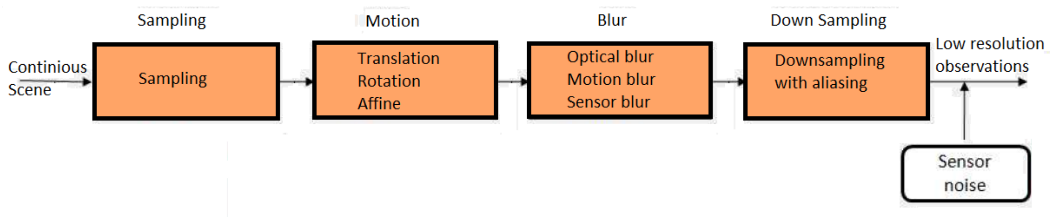


Figure- 2-1: Image Observation Model

The image generation is modelled by

$$\boldsymbol{Y} = D\ H\ F\ \boldsymbol{X} + V \tag{2-1}$$

Where X and Y are high resolution and low resolution images, D is downsampling operator, H models the blurring effect, F is motion information which is used for multiframe SR algorithms and V is the noise. The equation can be written in terms of a single matrix multiplication Y = M X + V where M is named as degradation matrix.

These matrices are very sparse and this equation is generally ill-posed. In real applications these matrices are unknown therefore they needed to be estimated which makes the problem more ill-posed.

The motion information is collected for multiframe algorithms because those algorithms are required to gather multiple frames under a single high resolution image. Since multiple frame usage requires too much storage capacity and increases data size to be processed we have limited our scope for singleframe approaches.

2.2 Spatial Domain Approaches

Spatial domain methods use only pixel information apparent in an image and try to interpolate new pixels by using low resolution pixels only. The degraded and downsampled high resolution image is to be recovered by these algorithms as shown in Figure- 2-2.

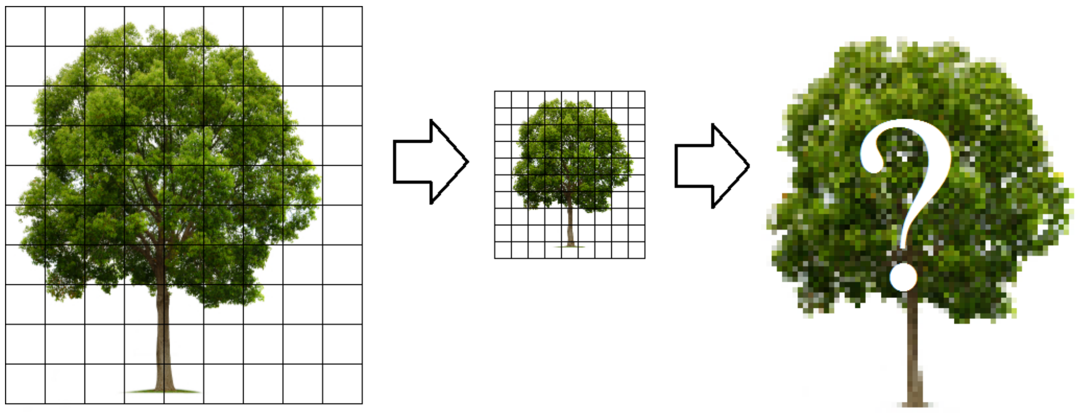


Figure- 2-2: Image Formation Reversal

2.2.1 Basic Methods

Time domain approaches use only pixel values without using any transform or library. Basic approaches such as bicubic and bilinear interpolation use predefined quadratic or linear functions to fit new pixels into the image. Even though these methods are simple to implement and fairly competitive in numerical comparisons they do not contain any statistical or prior knowledge therefore they are open for artifacts such as jagginess and aliasing.

2.2.2 Edge Adaptive Approaches

Adaptive interpolation algorithms have been introduced to create computational ease together with satisfying images for human visual system. Edge adaptive interpolations are divided into two classes, namely explicit and implicit methods. Explicit methods estimate edge directions explicitly and use the information to interpolate the low resolution image. This method can be problematic since the estimation of edge direction determines the overall quality. Since natural images have noise and aliasing effects it is inescapable to have degradations over estimation. Also for a real-time application edge orientation number should be limited for computational cost efficiency [1]. Implicit methods use parametric cost functions or statistical models to jointly estimate the edge directions and interpolate [1][2][3][4]. Siu et al [1] uses bilateral filter approach to estimate new pixel values. Since bilateral filter coefficients also needed to be estimated, geometric duality method is used [4]. Geometric duality is proposed by Li et al. [4] and it correlates low resolution image to high resolution image pixels by constraining their covariance matrices to be similar. Although the proposed new edge upsampling (NEDI) method is a useful tool it uses lots of matrix inversion. Siu et al. [1] have resolved the computational burden by introducing a method called BSAI which will be discussed in chapter 3. Other methods use LR image pixel (or group of pixels) information with statistical approaches to find edge direction.

Another interesting method has been suggested by Krishnan et al. [5] recently. The method uses spline approximation to estimate edges via bitmap tracing. The traced image is composed of vector information which is upsampled without any aliasing. Then the HR traced image is used on a guided filter, proposed by He et al. [6]. The guided image filter is based on bilateral filter which preserves edge information. This algorithm is not included in analyzed algorithms chapter since it uses bitmap tracing and such software that is for sure not implementable on an FPGA.

2.3 Transform Domain Approaches

Transform domain approaches are used to simplify filtering and deconvolution operations. Even though a 2D filtering operations becomes simple multiplication in transform domain, basic approaches create halos, ringing and other artifacts. This problem is overcome by the aid of image gradient statistics and adaptive or non-blind operations on Fourier domain [7][8][9][10].

Transform domain algorithms also contain wavelet domain applications. Wavelets transforms can be processed by statistical approaches to estimate unknown high resolution coefficients [11][12][13].

2.4 Statistical Approaches

Statistical approaches create stochastic models to optimally reconstruct high resolution image from low resolution image. The reconstruction problem can be described by a Bayesian framework where X is desired (high-resolution) image, Y is degraded (low-resolution) image, M is degradation matrix. The Bayesian framework is formulized as

$$\begin{aligned} X &= arg_x max Pr(X|Y) \\ &= \mathrm{arg_x max\, Pr(X, M|Y)} \\ &= \mathrm{arg_x max} \frac{\mathrm{Pr(Y|X, M)\, Pr\,(X, M)}}{\mathrm{Pr(Y)}} \\ &= \mathrm{arg_x max\, Pr(Y|X, M)\, Pr(X)\, Pr(\, M)} \end{aligned} \qquad (\ 2\text{-}2)$$

$\mathrm{Pr(Y|X, M)}$ is data likelihood and $\mathrm{Pr(X)}$ is prior information on desired image and $\mathrm{Pr(\, M)}$ is prior term of degradations. X and M are statistically independent. Pr(X) is typically [14] defined by using Gibbs distribution

$$\mathrm{Pr}(X) \propto \exp\,(\alpha A(X)) \qquad (\ 2\text{-}3)$$

where A(X) is non-negative potential function.

2.4.1 Maximum Likelihood (ML)

If a uniform prior for desired image is assumed Eq. (2-4) is reduced to ML estimation. ML estimator uses initial observations to estimate the most suitable solution which maximizes p(Y|X) which is assumed to be Gaussian generally. ML estimator is given as

$$\mathrm{X_{ML} = argmin\{\|Y - MX\|^2\}} \qquad (\ 2\text{-}4)$$

The solution to this equation is found by pseudo inverse method

$$\mathrm{X_{ML} = (M^T M)^{-1} M^T Y} \qquad (\ 2\text{-}5)$$

If $(\mathrm{M^T M})$ is singular there are infinitely many solutions i.e. the problem is ill posed. To create unique solution regularization is required, which will be discussed in next topic. The inverse matrix also constitutes a problem because of huge sizes of matrix inversion is time consuming. There are number of suggested methods in the literature.

2.4.2 Maximum a Posteriori (MAP)

Many works [15][16][17][18] have used MAP approach to the solution of statistical equations. Maximum a Posteriori estimation maximizes a posterior probability by estimating a priori information as in Eq (2-6). Commonly used MAP based reconstruction techniques are described below.

$$\mathrm{X} = \mathrm{argmax}\ \Pr(\mathrm{Y}|\mathrm{X},\mathrm{M})\Pr(\mathrm{X}) \tag{2-6}$$

2.4.2.1 Gaussian Markov Random Field

The GMRF is formulized as

$$A(X) = X^T Q X \tag{2-7}$$

Where Q is a symmetric positive matrix, capturing spatial relations between adjacent pixels in the image by its off-diagonal elements. Q is often defined [14] as $\Gamma^{\mathrm{T}}\Gamma$ where Γ acts as a derivative operation on X. The log likelihood of the prior in (2-6) is given as

$$\log p(X) \propto ||\Gamma X||^2 \tag{2-8}$$

This is also known as Tikhonov regulization. One interesting paper [19] uses GMRF to impose geometric regularity constraint on an image while interpolating the image with a MAP based cost function that is optimized by simulated annealing. Behnad et al [20] also uses hidden Markov Model for MAP estimation to estimate higher order statistical dependencies. Even though GMRF is useful for its analytical sides, it is known for over-penalization of sharp edges therefore creating blurry results.

2.4.2.2 Huber Markov Random Field

The over-penalization i.e. smoothing problem of GMRF can be addressed by using image gradients with heavier tails than Gaussian distribution which does not model the natural images well. This approach is called Huber MRF where A(X) Gibbs potential is determined by the Huber function

$$P(\mathbb{x}) = \begin{cases} \mathbb{x}^2 & |\mathbb{x}| < a \\ 2a|\mathbb{x}| - a^2 & otherwise \end{cases} \tag{2-9}$$

where $\mathbb{x}$ is first order derivative of the image. By using a prior, edges can be preserved while reconstructing.

In literature Shan et al [8] have used this prior to constraint a deconvolution process which will be discussed in detail on chapter 3. Also [7][21] have addressed their problems with similar approaches.

2.4.2.3 Total Variation

Total Variation (TV) norm is a gradient penalty function. It is used in denoising, deblurring as well as super-resolution approaches [22][23][24][25][26]. The TV criterion penalizes total amount of change in the image which is measured by an L1 norm

$$A(X) = \|\nabla X\|_1 \tag{2-10}$$

where ∇ is the gradient operation. L1 norm in the equation above favors sparse gradients, meaning it helps preservation of sharp edges while creating a local smoothness. Farsiu et al [16] formulized a method called Bilateral TV for robust regularization. Also [27] and later [22] have used TV regularization for image decomposition and separately upsampling HF and LF components. This decompoisition approach will be discussed in chapter 3 in detail.

2.5 Example Based Approaches

Example based approaches use the idea of using existing information from a single image or a set of images to estimate high resolution pixels.

Many methods [25] [28][29][30][31][32] use example based approaches. Example based approaches are divided into two categories. First category algorithms use a dictionary of low-high resolution patch pairs for interpolation. Main workload of these algorithms is to optimize dictionary search time and proper matching criteria. Second category of algorithms uses self-examples where high resolution patch is not extracted from a library but multiple scales of the same image. Second category of algorithms is suitable for a real-time application.

2.6 Comparison of Methods

Among aforementioned methods every approach has various advantages. Statistical methods, if modeled correctly, can solve image observation model inversion problem successfully. However statistical methods cannot estimate missing high frequency information because of ill-posedness. Methods with dictionaries have the advantage of incorporating already known high frequency patches therefore better reconstruction chance. However as the dictionary size increases the search algorithm and matching algorithm become more important. Various algorithms for dictionary search are incorporated. Furthermore the dictionary based methods need to handle temporal consistency to avoid flickering. Also dictionary based algorithms are not suitable for real-time algorithms. However fast the search algorithm is implemented on an FPGA with parallel implementation, the size for the dictionary limits the implementation. Among these methods we did not include multiframe approaches because of space limitation on FPGAs. Multiframe approaches, on theoretical basis, can reconstruct lost high frequency information successfully due to subpixel image registration. However the problem of image registration is difficult to handle due to imperfect motion vectors, estimation of optical flow, white noise and lost pixel information.

We have analyzed algorithms from literature to find their weaknesses and strong points. These discussions are done on chapter 3.

CHAPTER 3

ANALYZED ALGORITHMS

We have analyzed four main algorithms that span all methods for superresolution. The algorithms are summarized below and the details are going to be described later on in chapter 3.

Fast image/video upsampling [8] proposed an iterative filtered back projection based algorithm which uses a minimization function to minimize the gradient distribution error and constraint the outcome to be bound to the input data. Even though algorithm as an iterative scheme it has been calculated that 3-4 iterations are adequate.

Image and video upscaling from Local Self Examples [29] proposed a learning based algorithm which uses self-examples. The algorithm exploits an assumption which states that an image is similar to its upsampled version if small ratios are used. This assumption shortens the duration of patch extraction.

Fast image interpolation Using the Bilateral Filter [1] proposed a MAP based minimization problem to estimate bilateral filter coefficients in return to obtain the interpolated pixel value depending on the edge direction. The algorithm also uses a cost function to maintain continuity among neighbor pixels. The calculations are limited to four basic operations.

Superresolution Utilizing Total Variation Regularization and a Shock Filter [22] uses total variation regularization based image decomposition to separate noisy texture component and structure component of an image. The structure component is linearly interpolated and substituted to an edge enhancement block where texture component is only linearly interpolated. TV Decomposition block uses iterative operations, but the latency of each epoch is limited to a line time of image transmission. This is going to be analyzed and justified in chapter 5 in detail.

3.1 Fast Image/Video Upsampling

One of basic approaches of superresolution is inversion of image formation. A newly proposed algorithm [8] incorporates a pixel substitution block connected to a de-re-convolution cycle. The framework of the algorithm is based on the image formation model [33] as described in chapter 2.

Figure- 2-1 shows the image formation model. This model can be described with two steps: filtering and decimation. A low resolution image can be represented as

$$Y = (f \otimes \mathrm{X}) \downarrow^{\mathrm{d}} \qquad (3\text{-}1)$$

Where X is high resolution image, Y is low resolution image; f is discretized blurring filter and $\downarrow^{\mathrm{d}}$ is down sampling operator. This operation can be split into two parts as

$$X' = f \otimes X \qquad Y = X' \downarrow^{\mathrm{d}} \qquad (3\text{-}2)$$

X’ denotes linearly filtered high resolution image. This separation splits the problem into two. First problem is inverting the downsampling. Inversion of downsampling is an ill-posed problem since it is required to generate d^2 pixels from one input pixel. Solution to this problem is given [8] by using a pixel-substitution block in a recursive algorithm. The second part of the problem is inverting the filtering, i.e. deconvolution. Jie et al. [34] proposed a non-blind deconvolution for inverse filtering process.

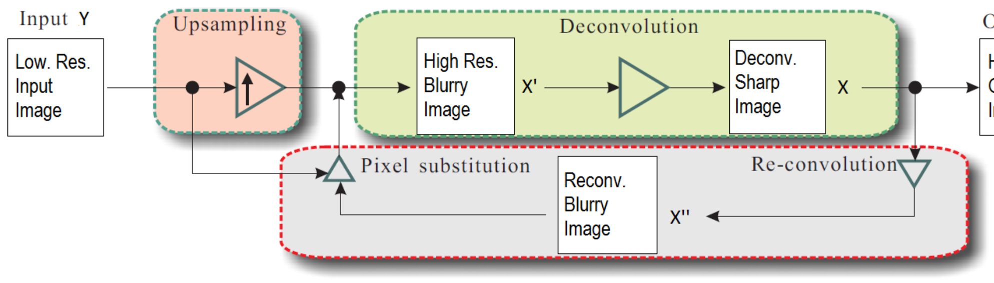


Figure- 3-1 : Framework for Image Reconstruction

Figure- 3-1 shows the framework mentioned in the paper. The algorithm incorporates a deconvolution step and an iterative feedback control to constrain the image upsampling.

3.1.1 Deconvolution

Deconvolution block estimates X using X' over the course of iterations. Deconvolution minimizes $||f \otimes X - X'||_2^2$. The formula can be written in the form of matrix multiplication therefore the minimization problem becomes minimization of $WX = Y$. Finding the inverse of W is not always possible therefore the problem may not have an exact solution. A prior term is introduced [8] to make the problem well-posed. It has been shown [35] that gradient distribution of natural images follows similar curves, thus using a gradient function φ(x) ensures a unique solution. The discrete blurring function is taken as a Gaussian kernel of size 13x13.

Figure- 3-2 shows the gradient distribution of natural images and a piece-wise continuous approximate function. Constant ‘k’ is given as 1.058, ‘a’ as 0.0002, ‘b’ as 13.97 and ‘lt’ as 13.

$$\varphi(x) = \begin{cases} -k|x| & x \leq lt \\ -(ax^2 + b) & x > lt \end{cases} \qquad (3\text{-}3)$$

Variable x is gradient of a pixel. After prior substitution, the minimization problem turns into

$$E(X) = ||f \otimes X - X'||_2^2 + k_1(||\varphi(\partial_x X)||_1 + ||\varphi(\partial_y X)||_1) \qquad (3\text{-}4)$$

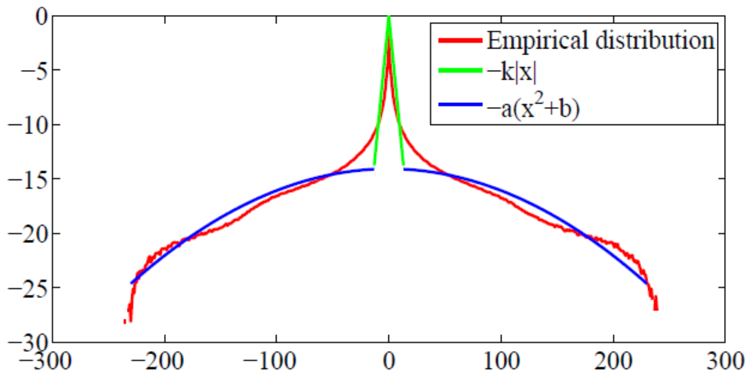


Figure- 3-2 : Logarithmic Distribution of Gradients and Approximated Curve

Variables μ_x and μ_y are substituted to gradient operators and a new minimization is defined in between to variables ($\mu_{x/y}$ $\partial_{x/y} X$)

$$E(X, \mu) = ||f \otimes X - X'||_2^2 + k_1(||\varphi(\mu_x)||_1 + ||\varphi(\mu_y)||_1) + k_2(||\mu_x - \partial_x X|| + ||\mu_y - \partial_y X||) \qquad (3\text{-}5)$$

The solution is obtained in two steps. First X is fixed to optimize μ variables.

$$E(\mu) = k_1(||\varphi(\mu_x)||_1 + ||\varphi(\mu_y)||_1 + k2(||\mu_x - \partial_x X|| + ||\mu_y - \partial_y X||) \qquad (3\text{-}6)$$

Consecutively, μ is fixed to minimize X.

$$E(X) = ||f \otimes X - X'||_2^2 + k_2(||\mu_x - \partial_x X||_2^2 + ||\mu_y - \partial_y X||_2^2) \qquad (3\text{-}7)$$

Square form of the energy minimization enables the usage of Parseval's theorem. Minimization becomes as in (3.8) where F is Fourier transform

$$E(X) = ||F(f) * F(X) - F(X')||_2^2 + k_2(||F(\mu_x) - F(\partial_x) * F(X)| + ||F(\mu_y) - F(\partial_y) * F(X)||_2^2) \qquad (3\text{-}8)$$

To obtain the optimal F(X) one needs to set $\partial E(X)/\partial F(X)$ to zero and the solution can be obtained as

$$F^*(X) = \frac{\overline{F(f)} * F(X') + k_2\overline{F(\partial_x)} * F(\mu_x) + k_2\overline{F(\partial_y)} * F(\mu_y)}{\overline{F(f)} * F(f) + k_2\overline{F(\partial_x)} * F(\partial_x) + k_2\overline{F(\partial_y)} * F(\partial_y)} \qquad (3\text{-}9)$$

Finally the result is obtained by taking inverse Fourier transform. Two parts iterate until convergence is achieved. It is recommended that taking k_2 as 20 and tripling its value on each iteration and taking $k_1$in between 0.01 and 0.3. 4 iterations were found to be enough for convergence. For 4 iterations the algorithm needs to use 13 FFT calculations.

3.1.2 Reconvolution

After calculating the deconvolved image, if iteration limit is not reached the image is reconvolved with the Gaussian kernel and submitted back to the feedback loop.

3.1.3 Pixel Substitution

The substitution process is done after each iteration of de-re-convolution. Low resolution pixel values are overwritten to their corresponding locations. Figure- 3-3 depicts this process.

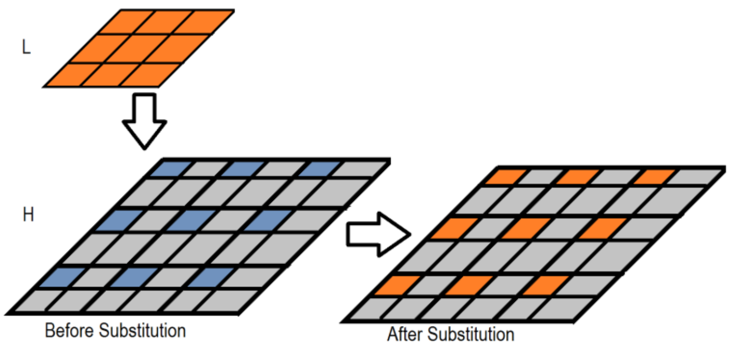


Figure- 3-3: Pixel Substitution Process

The benefit of this process [8] is that submitted pixel values are propagated through the high resolution image on the course of iterations. Therefore the upsampled image is constrained with the low resolution image.

3.1.4 Results

The implemented algorithm had artifacts due to pixel substitution block. The reconvolved image is altered due to prior operations therefore does not match the original image. This is shown in Figure- 3-4. Since the paper had given their results in only pictures and not in numerical figures, our numerical comparisons are going to be carried out with our results.

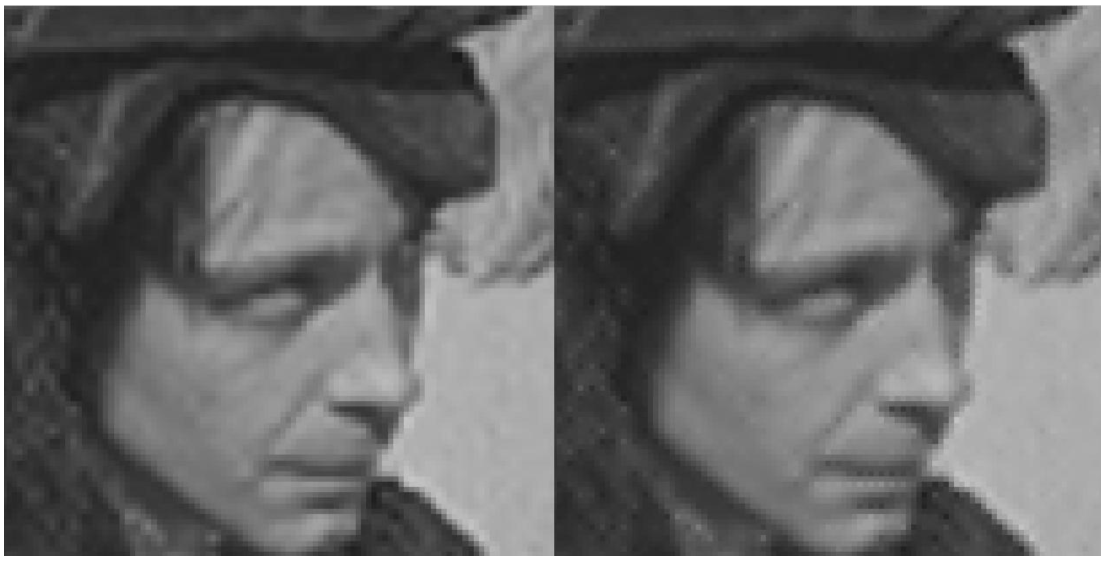


Figure- 3-4: Initial Upsampled Image and Pixel-Substitution Artifacts on Reconvolved Image

The iterative algorithm iterates on this artifacts and yield with Figure- 3-5.

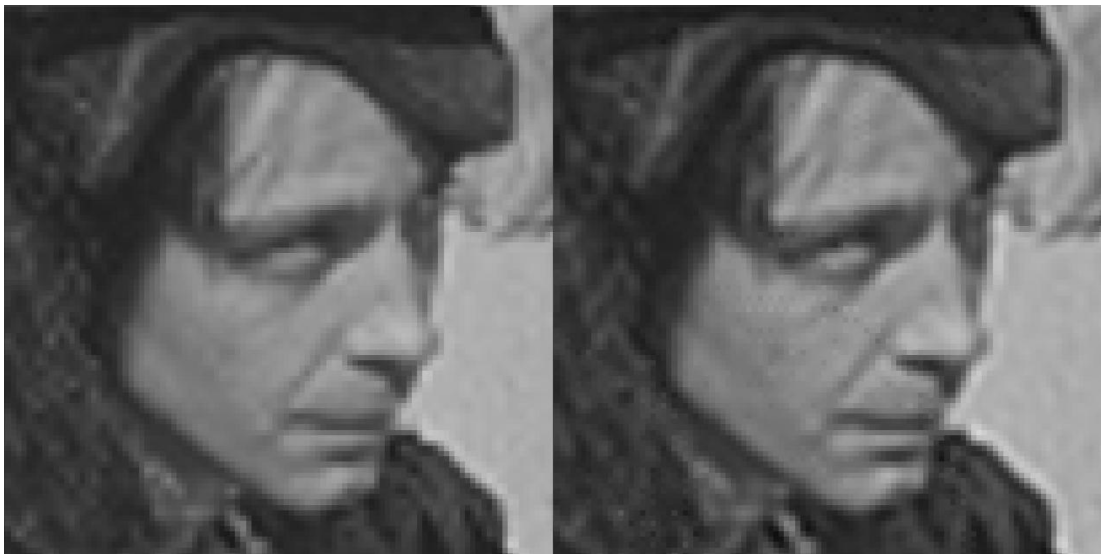


Figure- 3-5: Bicubic vs. Resulting Image of Fast Image Upsampling [8]

3.1.5 Computational Analysis

The computational analysis can be approximated by calculating FFT usage, since it is going to determine a majority of the calculations. The algorithm [8] uses 60 FFT operations. 15 FFT operations are used in a single epoch. Xilinx IP core for FFT can be used under 1 microseconds of latency for 8 bit data, therefore the total latency of the algorithm will not exceed 1us x 2 x 4 = 8 microseconds, which is enough by orders of magnitude to carry out operations for 60 fps video stream.

3.2 Video Scaling with Local Self-Examples

Self-examples are used for superresolution algorithms under the class of example based learning algorithms. Self-similarity assumption is used in SR methods to extract missing information for higher resolutions at multiple scales of the original image. In a recent paper [29] it has been proposed that small patches of images are similar to themselves for small scaling ratios under a close neighborhood. That assumption is named 'local self-similarity' as shown in Figure- 3-6. A yellow patch that is taken from the original image is almost identical to the red patch when down sampled by a small ratio.

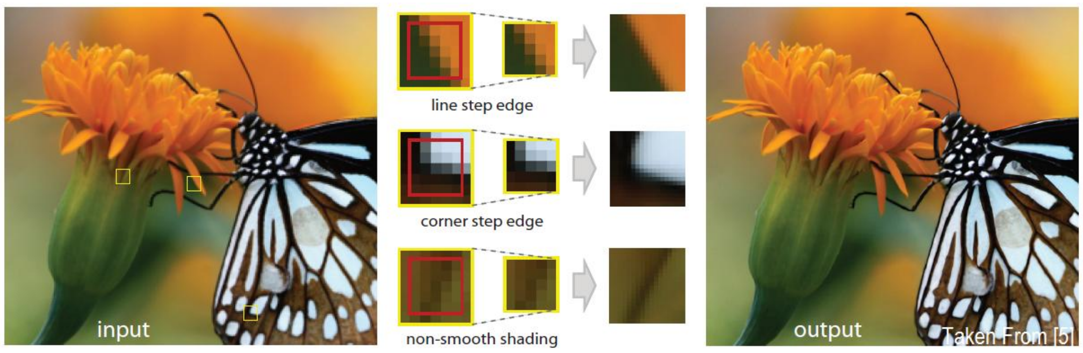


Figure- 3-6: Local Self-similarity Example

Therefore the framework of the algorithm begins by scaling an image with small ratios and continues by refining high frequencies with local patches from lower resolution image. The idea of matching a low frequency patch and making refinement by using high frequency of that patch is proposed by Freeman et al. [28] although the idea is applied to a library of patches instead of a local search. The framework is shown in Figure- 3-7.

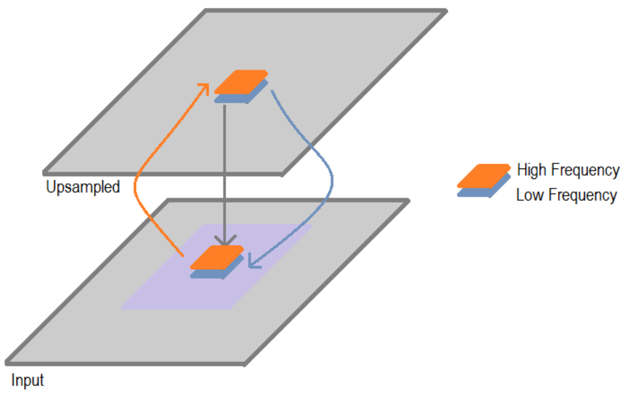


Figure- 3-7: Patch Extraction Framework

The algorithm begins with input Y_0 and upsamples it with the operator U as $u_1 = U(Y_0)$. Variable u is used for low pass component of upsampled image where v is high pass component, and X is the resulting high resolution image. This initial high resolution image lacks high frequency information. To obtain best patch matching results the original image is downsampled and then upsampled by the same filters. (Mentioned operation is done to match two images spectrally, as it will be discussed in chapter 4 this method deteriorates the result.) $u_0 = U(D(Y_0))$ is obtained. Afterwards each patch from u_1 is searched for a similar patch inside u_0 against a restricted region (purple region in Figure- 3-7). After best match is found, high frequency component of the original image that is obtained by $v_0 = Y_0 - u_0$ is added to u_1 to have $X_1(p) = u_1(p) + v_0(p)$.

3.2.1 Local Self-Similarity

Self-similarity assumption depends on preservation of edge information across scales, therefore scaling factor needs to be as small as possible for smooth upsampling. Figure- 3-8 shows an error value against increasing scaling value.

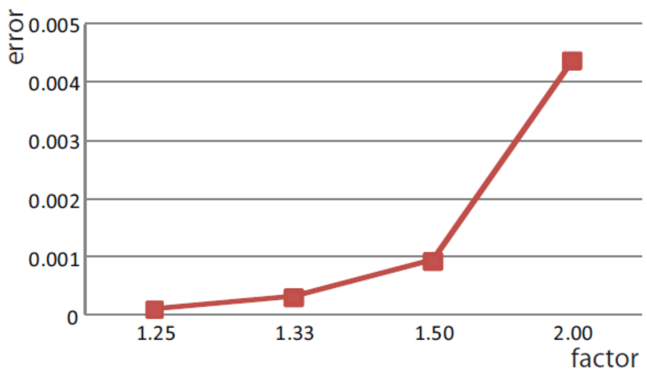
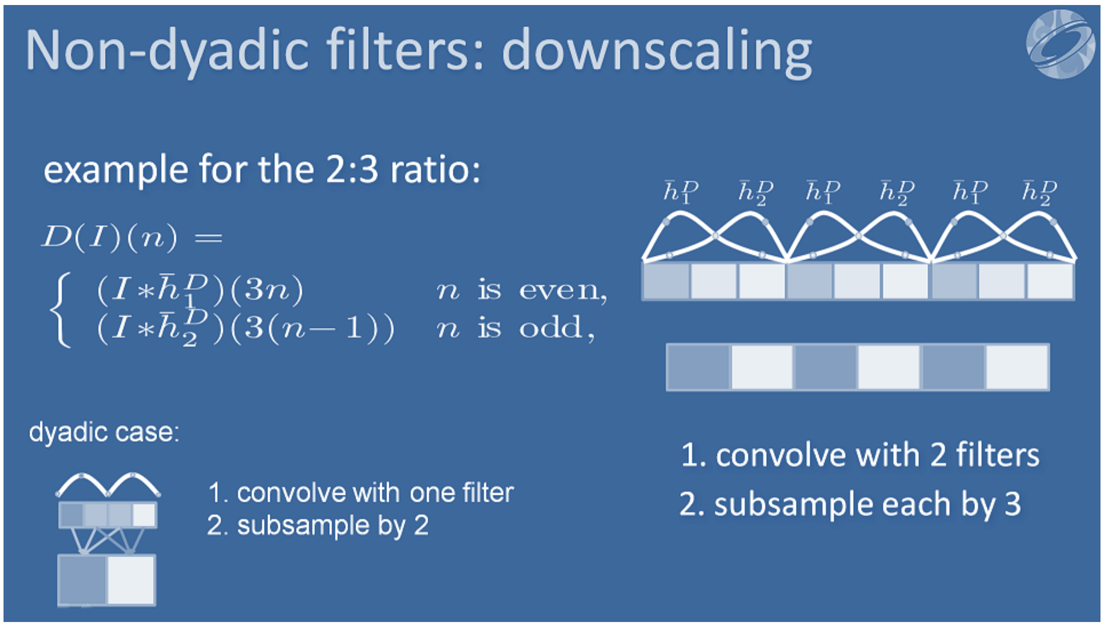


Figure- 3-8: Scaling Factor vs. Patch Error

Error decreases with scaling factor and time needed to obtain dyadic scaling increases. Therefore an optimization is required between number of upsampling operations and scaling ratios.

3.2.2 Non-Dyadic Filter Banks

As Fattal et al [29] mentioned, to create a downsample operator for N+1/N scaling factor, low resolution image needs to be filtered with N different filters and consequently subsampled by taking every N pixels out of N+1 pixels.



Non-dyadic filters: downscaling

example for the 2:3 ratio:

$$D(I)(n) = \begin{cases} (I * \bar{h}_1^D)(3n) & n \text{ is even,} \\ (I * \bar{h}_2^D)(3(n-1)) & n \text{ is odd,} \end{cases}$$

dyadic case:

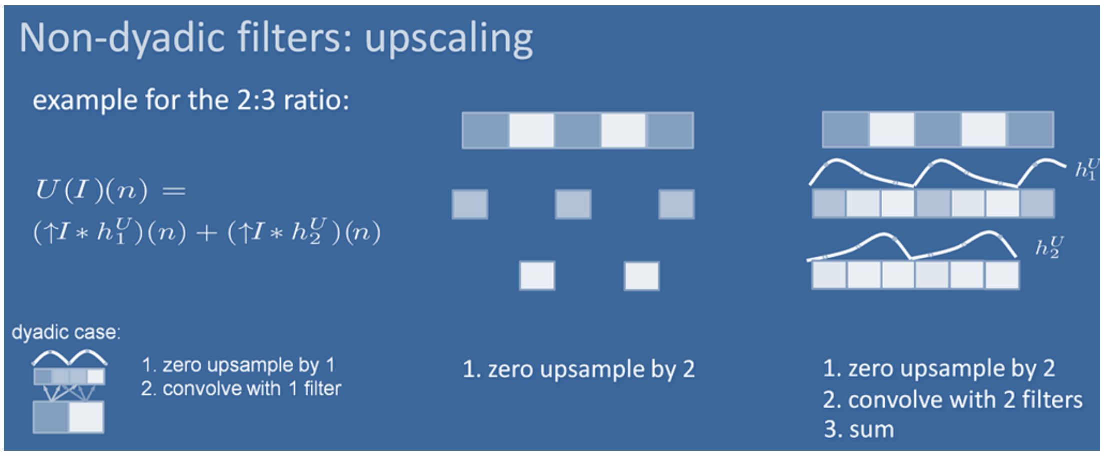
1. convolve with one filter
2. subsample by 2

1. convolve with 2 filters
2. subsample each by 3

$\bar{h}_1^D$ $\bar{h}_2^D$ $\bar{h}_1^D$ $\bar{h}_2^D$ $\bar{h}_1^D$ $\bar{h}_2^D$

Figure- 3-9 : Non-Dyadic Downsampling Operation

Similarly to perform N+1/N upsampling, the image must be separated into N zero padded images and filtered with N different filters as seen in Figure- 3-10.



Non-dyadic filters: upscaling

example for the 2:3 ratio:

$$U(I)(n) = (\uparrow I * h_1^U)(n) + (\uparrow I * h_2^U)(n)$$

dyadic case:

1. zero upsample by 1
2. convolve with 1 filter

1. zero upsample by 2

1. zero upsample by 2
2. convolve with 2 filters
3. sum

Figure- 3-10: Non-Dyadic Upsampling Operation

After using several upsample-patch cycles the resulting image is obtained.

3.2.3 Results

The result has certain defects and it is obvious that patch extraction framework had some issues. Figure- 3-11 shows the original outcome and Figure- 3-12 shows the resulting image when high frequency addition is diminished with a coefficient. It is obvious that blocky artifacts are due to a mismatch in block matching framework. The reasons are going to be analyzed on chapter 4. Since the paper had given their results in only pictures and not in numerical figures, our numerical comparisons are going to be carried out with our results. The image is used with the permission of Turgay et al. [36]

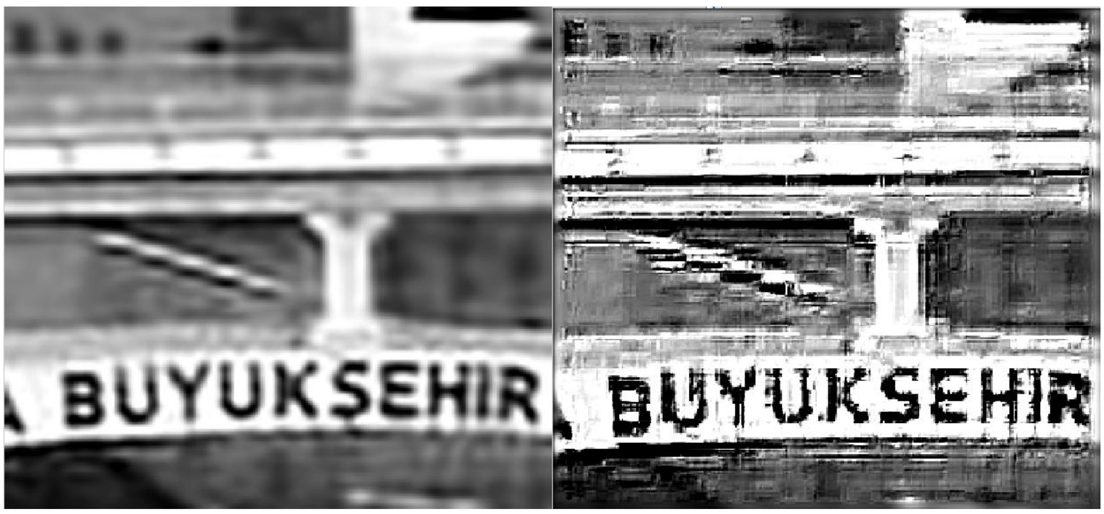


Figure- 3-11: Bicubic vs. Original Outcome of Self-Examples

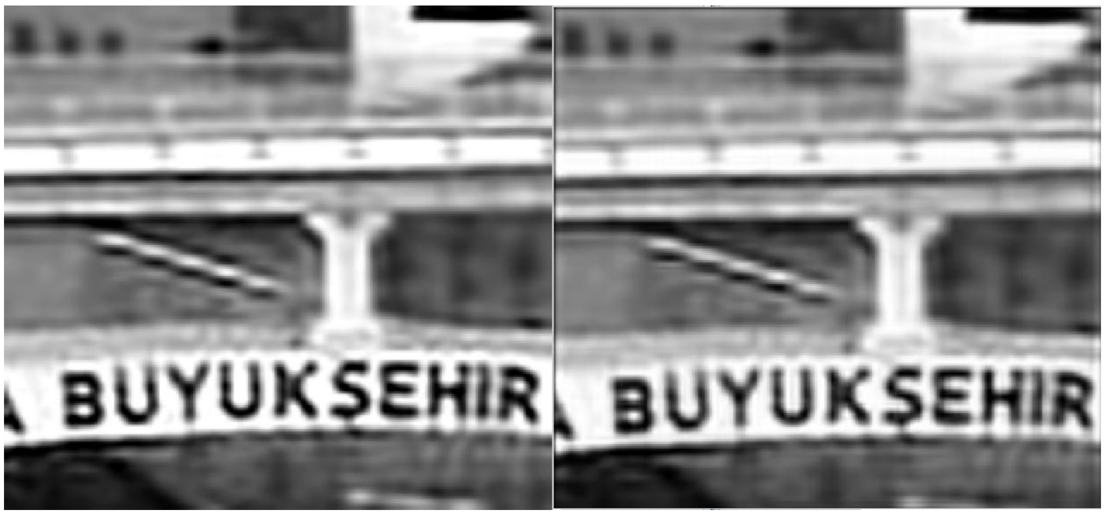


Figure- 3-12: Bicubic vs. Result of Diminished High Frequency Addition

3.2.4 Computational Analysis

Fattal et al. have implemented the algorithm on a GPU and obtained 24 fps full HD video output. Algorithm is inherently available for parallel implementation on an FPGA.

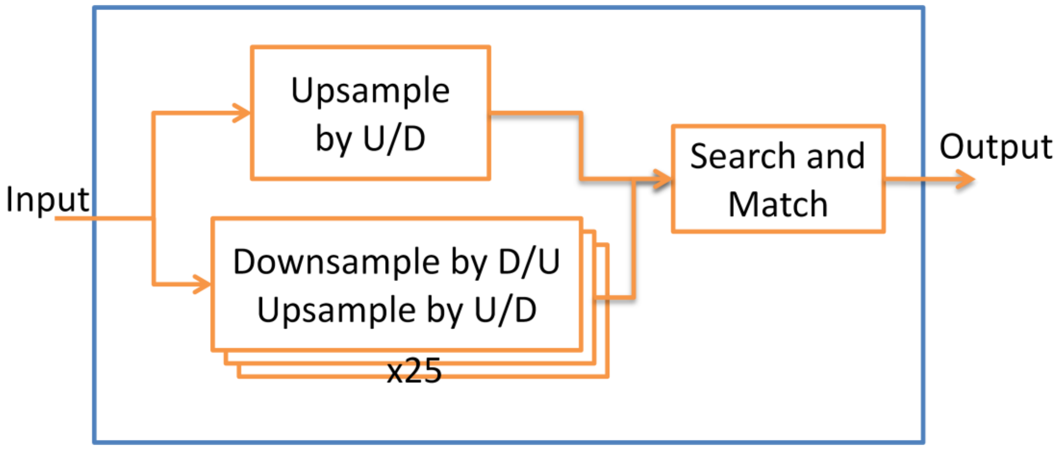


Figure- 3-13: Single Upsample Epoch

When 3x3 block size is used for block search and 5x5 neighborhood is used for local search range, the framework in Figure- 3-13 is achieved. Therefore as soon as 8 lines of an image is received, the algorithm can start with 26 parallel operations of upsampling. Output of each block can be connected to next block as in Figure- 3-14 to achieve minimal latency.

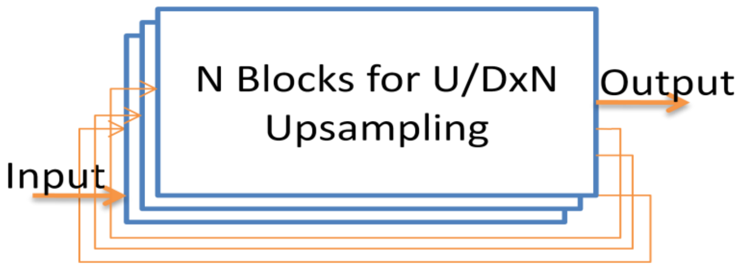


Figure- 3-14: Framework for Achieving U/DxN Upsampling

3.3 Edge Adaptive Interpolation with Bilateral Filter

Edge directed interpolation preserves edge direction by preventing edge-orthogonal filtering which causes jagginess. There are explicit and implicit methods for finding the edge orientation as discussed in Chapter 2 in detail. Siu et al [1] proposed a fast edge directed interpolation method recently. The method implicitly estimates bilateral filter coefficients to estimate edge orientation. Main algorithm is based on soft decision estimation (SAI) [37] although it has been thoroughly altered to reduce the computational time. Maximum A Posteriori estimation of range distance values for bilateral filter is proposed to incorporate horizontal, vertical and diagonal correlations into four parameters. The parameters are regularized with an adaptive regularization.

3.3.1 Bilateral Filter

Bilateral filter multiplies the pixel values with weights according to spatial and range distances between neighbor pixels and the pixel of interest. The interpolation is carried out according to the cost function [38]

$$arg_H min\left\{(H - H_{observed})^2 + \lambda \sum_{k=0}^{N} w_k (H - H_k)^2\right\} \qquad (\,3\text{-}10)$$

Where the prior is assumed to be Gaussian, H is the value of the pixel to be estimated, $H_{observed}$ is observed pixel value and H_k is the neighboring pixels. There is no observed data for superresolution problem therefore cost function is reduced to

$$arg_H min \sum_{k=0}^{N} w_k (H - H_k)^2 \qquad (\,3\text{-}11)$$

$(H - H_k)$ term is an approximation of first order derivatives used by bilateral filter. The bilateral filter [39] uses (3.12) to weight the derivatives.

$$w_k = \exp\left(-\frac{||H - H_k||_1^1}{2\sigma_1}\right) exp\left(-\frac{||z_H - z_{H_k}||_1^1}{2\sigma_2}\right) \qquad (\,3\text{-}12)$$

Where exp is exponential function and σ values represent the variances of range and spatial distances. Siu et al. [1] shows that four nearest neighboring pixels are adequate for estimating the range value. Since four nearest neighbors have the same spatial distance z_H terms can be dropped therefore yielding

$$w_k = exp\left(-\frac{|H - H_k|}{2\sigma}\right) \qquad (\,3\text{-}13)$$

After estimating the w values, those values can be substituted to (3-11) and the following closed form solution is obtained.

$$H = \frac{\sum_{k=0}^{3} w_k H_k}{\sum_{k=0}^{3} w_k} \qquad (3\text{-}14)$$

The closed form can be used as bilateral filter. However $|H - H_k|$ term requires the high resolution image to be known. Since there is no high resolution data the $|H - H_k|$ term is going to be estimated using MAP.

3.3.2 MAP Estimation of Range Distance

To estimate range distance, eight relations (operations) are used to estimate $R_k = |H - H_k|$ by exploiting geometric duality concept [4] that incorporates low resolution pixels to approximate high resolution range distances.

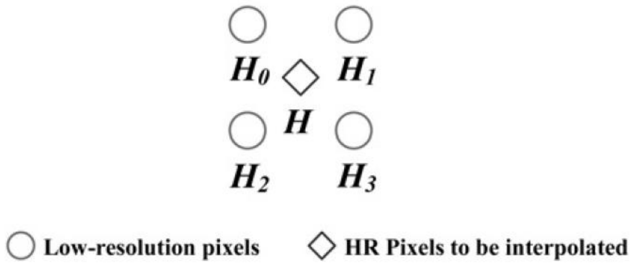


Figure- 3-15 : Locations of Neighbouring Pixels and the Pixel of Interest

The solution for estimation of range distances is given

$$R_k = \frac{\sum_{o=0,1,\ldots,7} R_k^o + \alpha R_k^{bic}}{8 + \alpha} \qquad (3\text{-}15)$$

The geometric duality concept is shown in Figure- 3-16.

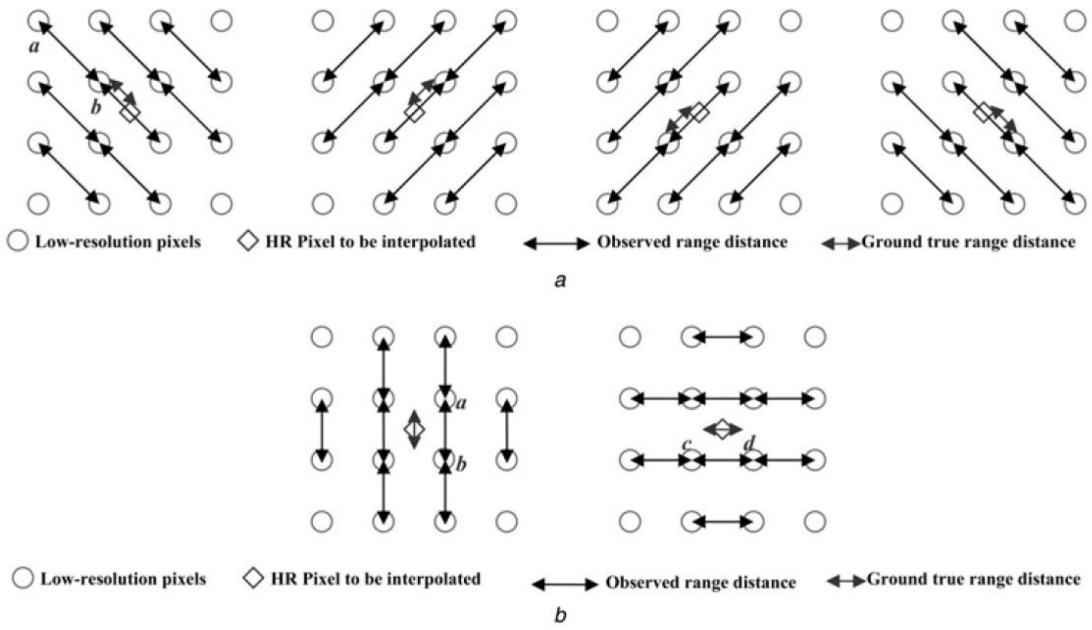

a
b
Low-resolution pixels
HR Pixel to be interpolated
Observed range distance
Ground true range distance
a
a
b
c
d
Low-resolution pixels
HR Pixel to be interpolated
Observed range distance
Ground true range distance
b

Figure- 3-16: Range Distance Approximation Using Geometric Duality

If a diagonal line exists in the low resolution image, range distances will weigh neighboring pixels according to edge orientation.

Although this approach is perfect for diagonal lines they fail in range estimation when vertical or horizontal lines appear. It has been proposed [1] to use an adaptive regularization for that problem. Extra terms Rh and Rv are calculated to detect horizontal or vertical line. Extra terms R_k^{bic} are calculated using a bicubic interpolated image. The regularization is then applied by mixing R_k^{bic} and R_k^{o} according to following equation.

$$\left\{ \sum_v R_v < \frac{1}{4.5} \sum_o R_k^o \quad OR \quad \sum_h R^h < \frac{1}{4.5} \sum_o R_k^o \right\} \ for\ k = 0,1,2,3, \ \ \alpha = 5 \tag{3-16}$$

$$\left\{\sum_{v} R^{v} < \frac{1}{12}\sum_{o} R_{k}^{o} \quad OR \quad \sum_{h} R^{h} < \frac{1}{12}\sum_{o} R_{k}^{o}\right\} for\ k = 0,1,2,3,\ \ \alpha = 8$$
$$\alpha = 0.5\ otherwise$$

3.3.3 Soft Decision Interpolation

Finding pixel values using (3-14) is called hard-decision since it does not take into account continuity of the pixel values. Recently proposed pixel-based SAI [40] provide higher accuracy than the original SAI. Pixel based SAI minimizes the cost function

$$arg_{H,\{H_{ki}\}} min \left\{ (H - \sum_{K=0}^{3} w_k H_k)^2 + \sum_{k=0}^{3} (H_k - \sum_{i=0}^{3} w_i H_{ki})^2 + \sum_{k=0}^{3} \sum_{\substack{i=0 \\ i \neq 3-k}}^{3} (H_{ki} - \sum_{j=0}^{3} w_j H_{kij})^2 \right\} \quad (3\text{-}17)$$

where H_{ki} is neighboring pixels of H_k and H_{kij} are of H_{ki}. The locations of pixels are given in Figure- 3-17. To refine (3-17) a weighting parameter U_k is introduced for least squares estimation, yielding in

$$arg_H min \left\{ (H - \sum_{k=0}^{3} w_k H_k)^2 + \sum_{k=0}^{3} U_k (H_k - \sum_{i=0}^{3} w_i H_{ki})^2 \right\} \quad (3\text{-}18)$$

$$U_{k=d(c+w_k)} \quad (3\text{-}19)$$

Constant 'd' is zero if gradient variable 'g' is less than 16 or limited to 1 if g is larger than 32 and d is equal to g/16 if g is between 16 and 32. Gradient g is calculated as

$$g = |H_0 - H_2| + |H_1 - H_3| + |H_0 - H_1| + |H_2 - H_3| \qquad (3\text{-}20)$$

If gradient is less than 16 which means the pixels have similar values, all U weights become zero yielding (3-18) to

$$arg_H min(H - \sum_{k=0}^{3} w_k H_k)^2 \qquad (3\text{-}21)$$

If gradients are higher than 16 the solution to the estimation is found as

$$H = \frac{\sum_{k=0}^{3} w_k H_k + \sum_{k=0;l=3-k}^{3} U_k w_l (H_k - \sum_{i=0;i=\neq 3-k}^{3} w_i H_{ki})}{1 + U_0 w_3^2 + U_1 w_2^2 + U_2 w_1^2 + U_3 w_0^2} \qquad (3\text{-}22)$$

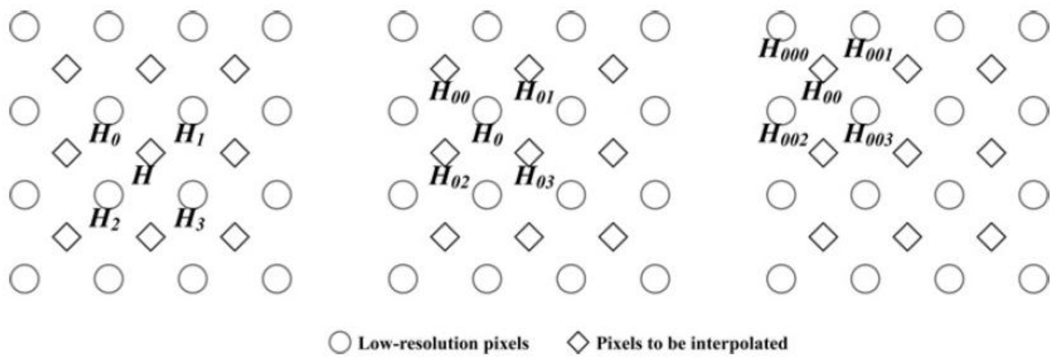


Figure- 3-17: Location of Pixels

3.3.4 Adjacent Pixels

After calculating first internal pixels, the remaining pixels are required to be calculated. Since the output image can be used as a new image, that is 45 degrees turned, same equations can be applied in follow up, as shown in Figure- 3-18. It is stated that [1] adaptive regularization is not necessary for adjacent pixels since bicubic interpolation will not provide better results.

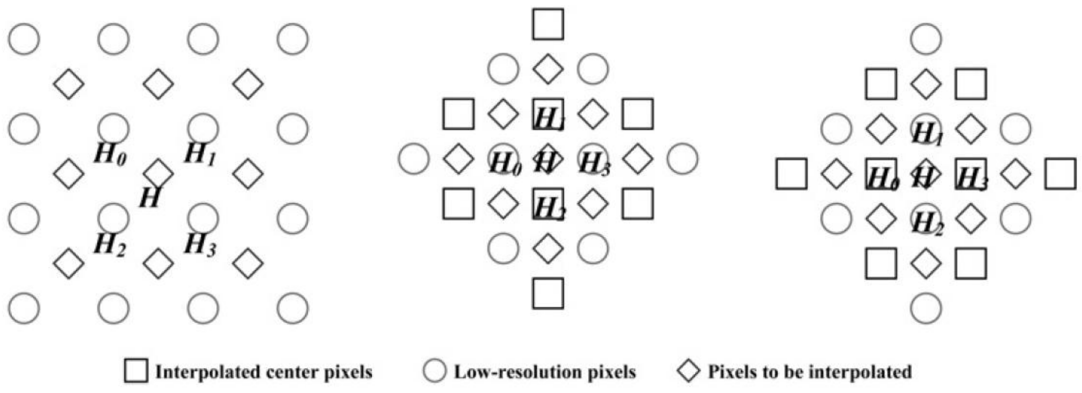


Figure- 3-18: Adjacent Pixel Locations

3.3.5 Results

The results of the algorithm are similar, if not the same, to the given figures. Resulting image is given in Figure- 3-19.

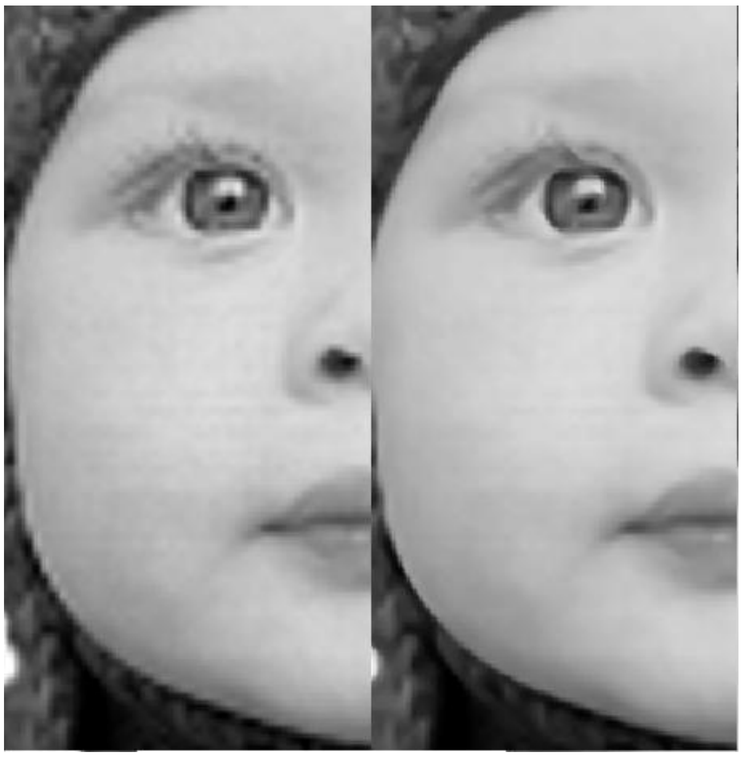


Figure- 3-19: Comparison of Bicubic vs. Edge Adaptive Interpolation

3.3.6 Computational Analysis

The analyzed algorithm does not use any complex operations such as FFT although its process is fairly straight and not suitable for parallelization. The only complex operation is exponentiation, which is easily resolved with the usage of a look up table. Details of computational analysis are going to be discussed in chapter 5.

3.4 Super Resolution Using Total Variation and Shock Filter

The Total Variation regularization is first proposed by [41]. Later TV regularization is used as a method for regularizing interpolations [24]. It has been proven to be very successful for super resolution reconstruction besides other applications. The regularization required too much computational time beside its success thus improved versions have been proposed [23].

In recent years a framework has been proposed [22]. As TV regularization is accepted to be an effective method for edge sharpening it does not improve texture component. For that matter, a separate frequency band operation is proposed. Figure- 3-20 shows the proposed [22] framework.

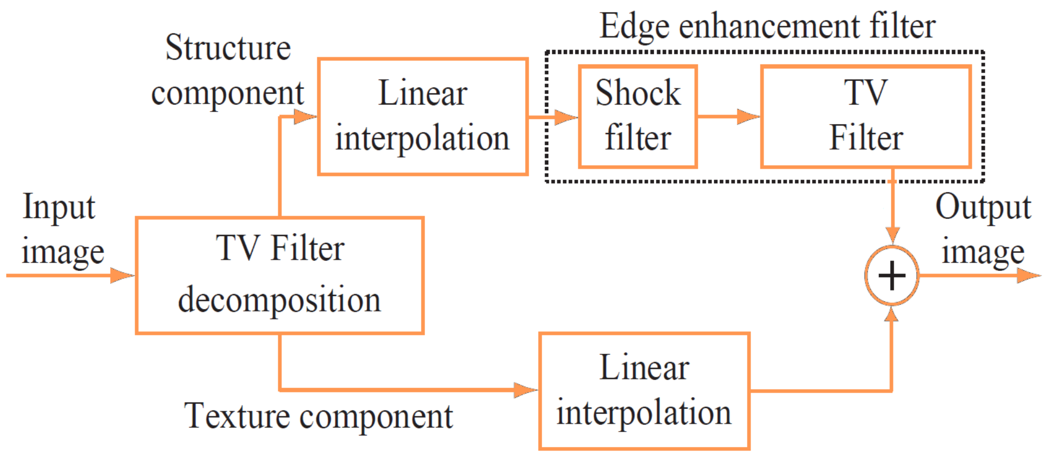


Figure- 3-20: TV Decomposition and Shock Filter

TV regularization is used to decompose the image into two bands namely structure (LF) and texture (HF). Newly introduced shock filter [42] is a non-linear filter that sharpens the edges of an image although it deteriorates the SNR of the image. Since the shock filter is used only for structure part which is low frequency band, there will be minimal degradation caused by shock filter's drawbacks.

3.4.1 Total Variation Decomposition

The TV regularization is carried out by the formula below

$$p_{i,j}^{(n+1)} = \frac{p_{i,j}^{(n)} + \tau\{\nabla(divp^{(n)} - Y/\lambda)\}_{i.j}}{1 + \tau|\nabla(divp^{(n)} - Y/\lambda)_{i,j}|} \qquad (3\text{-}23)$$

Where Y denotes the input image, p denotes a dual vector which will be used inside the iterations to propagate the updates of iterations, τ is a coefficient and div denotes divergence operation. After p converges texture component v and structure component u is obtained.

$$v = \lambda div\, p \tag{3-24}$$

$$u = Y - v \tag{3-25}$$

It is proposed [27] that TV regularization can be calculated by below formulas for interpolation.

$$H_{i,j}^{(n+1)} = H_{i,j}^{(n)} + \varepsilon \left\{ \frac{1}{2\lambda} div \frac{\nabla H_{i,j}^{(n)}}{|\nabla H_{i,j}^{(n)}|} + \sum_{k,l} |\phi_{i-k,j-l}. e_{k,l}| \right\} \tag{3-26}$$

$$e_{i,j} = \sum_{k,l=-L}^{L} \phi_{k,l}. u_{i+k,j+l}^{(n)} - Y_{i,j} \tag{3-27}$$

H is upsampled pixel value, Y is input image, ∇ is gradient operation, $\lambda\ \varepsilon$ are coefficients, n denotes iteration number and ϕ is a weight matrix of a blurring operator.

3.4.2 Shock Filter

Shock filter is calculated by using below formulas as described in [42]

$$Y^{(n+1)} = Y^{(n)} - sign(\Delta Y^{(n)})||\nabla Y^{(n)}||dt \tag{3-28}$$

$$\Delta Y = \partial_x^+ \partial_x^- Y.(\partial_x Y)^2 + \partial_y^+ \partial_y^- Y.(\partial_y Y)^2 + (\partial_x^- \partial_y^- Y + \partial_x^+ \partial_y^+ Y).\partial_x Y.\partial_y Y \tag{3-29}$$

$$||\nabla Y|| = \sqrt{(\partial_x Y)^2 + (\partial_y Y)^2} \tag{3-30}$$

$$\partial_x Y = m(\partial_x^+ Y, \partial_x^- Y) \tag{3-31}$$

$$\partial_x^\pm Y(x,y) = \pm\big(Y(x \pm 1, y) - Y(x,y)\big) \tag{3-32}$$

$$\partial_y^\pm = \pm\big(Y(x, y \pm 1) - Y(x,y)\big) \tag{3-33}$$

Y is input image, dt determines step size, function m(x,y) is defined as

$$m(x,y) = \begin{Bmatrix} sign(x).min(|x|,|y|) & (xy > 0) \\ 0 & (xy \leq 0) \end{Bmatrix} \tag{3-34}$$

3.4.3 Results

The results of the algorithm are similar, if not the same, to the given figures. Resulting image is given in Figure- 3-21.

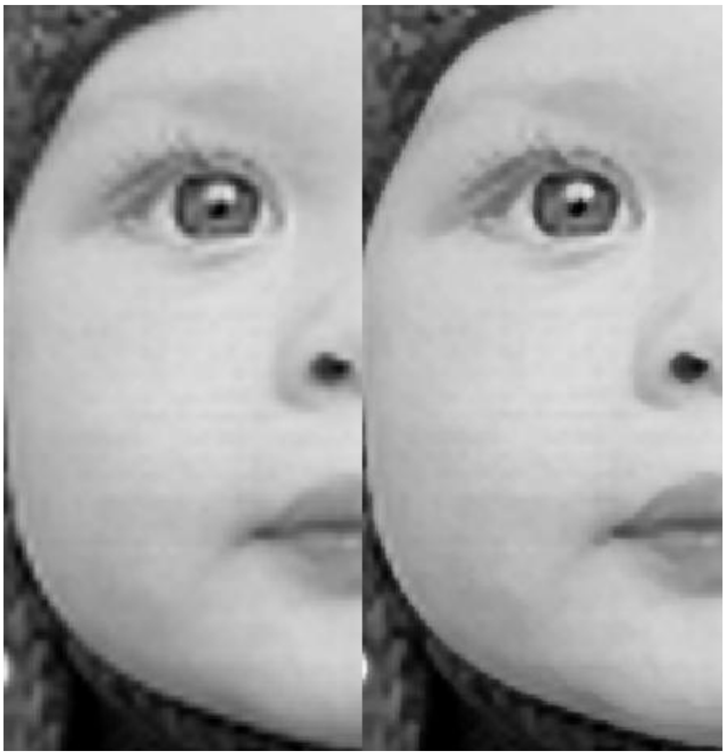


Figure- 3-21: Comparsion of Bicubic vs. SR with TV and Shock Filter

3.4.4 Computational Analysis

Resulting algorithm uses simple operations such as addition and subtraction. The iterative scheme only creates a delay of one line of an image transmission per iteration. Details of computational analysis are going to be discussed on chapter 5.

CHAPTER 4

PROPOSED ALGORITHMS

In chapter 3, the formulization and descriptions are given for various algorithms for super resolution. In this chapter potential weaknesses and proposed solutions are going to be given.

4.1 Super Resolution Using Non-Dyadic Upsampling and Non-Blind Deconvolution

As stated in chapter 3, two algorithms [8] [29], which will be merged into one, had been described. Interpolation with local self similarity assumption [29] had described a method that upsamples the image smoothly although with problems at HF extraction. Fast image/video upsampling [8] proposed a method for generating sharp images by using an iterative back-projection method although it had problems on pixel-substitution inside the framework.

In this chapter we propose an algorithm that combines these two algorithms by fixing their weaknesses and making proposed blocks to be complementary at their weaknesses.

4.1.1 Self-Similarity and Non-Dyadic Upsampling

The self-similarity assumption is a widely discussed topic in super resolution algorithms, although localized patch extraction is a unique application of this assumption. As it has been discussed in chapter 3, using non-dyadic filters for small upsampling ratios enables the localized patch extraction.

The algorithm suggests [29] upsampling the original image with ratios 5/4, 4/3 or 3/2. Then to spectrally match the upsampled image and the target image, it is suggested to do a Down-Upsampling to achieve $u_0 = U(D(Y_0))$ where Y_0 is the low resolution image. Theoretically, matching frequency content of an upsampled image to a down-upsampled image is impossible, since upsampling requires one filtering and down-upsampling requires two filtering operations. The comparison of two operations for 5/4 ratio are shown in Figure- 4-1.

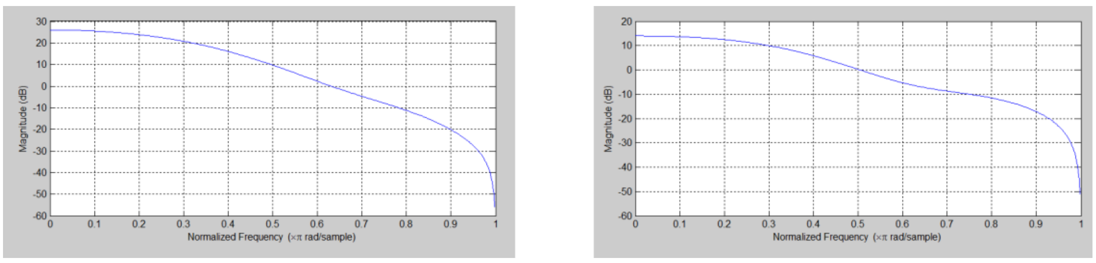


Figure- 4-1: Frequency Response of Down-Upsampling and Upsampling Operations

The observed decline in high frequency content in first graph in Figure- 4-1 threatens the optimal patch extraction therefore causes blocky artifacts.

In our tests we have discovered that, even for 5/4 ratio, the matched patches do not mate with each other. The local self-example extraction failed.

However the non-dyadic upsampling and high frequency enhancement is still a useful method to smoothly enhance an image. The small upsample rate enables the opportunity to decrease aliasing. Figure- 4-2 shows a comparison between bicubic upsampling and non-dyadic upsampling cascade + high frequency enhancement.

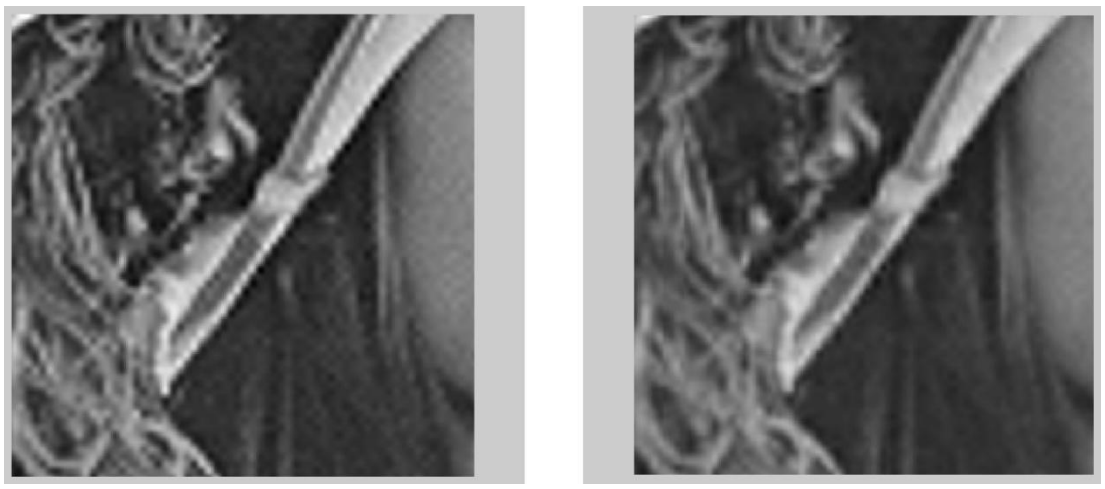


Figure- 4-2: Bicubic vs Non-Dyadic Upsampling Cascade

Notice that eventhough edges are not aliased they have lost some of their high frequency content due to soft high frequency enhancement that we have used. We have replaced patch extraction framework with another block. We have stayed loyal to the main algorithm and obtained $u_0 = U(D(Y_0))$ at each step of non-dyadic upsampling. Then we have added upsampled version of the remainder high frequency component to the upsampled image with a coefficient depending on the iteration number $X_1 = U(Y_0) + (0.1 + e^{-6n/5})U(v_0)$ where $v_0 = Y_0 - u_0$ and n is the iteration number. The coefficient is obtained empirically.

The resulting image from non-dyadic upsampling cascade is suitable for more high frequency enhancement due to its smooth structure.

4.1.2 Non-Blind Deconvolution

The first algorithm on chapter 3 had described a method for reversal of image formation. The algorithm [8] begins by upsampling the input image. Then initial upsampled image is submitted to a framework where it is deconvolved, reconvolved and pixel-substituted recursively.

Shan et al. [8] suggested that by substituting original pixels to an iterated image, the apparent information would be preserved as the image is evolved. Although it has been claimed that 4 iterations were enough to converge to a result, in our tests this has not been the case.

The pixel-substitution block did not work as it has been described. The reconvolved image had subtly smoothed version of the original values. Substituting original pixels created singularities which have been emphasized by deconvolution stage therefore the algorithm crushed.

However the non-blind deconvolution block is useful for edge enhancement. The block uses a MAP minimization solution which includes gradient distribution information from a library of images.

4.1.3 NDUID

We have called the upsampling structure "non-dyadic upsampling cascade" because it upsamples an input image 4 to 6 times to obtain a result. The outcome of the cascade lacks high frequency content therefore the iterative non-blind deconvolution from the first analyzed algorithm is used after the cascade. Since non-dyadic upsampling does not cause aliasing in an observable manner, the deconvolution block, which normally causes halos and artifacts, did well in increasing high frequency content while keeping artifacts to a minimum Figure- 4-3 shows a block diagram of the algorithm.

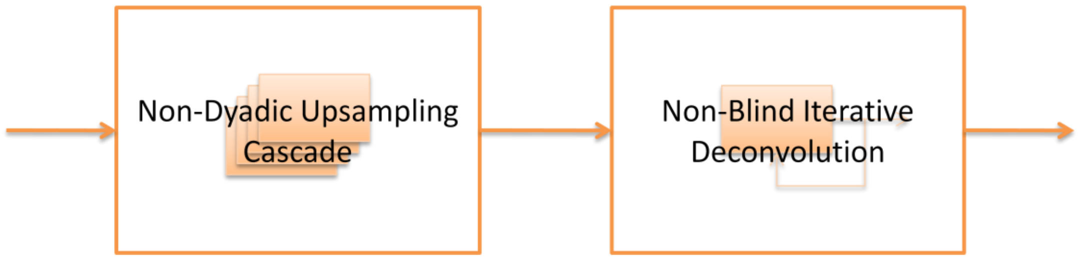

Non-Dyadic Upsampling Cascade
Non-Blind Iterative Deconvolution

Figure- 4-3 Framework of NDUID

Figure- 4-4 shows a comparison of a detail and the final comparison of bicubic vs non-dyadic upsampling cascade using non-blind iterative deconvolution (NDUID) is shown in Figure- 4-6. Also Figure- 4-5 compares to the analyzed algorithms. It can be seen that artifacts from Figure- 3-11 and Figure- 3-5 have been resolved and a sharper image than Shan et al. algorithm is obtained.

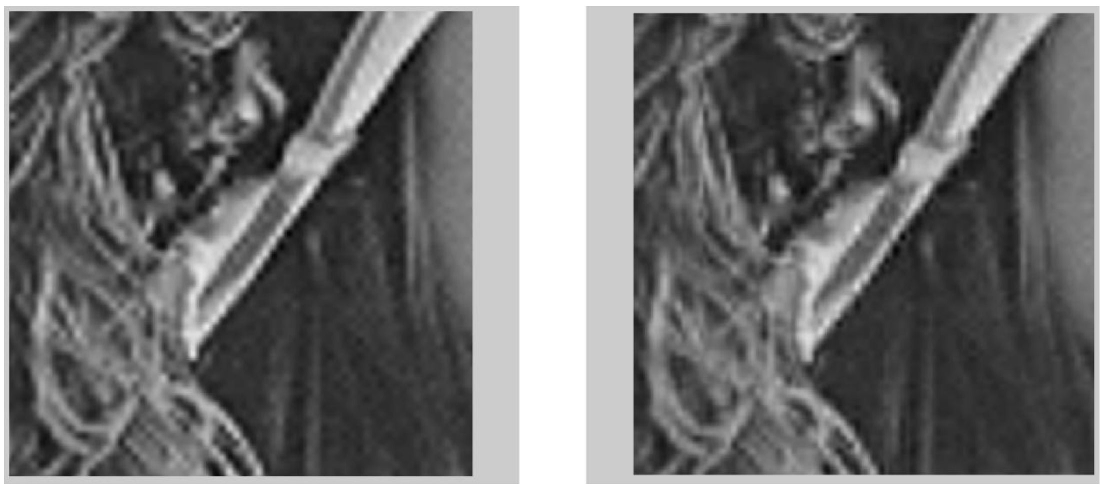


Figure- 4-4: Close-up Details of Bicubic vs NDUID

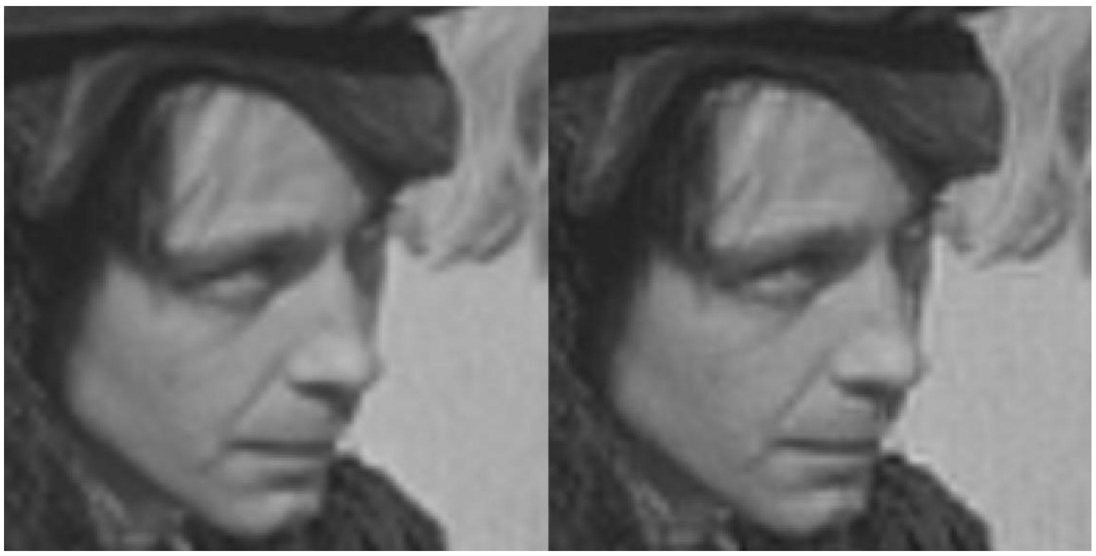


Figure- 4-5: Detail Comparison of Bicubic vs NDUID

Figure- 4-6: Comparison of Bicubic vs. NDUID

4.2 Super Resolution Using Total Variation Decomposition and Edge Adaptive Interpolation

The framework from Sakurai et al. [22] used Total Variation decomposition to separate noisy parts of an image from its structure. Structure component, then substituted to linear interpolation and filtered with a special filter called shock filter [42]. The shock filter is used to enhance the outlines of the image while protecting the resulting image from jaggy edges by simply keeping noisy texture component away from shock filter. In our tests we have discovered that the smoother and less aliased the edges are the better preserved the shock filter outcome would be. While smooth edges can be achieved by increasing the smoothing ratio of TV decomposition it in turn causes the outcome to converge to linear interpolation because it means transferring more information to texture component. Other alternative for generation of smooth edges, while preserving adequate information on structure component, is using edge adaptive interpolation. For this purpose we have proposed to use an interpolation that uses MAP minimization based on bilateral filter [1].

4.2.1 EDAT

By fusing Total Variation regularization [23] and Edge Adaptive interpolation [1] in a framework of two separate bands [22] we have achieved the framework in Figure- 4-7.

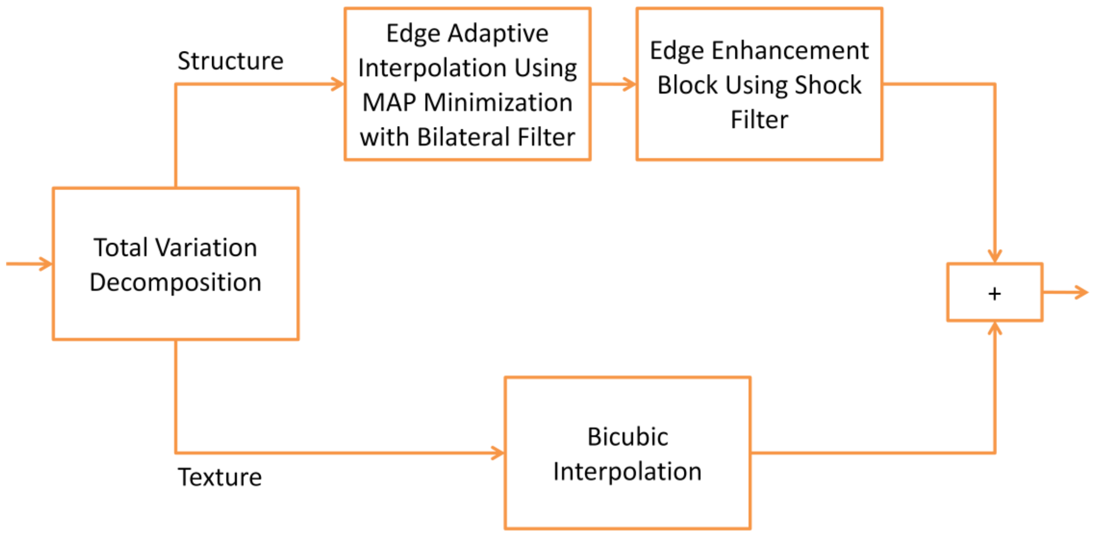

Structure
Edge Adaptive Interpolation Using MAP Minimization with Bilateral Filter
Edge Enhancement Block Using Shock Filter
Total Variation Decomposition
+
Bicubic Interpolation
Texture

Figure- 4-7: Framework of EDAT.

A comparison of bicubic interpolation and Edge Adaptive Interpolation Using TV Decomposition (EDAT) is given in Figure- 4-8. Figure- 4-9 shows the proposed EDAT does not contain aliasing effects from Figure- 3-21 and is has better high frequency reconstruction compared to Siu et al. algorithm in Figure- 3-19.

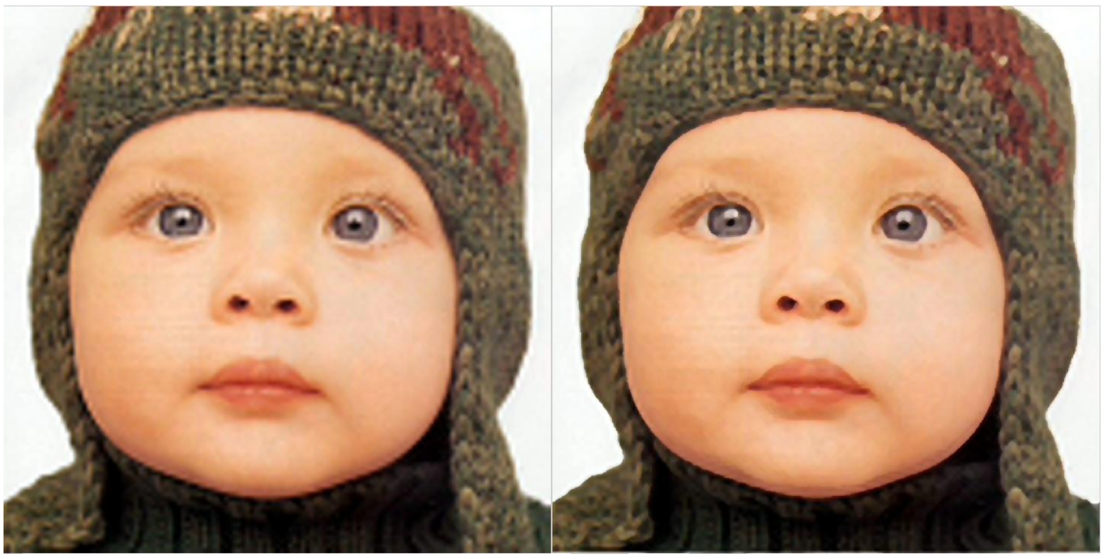


Figure- 4-8: Comparison of Bicubic (left) vs. EDAT (right)

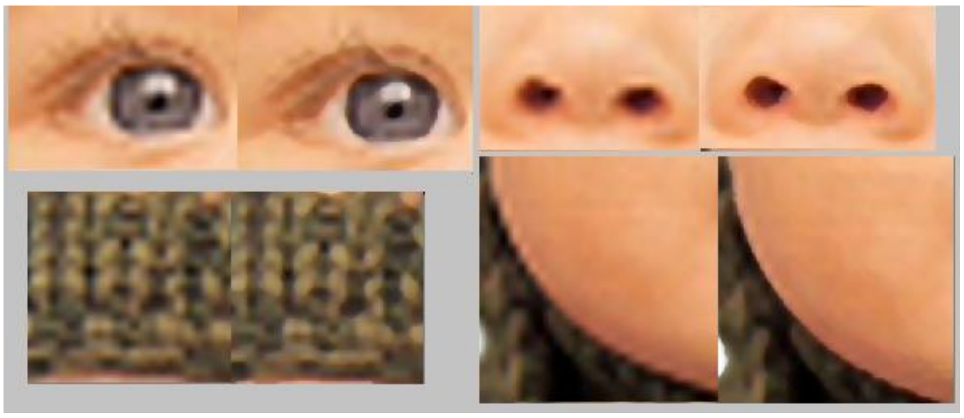


Figure- 4-9: Close-up Details of Bicubic (left) vs. EDAT (right)

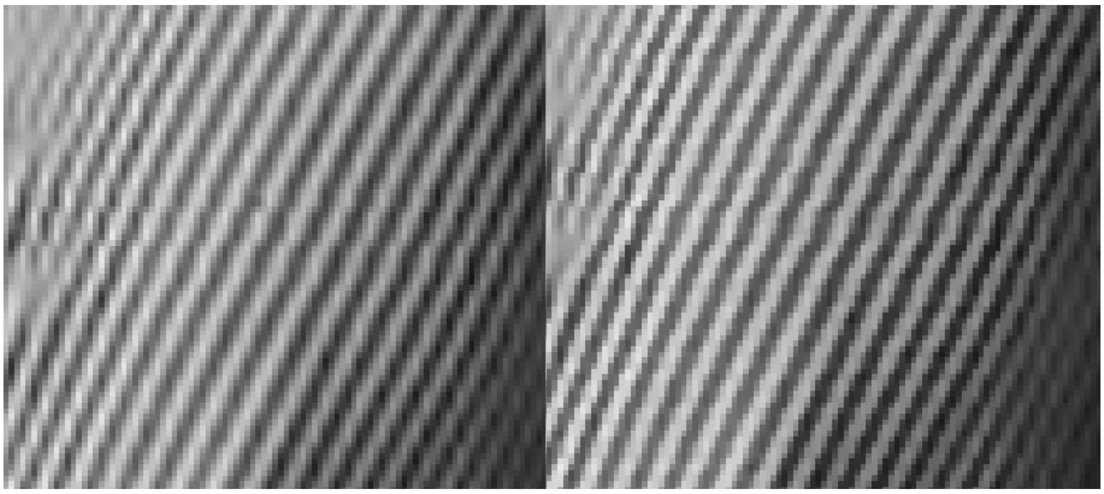


Figure- 4-10: Barbara Image Close-Up Bicubic (left) vs. EDAT (right)

There is another alternative of this method which only will be briefly discussed for further studies. In our tests we have discovered that separating the input image into more than two bands with TV decomposition block had created the opportunity to enhance different frequency contents carefully. Specifically, repeating lines in an image can be separated to different wavelet transform images in wavelet domain, which can be substituted to edge adaptive interpolation while holding the right to use enhancement block or not.

4.2.2 Edge Enhancement

The edge enhancement block is altered for our proposal. Since the edge adaptive interpolation keeps generated singularities and artifacts to a minimum, shock filter does not generate additional noise. Therefore originally proposed TV filter that is used after the shock filter is removed since it is no longer needed.

4.2.3 Edge Adaptive Interpolation

The interpolation algorithm [1] uses a method called ‘adaptive regularization’ for vertical and horizontal lines. Input image is upsampled by bicubic in parallel to the original algorithm. Depending on the angle of the line, which is calculated by range estimates from both images, the pixel value is converged to bicubic value. Therefore if a theoretical image that consists of only vertical lines is substituted to the algorithm the result will be identical to bicubic interpolation. Even though the idea is understandable it created discontinuities among other pixels causing misalignments. Therefore the adaptive regularization part is dismissed from the original algorithm which in turn increased the PSNR values on average by 0.07 dB.

4.3 Numerical Comparisons

Numerical comparisons are carried out on standard test images from Kodak image.

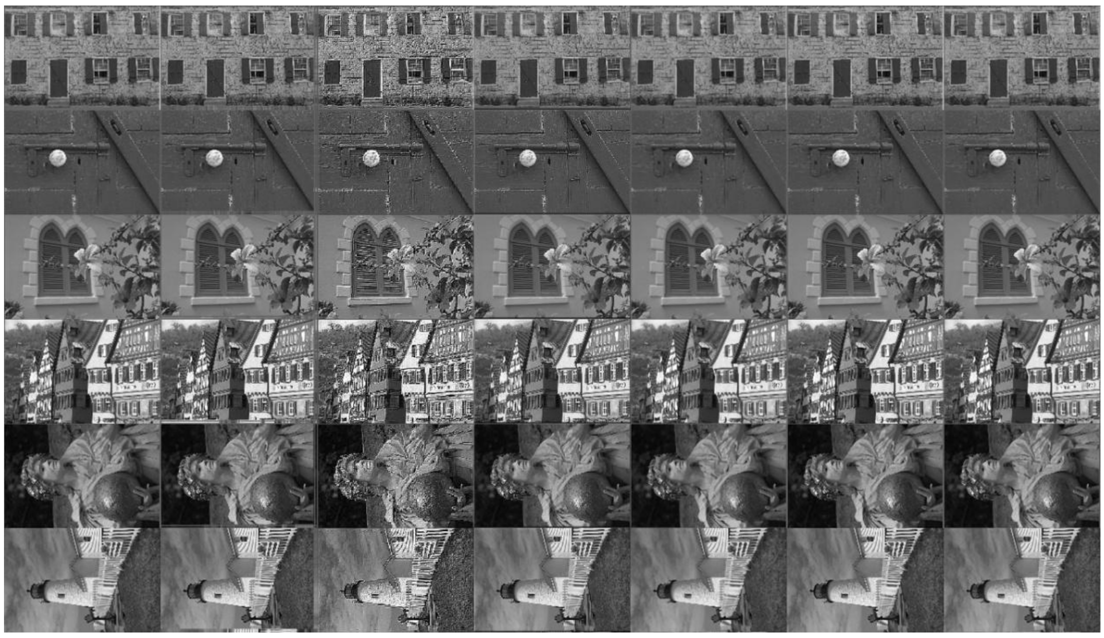


Figure- 4-11: Kodakimg Test Results (1 thru 6 from Top to Bottom)

The numerical comparisons are shown in Table- 4-1 using PSNR method. Although PSNR is widely used and accepted as a comparison method it is also known for its lack of structural measurement efficiency. Therefore we have made second calculations in

Table- 4-2 with Complex Wavelet SSIM[43] method, which takes into account pixel shift and rotation while measuring the structural similarity.

PSNR calculation is given by $PSNR = 10\log\frac{255^2}{\sum|I_1-I_2|^2/_{MxN}}$ where I is pixel value of first and second compared images, MxN is the dimension of the image.

Table- 4-1: PSNR Comparison of Theoretical Results

PSNR (x4)	Bicubic	Shan[8]	Fattal[29]	NDUID	Siu[1]	Sakurai[22]	EDAT
Kodakimg1	20,0625	19,1406	12,9687	18,8750	**20,5468**	19,8750	20,2656
Kodakimg2	27,5625	25,9062	19,3593	24,7375	**27,7968**	27,4687	27,5468
Kodakimg3	25,8593	23,9062	15,8906	22,8906	**26,3593**	25,8437	26,2031
Kodakimg4	17,7343	15,9531	11,7968	16,5781	**18,0156**	17,5312	17,8125
Kodakimg5	25,9218	22,2968	17,1250	24	**26,9062**	25,8437	26,4687
Kodakimg6	21,8437	19,2656	15,1250	19,828	**22**	21,6250	21,8281

PSNR (x2)	Bicubic	Shan[8]	Fattal[29]	NDUID	Siu[1]	Sakurai[22]	EDAT
Kodakimg1	24,5685	22,4163	16,4129	22,4636	**24,6296**	24,4708	24,5803
Kodakimg2	30,4216	28,4440	22,8773	29,2504	**30,7109**	30,4063	30,4589
Kodakimg3	31,1171	26,8100	21,4757	28,2385	**31,4414**	31,2563	31,4221
Kodakimg4	22,1925	18,6786	15,2238	19,7892	**22,2972**	22,1348	22,2952
Kodakimg5	31,3140	24,1044	21,8345	27,8108	**31,6342**	31,2762	31,5496
Kodakimg6	26,9734	21,7489	18,9365	24,0575	27,0172	26,9673	**27,0695**

The reason for Fattal et al[29] algorithm to have such low values on PSNR is mainly due to the lack of knowledge on how to mix high frequency patches with high resolution patches. The results are obtained by default 1-1 addition of HF and HR patches.

Table- 4-2: SSIM Comparison of Theoretical Results

SSIM(x2)	Bicubic	Shan[8]	Fattal[29]	NDUID	Siu[1]	Sakurai[22]	EDAT
Kodakimg1	0,8987	0,8488	0,6121	0,8355	0,8999	0,8978	**0,9000**
Kodakimg2	0,8428	0,7870	0,5217	0,7853	**0,8483**	0,8422	0,8443
Kodakimg3	0,9708	0,9404	0,7408	0,9268	0,9712	0,9712	**0,9713**
Kodakimg4	0,9065	0,8623	0,6555	0,8596	0,9066	0,9076	**0,9098**
Kodakimg5	0,9460	0,9053	0,6466	0,9017	**0,9471**	0,9458	0,9467
Kodakimg6	0,9021	0,8623	0,6572	0,8459	0,8986	0,9024	**0,9028**

SSIM(x4)	Bicubic	Shan[8]	Fattal[29]	NDUID	Siu[1]	Sakurai[22]	EDAT
Kodakimg1	0,5682	0,5382	0,2864	0,5451	0,5609	**0,5709**	**0,5709**
Kodakimg2	0,5182	0,4762	0,2253	0,4848	0,5217	0,5202	**0,5262**
Kodakimg3	0,6758	0,6464	0,3117	0,6578	0,6737	0,6869	**0,6883**
Kodakimg4	0,5833	0,5516	0,3149	0,5586	0,5821	0,5904	**0,5913**
Kodakimg5	0,6273	0,5883	0,2599	0,6013	0,6359	0,6374	**0,6425**
Kodakimg6	0,5458	0,4986	0,2549	0,5136	0,5484	0,5472	**0,5532**

Proposed algorithm EDAT exceeds other algorithms on SSIM comparison and scores well on pSNR. EDAT also performs well on visual results. Even though NDUID does not perform well compared to bicubic and EDAT, it exceeds its parent algorithms in average.

CHAPTER 5

FPGA IMPLEMENTATION

In this chapter the implementation details for the chosen algorithm is going to be given. In chapter 4, two superresolution algorithms have been proposed. NDUID used non-dyadic upsampling cascade for smoothly upsampling an image and non-blind iterative deconvolution is used to increase high frequency content. EDAT used Total Variation decomposition to separate image into two frequency bands. LF structure channel is upsampled by Edge Adaptive interpolation and edges are enhanced by shock filter. The result is obtained by summing bicubic upsampled HF texture channel with structure channel outcome.

According to our tests in chapter 4, both numerically and visually, EDAT is a better proposal. Also running times in Matlab gives an insight to the complexity of the algorithms. Although NDUID is a parallel implementable algorithm it takes 20 times longer to achieve a result than EDAT.

Therefore in this chapter proposed EDAT is going to be described in detail as to how it has been implemented in FPGA, what optimizations are made, how iterative parts are implemented.

5.1 Introduction

In today's technology most of the algorithms, if not all, are developed on PC's with software that run on processors. A processor, to put it simply, is a device that

accepts instructions sequentially and runs a specific routine to have that instruction processed.
Every command in a software code consists of multiple instructions which are handled by processor(s) that run on GHz of clock rates one by one. Although immense speeds of processors create the illusion of speed, algorithm developers know that it is not the case.

In a real-time application usage of processors are almost always dismissed because of its unpredictable running times and the limitation of one instruction per cycle.

Therefore only Field Programmable Gate Arrays (FPGA) and DSPs are solutions for a real time application.

A DSP is an operationally evolved processor that has multiple (multiples 100 most of the case) small processors that have functional blocks inside. Those functional blocks consist of adder-subtracter, FIFOs, multiplier-accumulator and fixed-floating point converters.

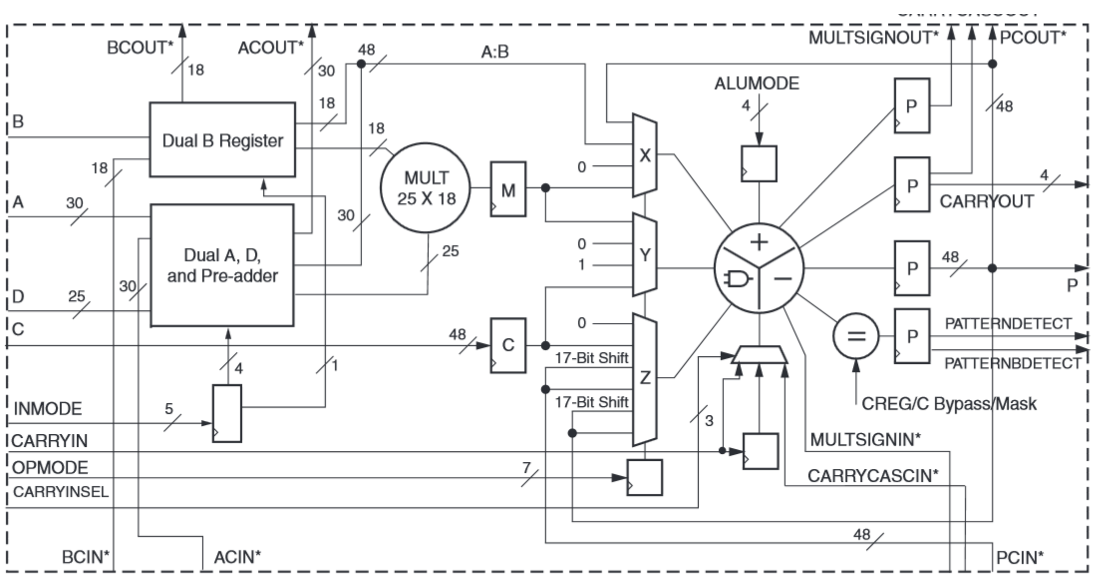


Figure- 5-1: Example DSP Design from Xilinx DSP Slice

DSP cores are powerful in floating point operations and they provide solutions for software developed designs.

FPGAs on the other hand are completely different than a DSP solution. FPGAs are programmed with Hardware Description Language (HDL) which requires a different mindset then of a software (C/C++) programmer.

In HDL, every action is defined with clock edges as in a real hardware of flip-flops. Sequential commands, commands one under other, do not execute one after other but rather they “execute” at the same time.

Actually the HDL commands do not execute but instead they are converted into hardware. Therefore one can define a hypothetical ‘and’ ‘or’ gate circuit by using HDL.

The drawback of using FPGA is that it uses fixed point arithmetic which is open to overflowing and underperforms for small number operations. There are floating point solutions for recent technology hardware, although they consume a major part of the resources and every calculation, even the simplest addition, is required to be carried on with floating point cores which will eventually deplete limited DSP slices presented in an FPGA.

The reason for FPGA being preferred for real-time applications is that every design is utterly predictable in the sense that every clock edge means a change in every state of every block that is connected to the same clock net. Therefore when a time critical algorithm is implemented on FPGA the designer is able to calculate, up to the number of clock cycles, how long will the algorithm take to complete. This is the main reason of an FPGA being preferred over a DSP.

There are implementable operations and there are hard-to implement operations on FPGA. Simple operations such as addition and subtraction can be achieved under a single clock cycle. Multiplications can be achieved under 5 clock latency and divisions can be achieved under 25 clock latency. As operations become more complex, such as exponentiation and square root, third party IP cores come into play making the precise calculation harder, resource utilization difficult and furthermore increasing the potency of the failure of implementation due to clock skew, delay, fan-out limitations etc.

As design becomes complicated the clock nets become longer; causing clock delays. When clock delay exceeds the period of the clock, further logic blocks receive different inputs then front blocks; eventually limiting the clock speed. As logic block number increases, global clock buffers reach to their design limitation in providing enough current to all the nets therefore causing clock skews which again limits the clock speed.

Although this seems as a catastrophic description, today’s technology allows a lot of space for development without limiting the user.

5.2 FPGA Board

For this thesis we have used Spartan3A DSP 3400 development board. The board has DVI video output 2 FMC connectors for DVI/VGA input daughter boards, SRAMs etc.

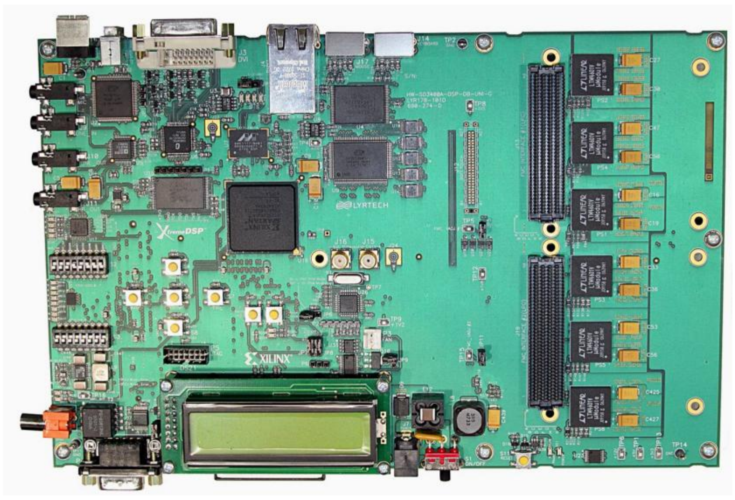


Figure- 5-2: Spartan 3A DSP 3400 Development Board.

We have used fairly simple portion of the board for our application as shown in Figure- 5-3. Even though VGA input can be used as a source of data we have chosen to devote our time to develop the algorithm and leave the acquisition and transmission part for further studies. RS232 serial communication channel is chosen for receiving input data and transmitting the output. Since internal capacity of the FPGA is limited to 1Mbits the data is stored on Cypress brand SRAM which can operate up to 166MHz clock.

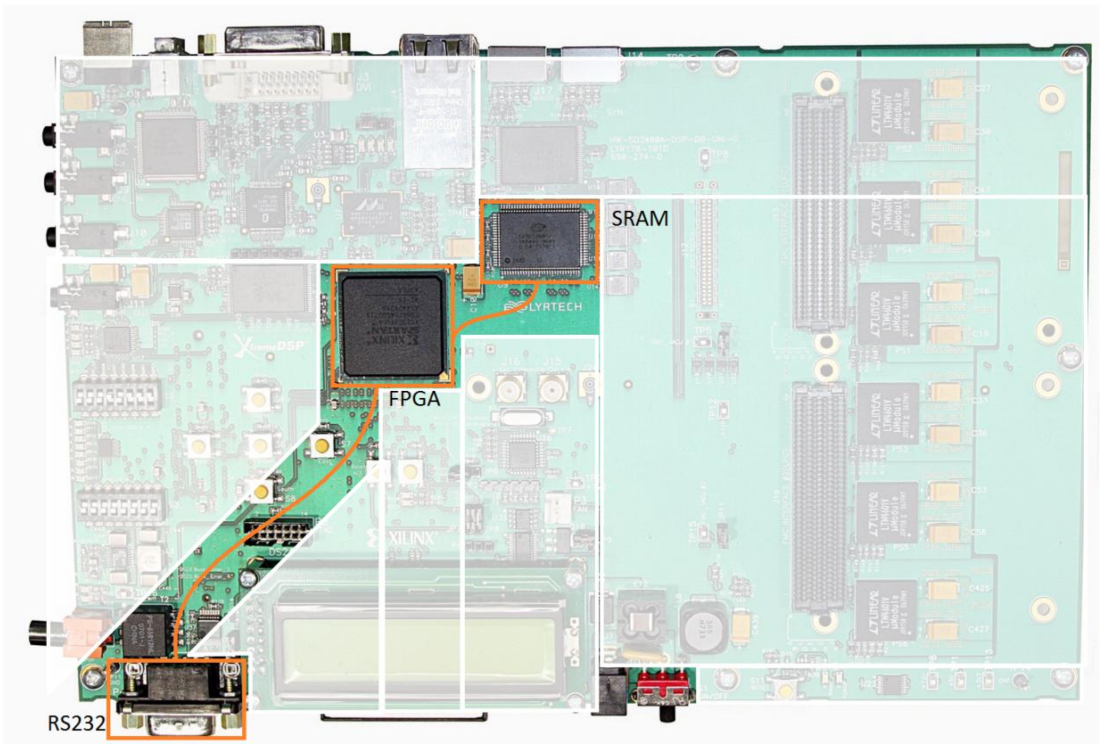


Figure- 5-3: Used Parts of the Board

5.3 FPGA Specific Optimizations

5.3.1 Using Switching Line Buffers

Spartan 3A DSP FPGA has 126 line buffers with size of 18Kbits. Since a single buffer can be reached from a single port, each line buffer is used for a single line not depending on the line’s size. This limitation has lead us to use line buffers by shifting them. In Matlab application, a single code *line1 = line2;* will do this operation. In FPGA however a buffer content cannot be shifted to another buffer in one clock therefore we have used multiplexers for line buffers as shown in Figure-5-4.

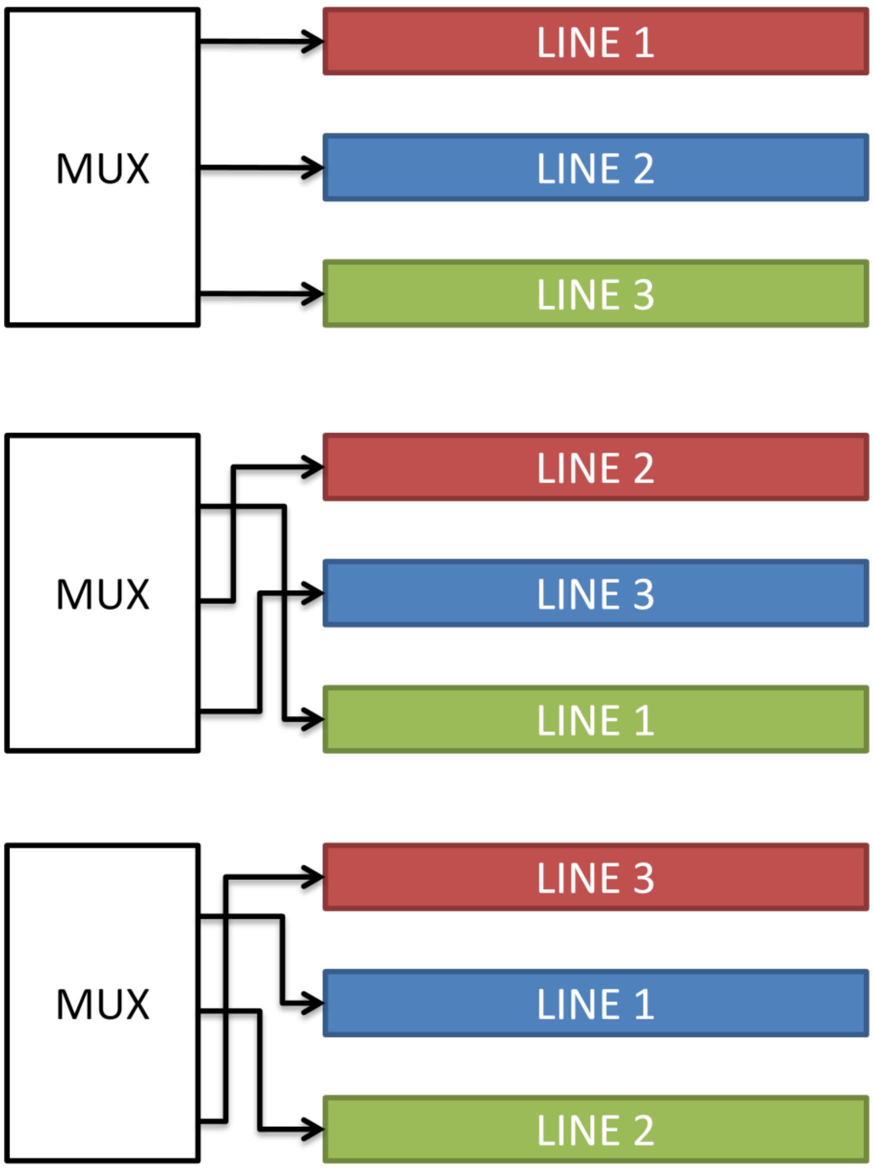


Figure- 5-4: Line Buffer Usage with Multiplexers

5.3.2 Decimal Numbers

Since floating point operations have been dismissed the decimal number problem needs to be addressed. For example the bilateral filter coefficients, for edge adaptive interpolation, are calculated by $w_k = exp\left(-\frac{R}{2\sigma}\right)$. It is obvious that the coefficient is going to be smaller than one. In fixed point environment these problems are addressed by multiplying the result by a factor of 2^n.

5.3.3 Multiplication, Division, Addition, Subtraction

Even though one does not think about using basic operations while proposing an algorithm it becomes crucial when implementing that algorithm in an FPGA. The simplest operation addition and subtraction are consequently are the easiest. The only important point about addition is that in a single operation no more than two variables can be added/subtracted. Multiplication is the second easiest operation in the sense that it consists of multiple additions. There is a block for multiplication which should be defined inside an HDL code, where the latency and resource utilization can be calculated. If a multiplication is required to be done in single clock the resource utilization will increase and potential timing errors will appear. Therefore it is wise to use a multiplication in 3-5 clock latencies to obtain optimal performance and utilization.

The division is the hardest because it is not as straightforward as addition/ multiplication. The division block uses radix-2 algorithm to iteratively calculate the quotient and remainder. Therefore it requires a minimum latency to output a result. Although the latency can reach to 25 clock edges it is not a loss in terms of a streaming data line because the block accepts input on every clock and outputs the result at every clock after the required latency.
The main optimization for division multiplication is input data size since it increases the resource usage exponentially.

5.4 Top Level Design

The design of HDL code is made hierarchically. There is always a top module that connects outer world signals inside the functional sub-modules. Figure- 5-5 shows the top module and connection of sub-modules to the outer world. Top module has three stages. First stage is input stage, RS232 pins are connected to Electronic Development Kit (EDK) core. EDK is a microprocessor created from internal resources of FPGA. There are useful blocks for functions such as RS232, I2C and Ethernet etc. Other than these blocks the processor runs with a fairly limited clock speed therefore is not suitable for further usage. Received data is transmitted from EDK to Top module to be sent to the SRAM. After the low resolution input image is obtained the Top module connects SRAM signals to sub-modules. Since internal capacity is limited FPGA reads data and writes the output on-the-run back to the SRAM. To achieve dual operation TV Filter block is assigned as master to control the connection. While TV filter is reaching to SRAM operation is put to read mode. After TV Filter reads a single line it connects the pins to Bicubic Transmission Module and Shock Filter. As operations are finalized output is written to SRAM line by line. The texture and structure component results are not combined inside the FPGA for debugging reasons.

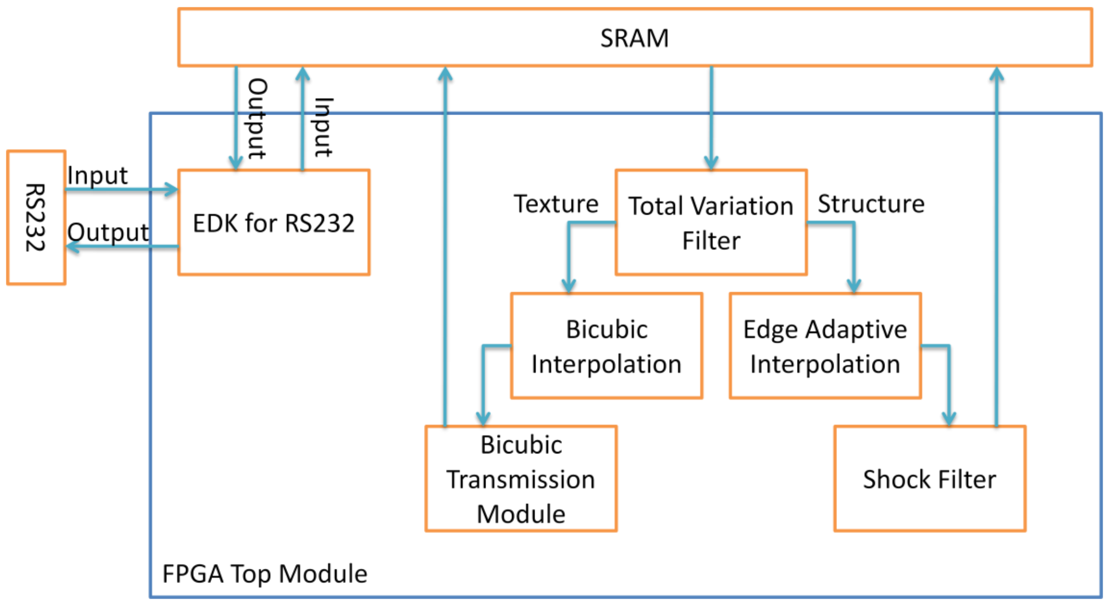


Figure- 5-5: FPGA Top Module

Upsampling means an increase in data rate, therefore for each pixel input there will be four pixel outputs. In two frequency bands case there will be 8 times more data to output. To achieve this data rate, clock rate of Edge Adaptive Interpolation, Shock Filter and Bicubic Transmission Module is quadrupled. The synchronization of two output modules is achieved by design. Edge Adaptive Interpolation calculations together with shock filter operations takes long enough for bicubic results to be transmitted to SRAM.

To sum up the control flow is as follows:

```
*Top module captures all lines into SRAM*
    *Top Module changes the mode from input stage to algorithm stage*
    *TV Filter captures one line of N pixels with clock rate of C*
        *TV Filter changes mode of operation from read to write*
    *Bicubic Transmission Module outputs 2N pixels with clock rate of 4C*
    *Shock Filter Outputs 2N pixels with clock rate of 4C returns ack. to TV Filter*
        *TV Filter changes mode from write to read*
    *If all lines are read return ack. to Top module for output stage*
*If Algorithm is finished Top Module changes the mode to Output Mode.*
    *Top module transmits all data from SRAM through RS232.*
```

Figure- 5-6 shows a result from simulations. CE is an abbreviation for clock enable. When CE_TV is low, TV filter is reading from SRAM and as soon as TV filter leaves the control by pulling CE_TV to high, Bicubic Transmission Module starts sending results to SRAM. After Edge Adaptive Interpolation and Shock Filter finishes operations, shock filter block start to send the results, thankful to the precise calculation to ensure that bicubic is finished before shock filter block starts.

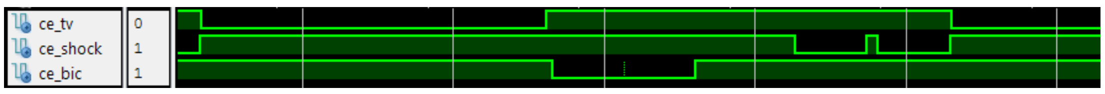


Figure- 5-6: Write/Read Cycle

5.5 Total Variation Filter

In equation (3-23) TV filter operation is summarized.

$$p_{i,j}^{(n+1)} = \frac{p_{i,j}^{(n)} + \tau\left\{\nabla(divp^{(n)} - Y/\lambda)\right\}_{i.j}}{1 + \tau|\nabla(divp^{(n)} - Y/\lambda)_{i,j}|} \tag{5-1}$$

A dual vector p, for x and y dimensions, is iterated over itself to reach a solution. Gradient operation, for FPGA, can be broken down into two difference operations. λ is taken 1 for simplicity. $u = \tau.divp^{(n)} - Y$ is defined, for gradient result $dux = u_{(x+1)} - u_{(x)}$ and $duy = u_{(y+1)} - u_{(y)}$ is defined. $divp^{(n)} = p_{(x+1)} - p_{(x)} + p_{(y+1)} - p_{(y)}$.

Then for the denominator L2 norm $d = 1 + \sqrt{dux^2 + duy^2}$ is defined. Although a square root operation is difficult to implement in FPGA, square root of sum of squares is a distance and it can be approximated according to relative distance of two variables. If dux is larger than duy by an order of magnitude, "d" can be approximated as d = 1 + dux. L2 Norm approximation is done by piecewise continuous function:

$$\|a, b\|_2 = \begin{cases} a + \frac{b}{8}, & a > 8b \\ a + \frac{b}{2}, & 8b > a > b \\ b + \frac{a}{2}, & 8a > b > a \\ b + \frac{a}{8}, & b > 8a \\ \frac{3a}{2}, & a = b \end{cases} \tag{5-2}$$

In literature there are few examples of L2 norm approximations which are generally designed for processor based codes or floating point operations. The approximation that we have proposed is an easier-to-implement method whose artifacts are negligible due to its usage for summation normalization inside the algorithms.

The result can be discretized depending on relative magnitudes of two variables. The dual vector p is also split into px and py.

In our tests we have found that 4 iterations are enough for convergence. In a software environment iterations are done on whole image. Since in Spartan 3A DSP FPGA there are 126 line buffers available, the iterative scheme is reduced to line by line scheme.
The overall algorithm is shown in the pseudo code and in Figure- 5-7.

```
Do for all lines
    Capture 1st line.
    As capturing 2nd line
        Calculate dux1, duy1, d, px1, py1, divp1
        Update 1st line
    As capturing 3rd line
        Calculate dux2, duy2, d, px2, py2, divp2
        Update 2nd line
        As updating 2nd line calculate dux1, duy1, d, px1, py1, divp1
        Update 1st line
    Carry on until fifth line
    Output first line
End Do;
```

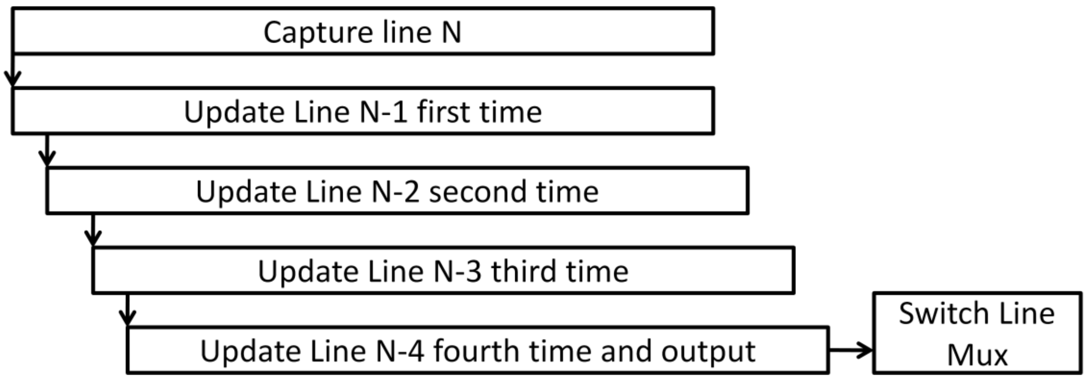
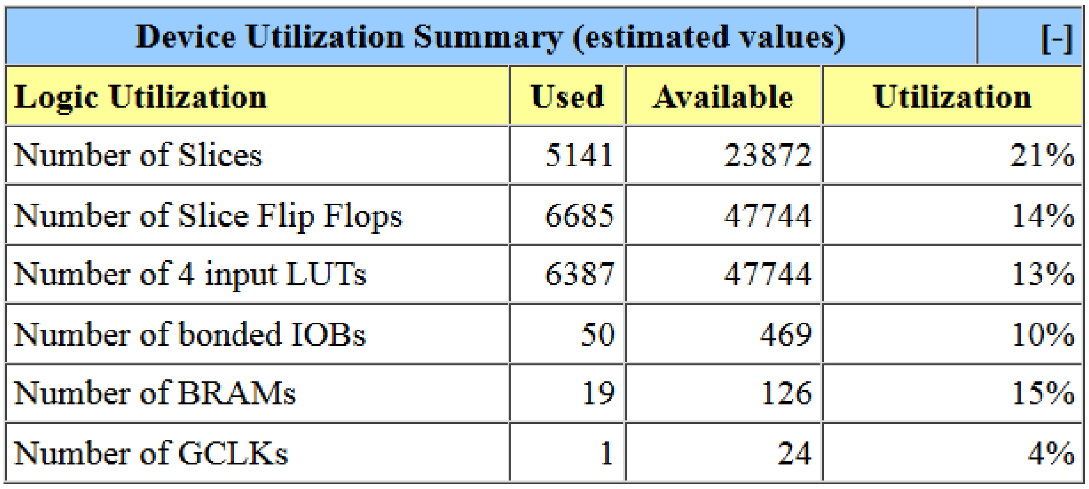

Capture line N
Update Line N-1 first time
Update Line N-2 second time
Update Line N-3 third time
Update Line N-4 fourth time and output
Switch Line Mux

Figure- 5-7: Total Variation Filter Line Based Iterative Scheme

The resource requirements are listed below:

1. 8 Division modules for dux1,2,3,4 and duy1,2,3,4 calculations
2. 10 line buffer for 5 lines and 5 updated lines
3. 8 line buffers for px1,2,3,4 and py1,2,3,4
4. Other math operations and pipeline delay vectors.

The final code is synthesized and the resource utilization is found as in Figure- 5-8. The utilization is in estimated state since only final TOP level module is implemented. The estimated values reflect non-optimized values for place and route results therefore they can be accepted as a maximum value for utilization.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	5141	23872	21%
Number of Slice Flip Flops	6685	47744	14%
Number of 4 input LUTs	6387	47744	13%
Number of bonded IOBs	50	469	10%
Number of BRAMs	19	126	15%
Number of GCLKs	1	24	4%

Figure- 5-8: TV Filter Resource Utilization

5.6 Edge Adaptive Interpolation

This is the largest module to be designed because of its operational load. The algorithm starts by calculating the first internal pixels that will be placed on interpolated lines. To calculate a new pixel, by using bilateral filter minimization problem, range distance estimates need to be calculated. Range estimates are calculated using (3-15). Therefore 4 pixels from 4 lines are required. Once range distance estimates are calculated, bilateral filter coefficients are obtained using (3-13). The equation contains exponentiation which requires floating point operations strictly, but since there are 0 to 255 possible values for range distance, the value of filter coefficients can be obtained from a look up table that is filled with known results for each range value. As mentioned previously, these coefficients are decimal numbers the look up table is filled with numbers that are multiples of 2^7.

There is an optimization made for the multiplication value. If the multiplication value is chosen to be quite large it will allow all possible coefficient value to be almost precisely represented with integers. However those values are used for multiplication, as in (3-14), therefore choosing larger values means creating a multiplication that requires operations with more bits which in return increases the complexity and resource utilization. 2^7 was on a sweet spot in between being too large and cleaving nearly all coefficients. Before going into real application, all of these optimizations are reflected to Matlab codes where all variables were integers and gave insights on how discretization affected our result.

Another optimization was made about cascaded multiplications. If values with range 2^8 are multiplied over a cascade of four, the resulting value will have a range of 2^{64}. The resource utilization is going to be 24 times more than that of a single stage multiplication. Therefore, by confirming suitability on Matlab, cascaded multiplication results are diminished by an order of 2^8 every time they are substituted to another stage of multiplication.

After first internal pixels are calculated the newly found values are needed to be corrected according to pixel continuity. For that matter U_k values are calculated (3-19) and continuity criterion minimization is applied according to the solution (3-22).

The basic grid of initial pixels is obtained as in Figure- 5-9. Next step will be to calculate neighbor pixels using the same method for 45 degrees rotated grid. The pixel locations will be in different line buffers compared to initial pixels but the operations required to calculate new pixels are the same, once 16 pixel values are obtained.

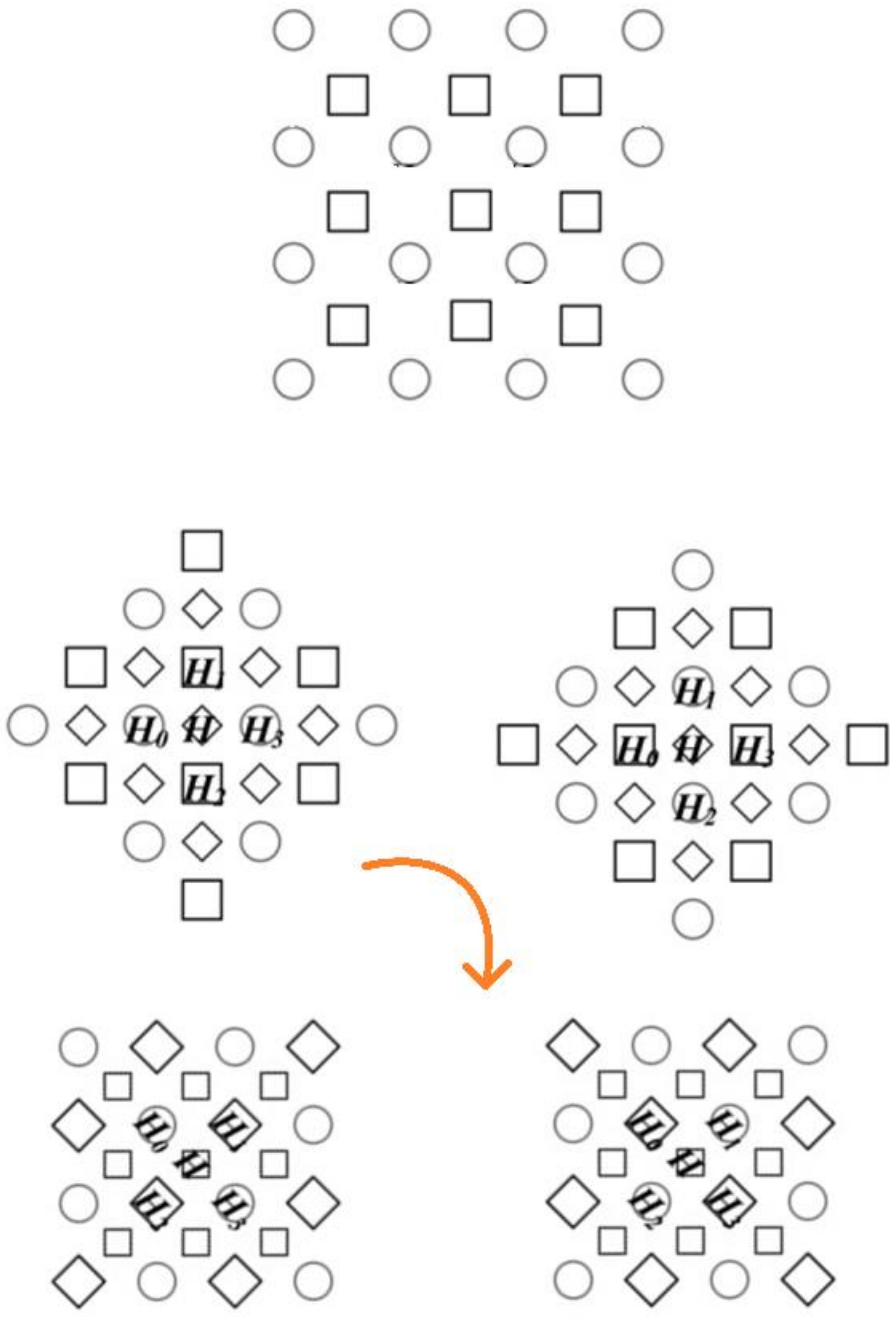


Figure- 5-9: Initial and Secondary Neighbour Pixel Locations

The ability to use the same operational basis for three different pixel calculations lead us to another optimization. As we have discussed previously, the clock rate for

Edge Adaptive Interpolation is quadruple of TV Filter. This enables the opportunity to do some calculations sequentially.

A detailed resource analysis set forth that to make all the calculations, 108 multiplications and 6 divisions are required. That is a huge number that will almost occupy 35% of FPGA with just the operations. By using the speed allowance, operations for initial pixels, secondary interpolated line pixels and secondary main line pixels are calculated and updated sequentially by using the same resources. This reduced multiplications to 36 and divisions to 1, saving 20% space.

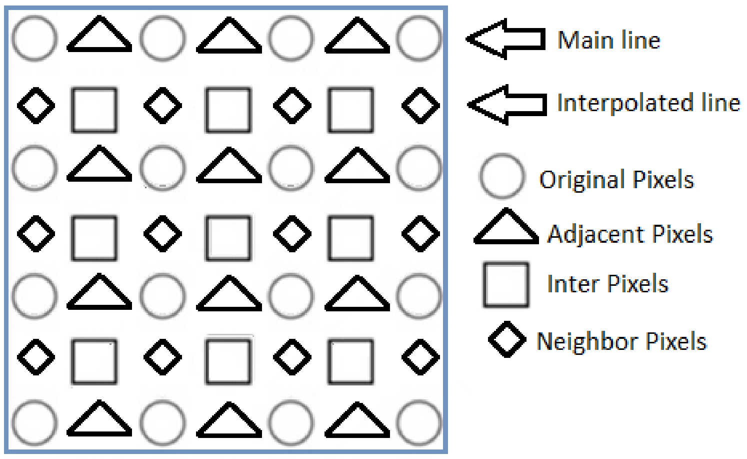


Figure- 5-10: Pixel and Line Naming

The overall algorithm is shown in pseudo code below, naming is made according to Figure- 5-10:

```
Do for all lines
Capture Nth line
        Update N-2nd  inter pixels
        Calculate N-3rd  neighbor pixels
        Calculate N-4th  adjacent pixels
        Update N-5th  neighbor pixels
        Update N-5th adjacent pixels
Output N-7th  main line
Output N-7th  interpolated line
SWITCH LINES
```

Pixel calculation and updating operations are common except the input pixels which will be handled with another process. Figure- 5-11 show operations of calculation and Figure- 5-12 show operations of updating. Relative delays between variables and order decimations between cascaded multiplications are not shown for simplicity. Pixel values and coefficient values are shown with letters and as w_{0-3} respectively. The coefficient values are calculated in pixel calculation step and stored inside a line buffer to be used in pixel update step.

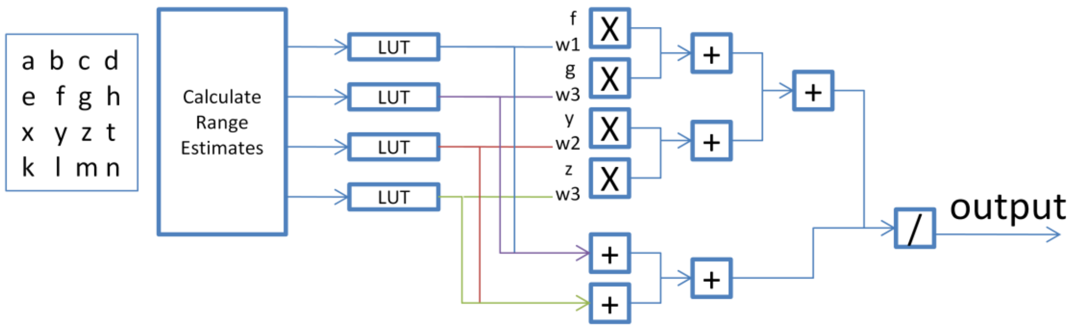


Figure- 5-11: Pixel Calculation Operations

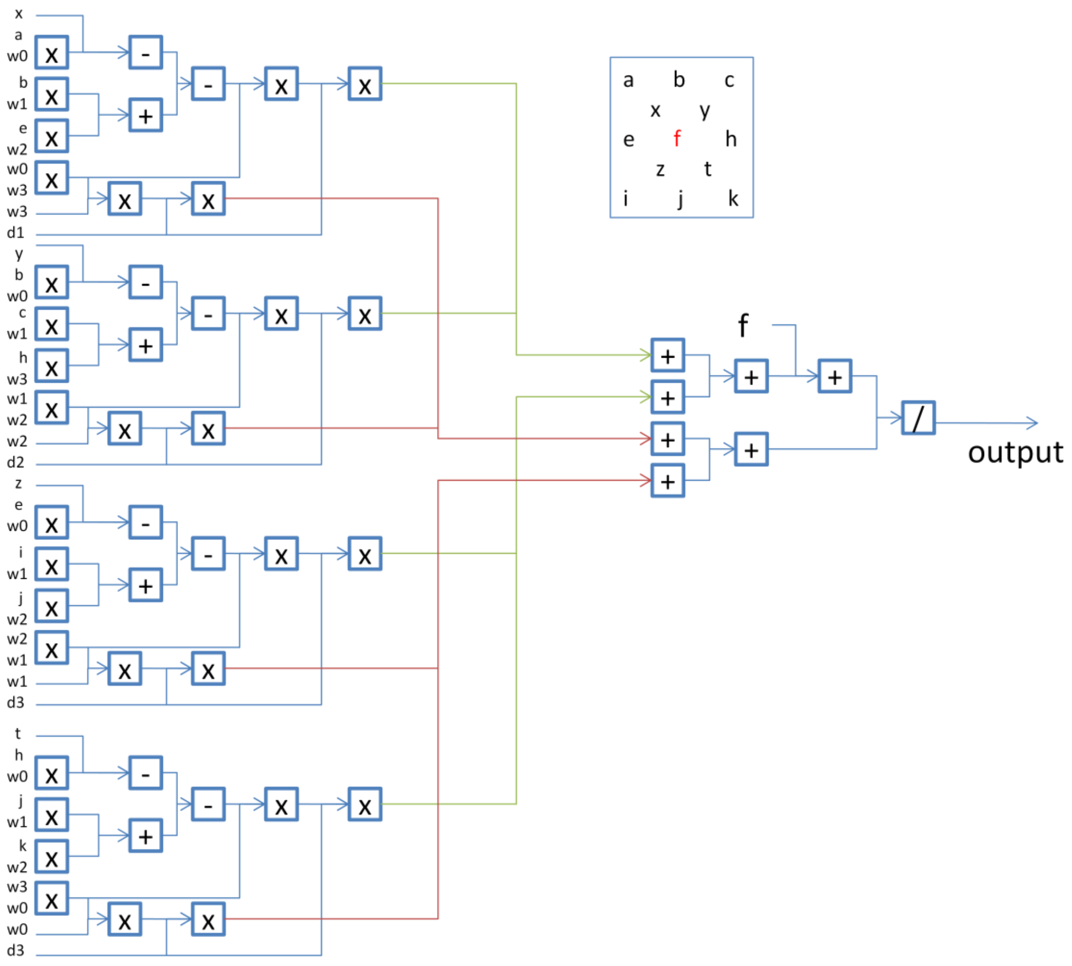
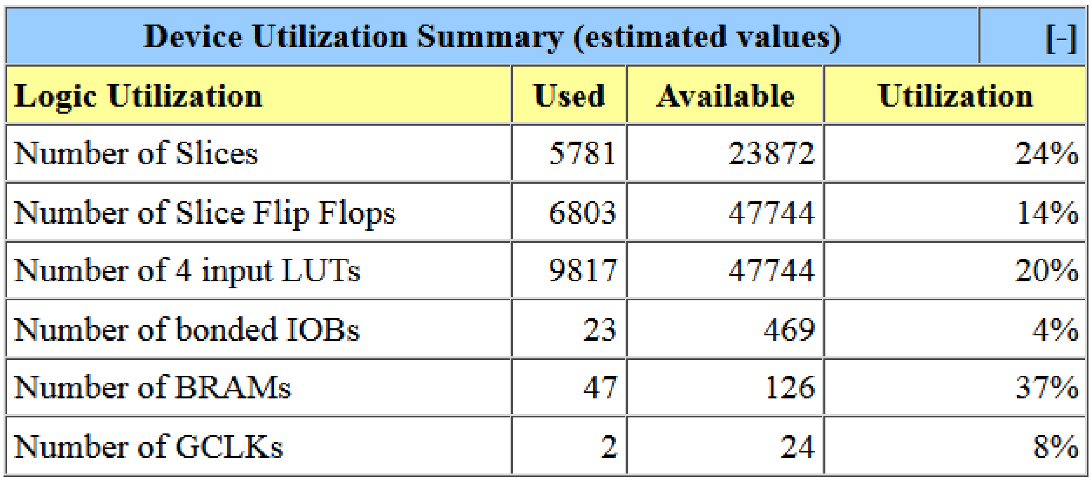


Figure- 5-12: Pixel Updating Operations

Figure- 5-13 show the estimated resource utilization for edge adaptive interpolation block only.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	5781	23872	24%
Number of Slice Flip Flops	6803	47744	14%
Number of 4 input LUTs	9817	47744	20%
Number of bonded IOBs	23	469	4%
Number of BRAMs	47	126	37%
Number of GCLKs	2	24	8%

Figure- 5-13: Edge Adaptive Interpolation Resource Utilization

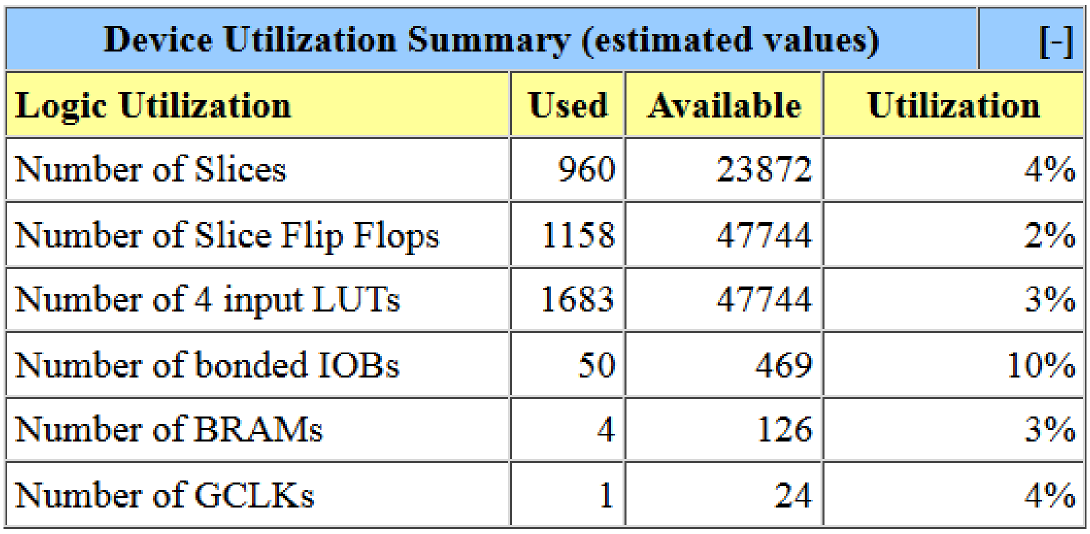
5.7 Shock Filter and Bicubic Interpolation

Shock filter is an iterative algorithm. However the step parameter is used to slowly converge the output. In our tests we have discovered that using single iteration with larger step variable yields the same result for smoothed images such as our case. Therefore shock filter is optimized to work as a batch algorithm.

Bicubic interpolation filters four pixel values to obtain high resolution pixel value both for in-line pixels and intermediate lines.

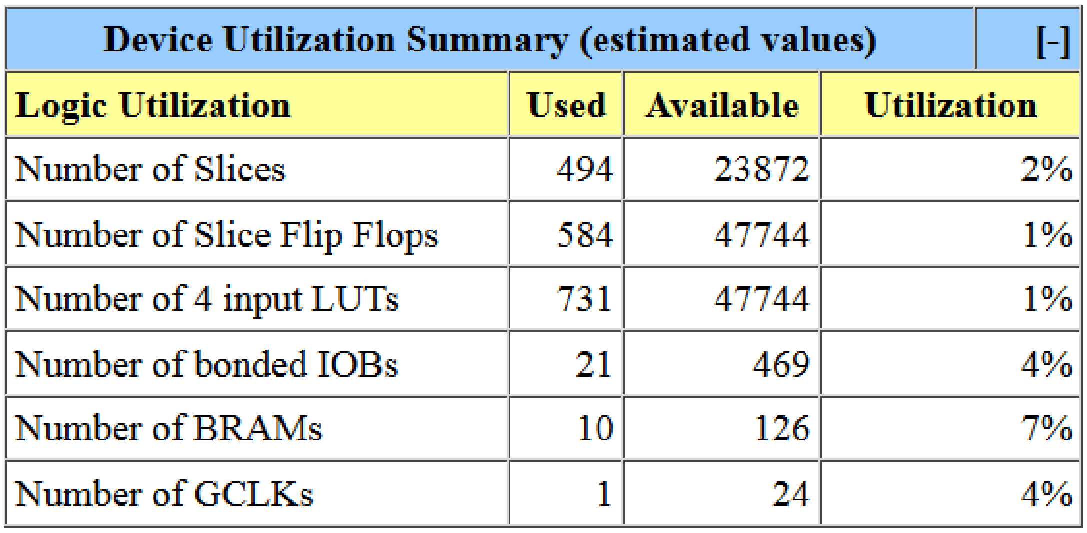
Bicubic Transmission Module is responsible for synchronizing bicubic outputs with Total variation filter input and shock filter outputs. It consists of four line buffers where two lines are accumulated and sent sequentially when third line arrives.

Resource utilizations of shock filter, bicubic interpolation, bicubic transmission module are given on Figure- 5-14, Figure- 5-15 and Figure- 5-16 respectively.



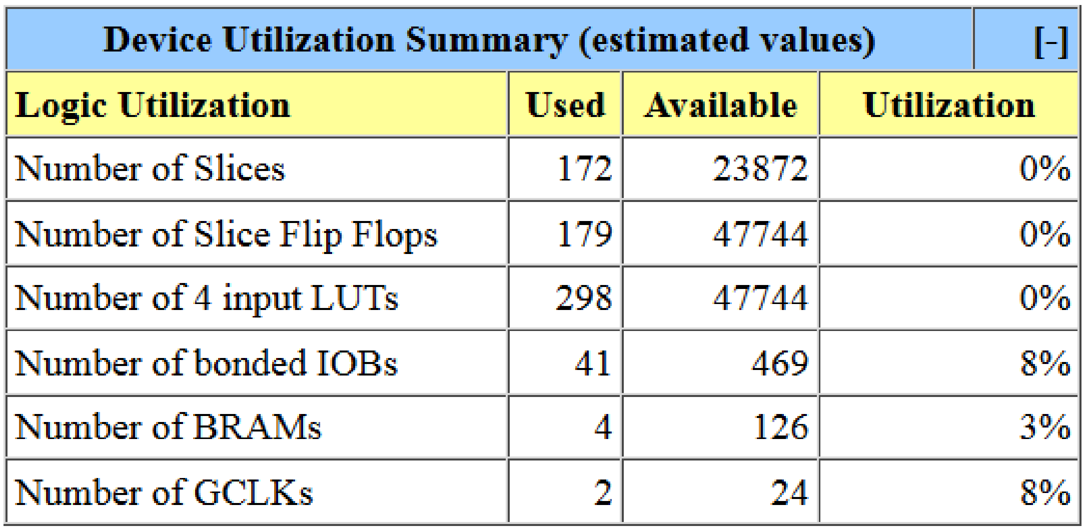
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	960	23872	4%
Number of Slice Flip Flops	1158	47744	2%
Number of 4 input LUTs	1683	47744	3%
Number of bonded IOBs	50	469	10%
Number of BRAMs	4	126	3%
Number of GCLKs	1	24	4%

Figure- 5-14: Shock Filter Resource Utilization



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	494	23872	2%
Number of Slice Flip Flops	584	47744	1%
Number of 4 input LUTs	731	47744	1%
Number of bonded IOBs	21	469	4%
Number of BRAMs	10	126	7%
Number of GCLKs	1	24	4%

Figure- 5-15: Bicubic Interpolation Resource Utilization



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	172	23872	0%
Number of Slice Flip Flops	179	47744	0%
Number of 4 input LUTs	298	47744	0%
Number of bonded IOBs	41	469	8%
Number of BRAMs	4	126	3%
Number of GCLKs	2	24	8%

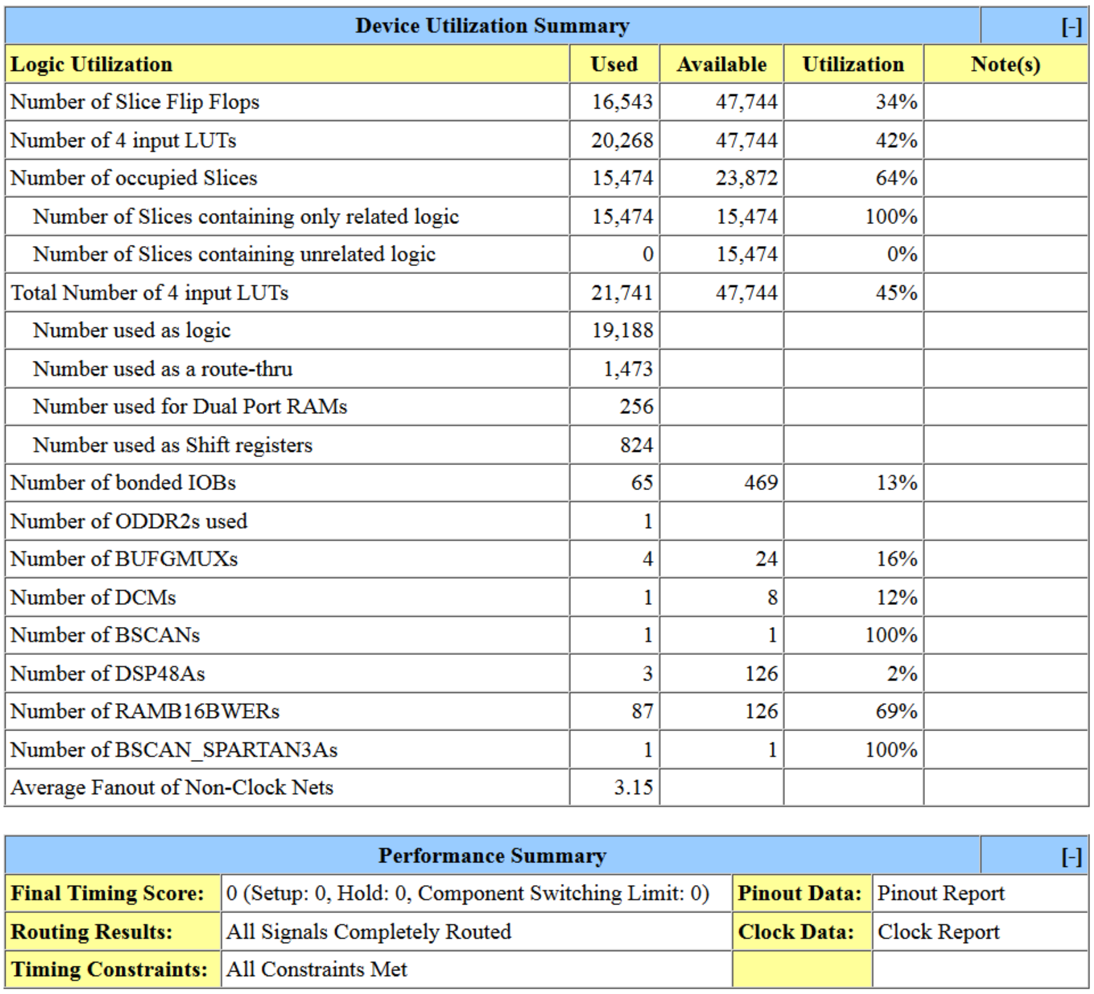
Figure- 5-16: Bicubic Transmission Module Resource Utilization

5.8 Final Design

The final design is collected under the top module. Top module signals are connected to RS232 pins, SRAM pins, clock generator pins. The clock on board has 27 MHz frequency. With 27MHz clock we can achieve 33 frames per second speed with chosen 640x480 output pixel rate. The chosen resolution is taken from cameras that are used widely for low resolution video capturing.

The final design, together with its constraints, has a limit of 42 MHz clock rate. This means the fps figure can go up to 51 fps if clock speed is changed.

The final design is implemented and tested on our board. Results are going to be given on next chapter. Figure- 5-17 shows the final implemented resource utilization.



Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	16,543	47,744	34%	
Number of 4 input LUTs	20,268	47,744	42%	
Number of occupied Slices	15,474	23,872	64%	
Number of Slices containing only related logic	15,474	15,474	100%	
Number of Slices containing unrelated logic	0	15,474	0%	
Total Number of 4 input LUTs	21,741	47,744	45%	
Number used as logic	19,188			
Number used as a route-thru	1,473			
Number used for Dual Port RAMs	256			
Number used as Shift registers	824			
Number of bonded IOBs	65	469	13%	
Number of ODDR2s used	1			
Number of BUFGMUXs	4	24	16%	
Number of DCMs	1	8	12%	
Number of BSCANs	1	1	100%	
Number of DSP48As	3	126	2%	
Number of RAMB16BWERs	87	126	69%	
Number of BSCAN_SPARTAN3As	1	1	100%	
Average Fanout of Non-Clock Nets	3.15			

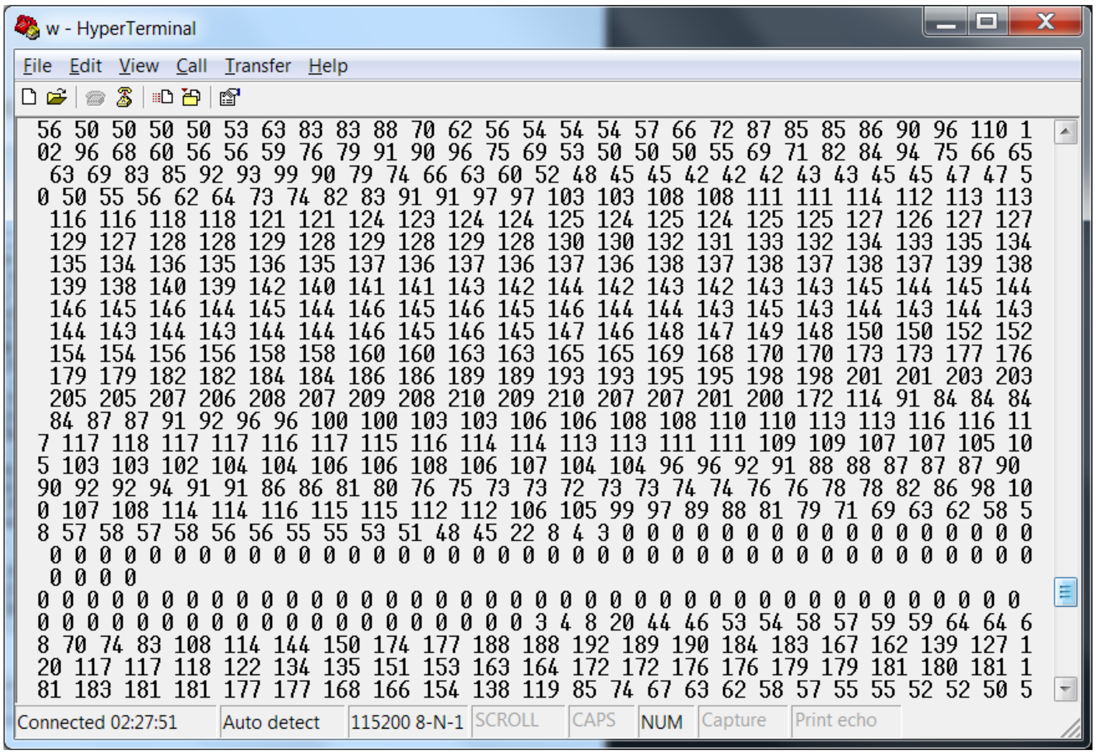
Performance Summary			[-]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Figure- 5-17: Implemented Design Resource Utilization

CHAPTER 6

EXPERIMENTAL RESULTS

Input to FPGA is sent and output from FPGA is received via RS232 serial communication channel using HyperTerminal software.



```
 56 50 50 50 50 53 63 83 83 88 70 62 56 54 54 54 57 66 72 87 85 85 86 90 96 110 1
 02 96 68 60 56 56 59 76 79 91 90 96 75 69 53 50 50 50 55 69 71 82 84 94 75 66 65
  63 69 83 85 92 93 99 90 79 74 66 63 60 52 48 45 45 42 42 42 43 43 45 45 47 47 5
 0 50 55 56 62 64 73 74 82 83 91 91 97 97 103 103 108 108 111 111 114 112 113 113
  116 116 118 118 121 121 124 123 124 124 125 124 125 124 125 125 127 126 127 127
  129 127 128 128 129 128 129 128 129 128 130 130 132 131 133 132 134 133 135 134
  135 134 136 135 136 135 137 136 137 136 137 136 138 137 138 137 138 137 139 138
  139 138 140 139 142 140 141 141 143 142 144 142 143 142 143 143 145 144 145 144
  146 145 146 144 145 144 146 145 146 145 146 144 144 143 145 143 144 143 144 143
  144 143 144 143 144 144 146 145 146 145 147 146 148 147 149 148 150 150 152 152
  154 154 156 156 158 158 160 160 163 163 165 165 169 168 170 170 173 173 177 176
  179 179 182 182 184 184 186 186 189 189 193 193 195 195 198 198 201 201 203 203
  205 205 207 206 208 207 209 208 210 209 210 207 207 201 200 172 114 91 84 84 84
  84 87 87 91 92 96 96 100 100 103 103 106 106 108 108 110 110 113 113 116 116 11
 7 117 118 117 117 116 117 115 116 114 114 113 113 111 111 109 109 107 107 105 10
 5 103 103 102 104 104 106 106 108 106 107 104 104 96 96 92 91 88 88 87 87 87 90
 90 92 92 94 91 91 86 86 81 80 76 75 73 73 72 73 73 74 74 76 76 78 78 82 86 98 10
 0 107 108 114 114 116 115 115 112 112 106 105 99 97 89 88 81 79 71 69 63 62 58 5
 8 57 58 57 58 56 56 55 55 53 51 48 45 22 8 4 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 4 8 20 44 46 53 54 58 57 59 59 64 64 6
 8 70 74 83 108 114 144 150 174 177 188 188 192 189 190 184 183 167 162 139 127 1
 20 117 117 118 122 134 135 151 153 163 164 172 172 176 176 179 179 181 180 181 1
 81 183 181 181 177 177 168 166 154 138 119 85 74 67 63 62 58 57 55 55 52 52 50 5
```

Connected 02:27:51 | Auto detect | 115200 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

Figure- 6-1: HyperTerminal Interface

6.1 Detailed Results

Details of the results are given in four figures from lena image. Figure- 6-2 shows edge reconstruction success and better estimation for thin lines. The images are original, FPGA result, Matlab Bicubic result from left to right respectively. The results of numerically compared images are given on the next topic.

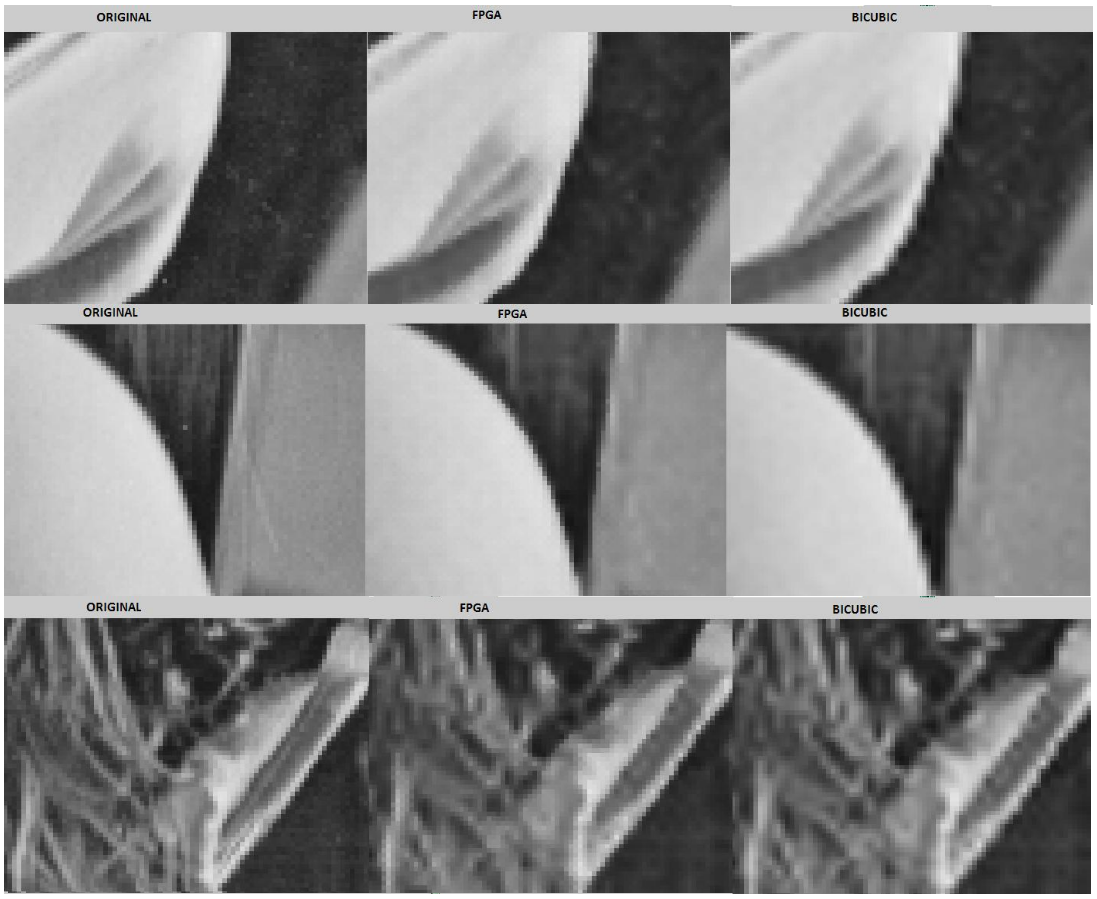


Figure- 6-2: Details of Images

6.2 Degradations

The main drawback of using FPGA is quantization of all values as mentioned on chapter 5. We have quantized bilateral filter coefficients to 2^7 to optimize between resource utilization and quality. That caused edge adaptive interpolation to fill new pixels with slightly different values than they should normally have. This is due to quantized possible values of a new pixel.

Other possible cause of artifacts is the approximation of L2 Norm, square root of sum of squares. The approximation is quantized to three levels where two values are equal, within close proximity and different with an order of magnitude. This causes only normalization problems but it might increase the value of singularities slightly.

Another cause of possible artifacts is division. First type of division error is due to rounded decimals of division results which is inconceivable. Second type of division error is due to bit loss from division which is done by bit shifting. Bit shifting may cause value of a pixel to change by one, two or four.

6.3 Numerical Results

The FPGA result is compared with the original image together with the theoretical result obtained from Matlab and theoretical bicubic result obtained from Matlab. Table 6.1 shows comparison results of theoretical and practical results.

Table- 6-1: PSNR Comparison of Experimental Results

PSNR	FPGA Bicubic	FPGA EDAT	Matlab Bicubic	Matlab EDAT
Kodakimg1	23.6578	**23.6837**	24.5685	24.5803
Kodakimg2	30.0495	**30.1378**	30.4216	30.4589
Kodakimg3	30.6926	**30.7509**	31.1171	31.4221
Kodakimg4	21.7534	**21.9398**	22.1925	22.2952
Kodakimg5	28.9176	**28.9295**	31.3140	31.5496
Kodakimg6	24.9994	**25.0287**	26.9734	27.0695
Average	26.6784	**26.7451**	27.7645	27.8959

Table- 6-2: SSIM Comparison of Experimental Results

SSIM	FPGA Bicubic	FPGA EDAT	Matlab Bicubic	Matlab EDAT
Kodakimg1	0.8879	**0.8897**	0.8987	0.9000
Kodakimg2	0.8463	**0.8467**	0.8428	0.8443
Kodakimg3	**0.9606**	0.9548	0.9708	0.9713
Kodakimg4	0.8862	**0.8896**	0.9065	0.9098
Kodakimg5	0.9264	**0.9296**	0.9460	0.9467
Kodakimg6	0.8407	**0.8420**	0.9021	0.9028
Average	0.8913	**0.8921**	0.9112	0.9125

The EDAT exceeds the quality of the basic method bicubic. Figure- 6-3 shows whole and close-up comparsions of bicubic (left) and EDAT (right).

Reisenbeikafie

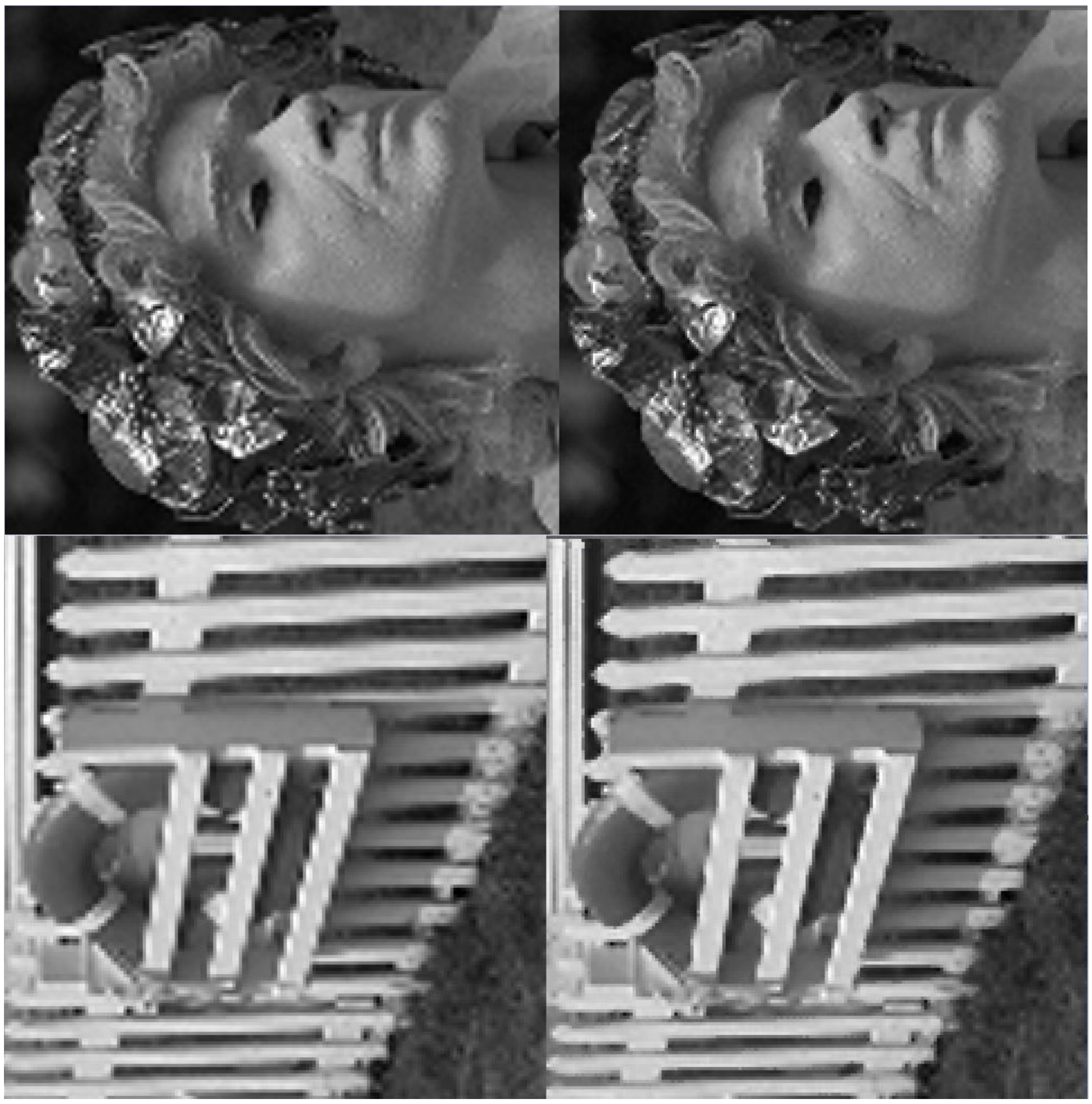


Figure- 6-3: Comparison of Bicubic(left) vs EDAT(right) from FPGA Results

Figure- 6-4 shows the comparison of FPGA and Matlab results respectively. Occasional artifacts on leftmost images can be seen which are caused by the aforementioned degredations.

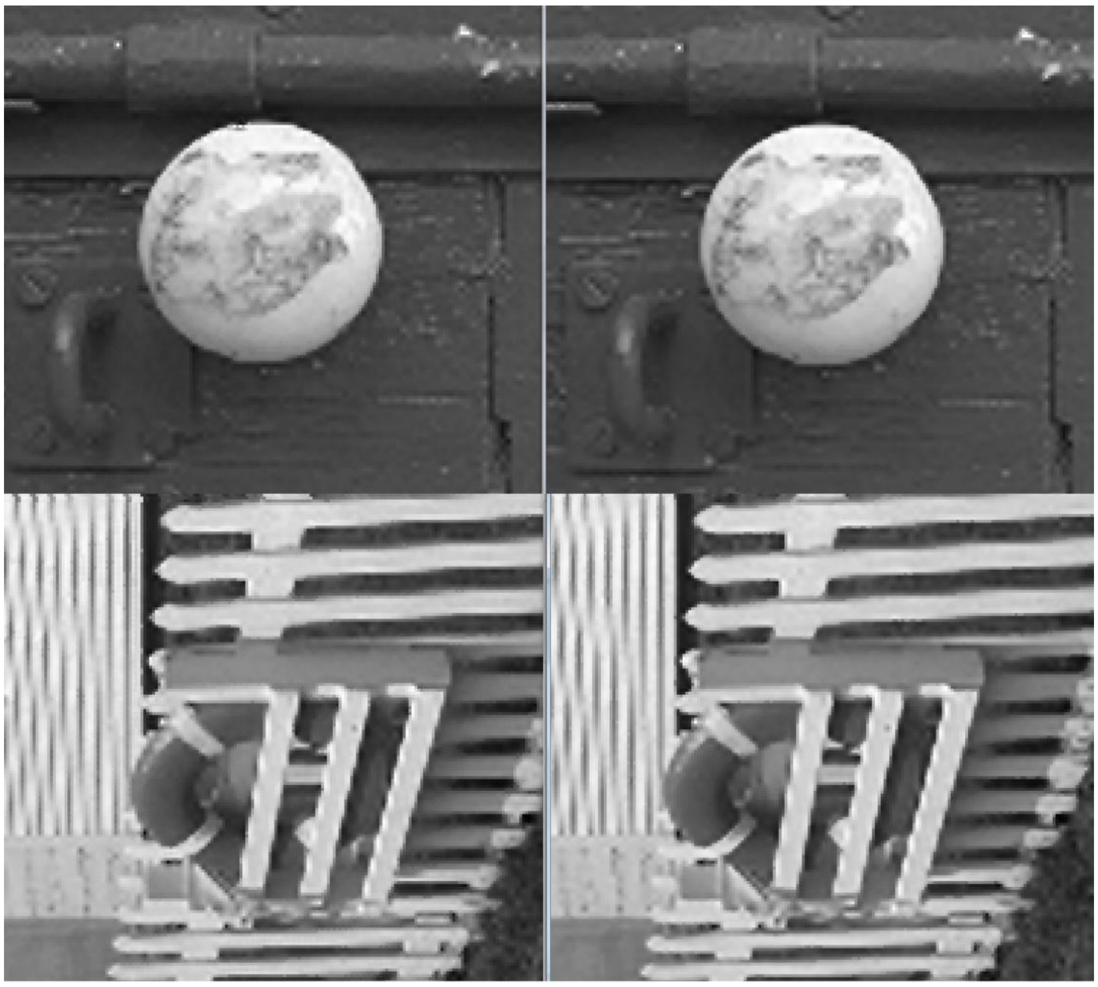


Figure- 6-4: Comparison of FPGA (left) vs. Matlab (right) Results

6.4 Running Time

The current FPGA board had 27MHz clock available for operation. By using available clock and Digital Clock Manager (DCM) blocks for faster clocks, we have achieved 33 frames per second for 640x480 output resolution. The reports from ISE software indicate that maximum achievable clock speed is 42.264 MHz which means the temporal speed can be extended to 52 frames per second or 800x600 resolution at 33 fps. With this configuration bicubic implementation can be run with speeds up to 96 MHz. Although bicubic can output twice the fps or approximately

1.5 times higher resolutions the reconstruction quality of EDAT may still surpass the need for speed.

Considering the used FPGA is 7 years old technology, these results are promising for higher resolution applications.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusions

In this thesis we have proposed two algorithms (NDUID, EDAT) and implemented one algorithm on FPGA for a real-time application (EDAT).

NDUID uses a non-dyadic upsampling cascade to slowly interpolate an image to a reasonable size for better edges. Since the upsampling cascade does not contribute much to high frequency content a texture enhancement block called non-blind deconvolution is used. The deconvolution process uses heavy tailed gradient distribution prior embedded into frequency domain iterative deconvolution.

EDAT uses Total Variation decomposition to split the image into high frequency texture component and low frequency structure component. Since both components carry different information they have been treated differently. Texture component is upsampled by bicubic interpolation and is not retouched since it does not include visual information other than local details. Structure component is, however, interpolated by an edge adaptive interpolation which is based on bilateral filter. The smooth and continuous edges together with lack of noise (since it is separated to texture component) enabled us to use a special filter called shock filter which, in short, converts a sinusoid into square-like function. Then two components have been combined.

The algorithms have been chosen according to their speed. Even though algorithms had learning based, frequency domain based, and MAP based minimizations included, all of them were implementable on an FPGA.

The resulting algorithm, EDAT, is chosen for FPGA because of fewer computations, better numerical and visual results. EDAT performed well on PSNR and SSIM results compared to conventional bicubic and the included original algorithms.

FPGA results are acceptable because of better edge reconstruction and visual results. The speed limit allows the implemented algorithm to be used with real cameras for low cost applications. Since the FPGA board that we have used is a seven years old technology, the algorithm is open for further enhancement in temporal and spatial resolutions.

7.2 Future Work

The FPGA implementation is not made for a real time application other than calculation of running time for real time application. The foremost refinement would be to add frame-grabber and frame-buffers for video input and output to process a real time video stream from a camera.

In our work we have not paid attention to texture component on EDAT because it cannot be modeled with any known method well enough to process it further with the availability for real-time application. One of the refinements can be to devise a method to further enhance texture component.

Another refinement for EDAT can be to separate the image into four, or more, wavelet transform pairs. Since some of the edge information is polluted due to composition of bicubic interpolated texture and edge adaptive interpolated structure component. TV decomposition block can be used to filter the image with more blurring to spare more information for remaining texture component. Then the remaining high frequency content can be decomposed into edge directional pairs for better signal processing.

REFERENCES

[1] Hung K W Siu W C, "Fast Image Interpolation Using the Bilateral Filter," *IET Image Processing*, vol. 6, no. 7, pp. 877-890, June 2012.

[2] Marquina A, Osher S, Dinov I, Toga A, Horn J Joshi S, "Fast Edge-Filtered Image Upsampling," *International conference on image processing*, vol. 18, pp. 1165-1167, 2011.

[3] Kameyama K, Ohmiya Y, Lee J, Toriaichi K Mori T, "Image resolution conversion based on an edge-adaptive interpolation kernel," *IEEE PACRIM*, pp. 497-500, 2007.

[4] Orchard M. T. Li X, "New Edge Directed Interpolation," *IEEE Trans. Image Process.*, pp. 1521-1527, 2001.

[5] Klosowski J T Krishnan S, "Guided image upsampling using bitmap tracing," *International conference on image processing*, September 2013.

[6] Sun J, Tang X He K, "Guided Image Filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397-1409, June 2013.

[7] Fattal R., "Image Upsampling vie Imposed Edge Statistics," in *ACM Transactions on Graphics*, 2007, pp. 95-100.

[8] Li Z, Jia J, Tang C-K Shan Q, "Fast Image/Video Upsampling," *ACM Trans. Graph.*, pp. 153:1-7, 2008.

[9] Hsu C-C, Lin C-W, Kang L Lin C, "Fast Deconvolution Based Image Super-Resolution Using Gradient Priors," Nation Science Council, 2010.

[10] Sun J, Long Q, Shum H Yuan L, "Image Debluring with Blurred/Noisy Image Pairs," *SIGGRAPH*, 2007.

[11] Cvetkovic Z, Vetterli M Chang S G, "Resolution enhancement of images using

wavelet transform extrema extrapolation," *Proc. ICASSP*, vol. 4, pp. 2379-2382, May 1995.

[12] Muresan D D, Parks T W Kinebuchi K, "Image interpolation using wavelet-based hidden Markov trees," *ICASSP*, vol. 3, pp. 7-11, May 2001.

[13] Vlachos T Temizel A, "Wavelet domain image resolution enhancement," *IEE Proc. Vis. Image Signal Process*, vol. 153, pp. 25-30, 2006.

[14] Huang J Yang J, *Image Super-Resolution: Historical overview and future challenges.*, 2010.

[15] Nikou C, Kondi L P Vrigkas M, "A Fully Robust Framework For MAP Image Super-Resolution," *Proceedings of IEEE International Conference on Image Processing*, pp. 2225-2228, 2012.

[16] Robinson M D, Elad M, Milanfar P Farsiu S, "Fast and robust multiframe super resolution," *IEEE Transactions on Image Processing*, vol. 13, no. 10, pp. 1327-1344, October 2004.

[17] Stevenson R L Borman S, "Simultaneous Multi-frame MAP Super-Resolution Video Enhancement using Spatio-temporal Priors," *IEEE*, 1999.

[18] Elad M Datsenko D, "Example based single document image super resolution: a global MAP approach with outlier detection," *Multidimensional System Signal Processing*, vol. 18, pp. 103-107, 2007.

[19] Nguyen T Li M, "Markov random field model-based EDge-directed image interpolation," *International Conference on image processing*, pp. II-93-96, 2007.

[20] Wu X Behnad A, "Image interpolation with hidden markov model," *Proc. IEEE Int. COnf. Acoustics, Speech, Signal Processing (ICASSP)*, pp. 874-877, March 2010.

[21] Zisserman A Capel D, "Computer vision applied to super-resolution," *IEEE signal processing magazine*, vol. 20, pp. 75-86, 2003.

[22] Sakuta Y, Watanabe M, Goto T, Hirano S Sakurai M, "Super Resolution Utilizing Total Variation Regularization and a Shock Filter," *ICIP*, pp. 1-10,

2012.

[23] Chambolle A, "An Algorithm for Total Variation Minimization and Applications," *Journal of Mathematical Imaging and Vision*, pp. 89-97, 2004.

[24] Malgouyres F Guichard F, "Total Variation Based Interpolation," 1997.

[25] Suzuki S, Goto T, Hirano S, Sakurai M Yoshikawa A, "Super Resolution image reconstruction using total variation regularization and learning based method," in *International conference on image processing*, vol. 17, September 2010, pp. 26-29.

[26] Osher S, Shen J Chan T F, "The digital TV filter and non linear denoising," *IEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 10, no. 2, pp. 231-241, 2001.

[27] Komatsu T Saito T, "ımage Processing Approach based on Nonlinear Image Decomposition," *IECE Trans. Fundamentals*, pp. 696-707, 2009.

[28] Jones T R, Pasztor E C Freeman W T, "Example-Based Super-Resolution," *Image-Based Modeling, Rendering and Lighting IEEE*, pp. 56-65, 2002.

[29] Freedman G Fattal R, "Image and Video Upscaling from Local Self-Examples," *ACM Trans. Graph*, pp. 106:1-11, 2011.

[30] Le A Bhanu B, "Image super-resolution by extreme learning machine," *International Conference on Image Processing*, pp. 2209-2212, 2012.

[31] Pan Q, "Single-frame image super-resolution based on local similarity projection model," *IEEE computer society International conference on digital home*, pp. 357-360, 2012.

[32] Bagon S, Irani M Glasner D, "Super-resolution from a single image," *ICCV09*, pp. 349-356, 2009.

[33] Kanade T Baker S, "Limits on super-resolution and how to break them," *CVPR, IEEE Computer Society*, pp. 2372-2379, 2000.

[34] Jia J, Shan Q Agarwala A, "High-quality Motion Deblurring from a Single Image," *ACM Trans. Graph.*, pp. 73:1-10, 2008.

[35] Singh B, Hertzmann A, Roweis S, Freeman W Fergus R, "Removing camera shake from a single photograph," *Proceedings of SIGGRAPH*, pp. 787-794, 2006.

[36] Akar G. B. Turgay E, "Texture preserving multi frame super resolution with spatially varying image prior," *19th IEEE International Conference on Image Processing*, pp. 2205-2208, January 2012.

[37] Wu X, Zhang X, "Image Interpolation by Adaptive 2-D Autoregressive Modelling and Soft Decision Estimation," *IEEE Trans. Image Process.*, pp. 877-896, 2008.

[38] Elad M, "On the Origin of the Bilateral Filter and Ways to Improve It," *IEEE Trans. Image Process.*, pp. 1141-1151, 1999.

[39] Maduchi R Tomasi C, "Bilateral Filtering for Gray and Color Images," *Proc. Int. Conf. Computer Vision (ICCV)*, pp. 839-846, 1998.

[40] Siu W C Hung K W, "Improved Image Interpolation Using Bilateral Filter for Weighted Least Square Estimation," in *Int. Conf. Image Processing (ICIP)*, Hong Kong, 2012, pp. 3297-3300.

[41] Osher S, Fetami E Rudin L, "Nonlinear Total Variation Based Noise Removal Algorithm," *Physica D*, pp. 259-268, 1992.

[42] Rudin L I Osher S, "Feature-Oriented Image Enhancement Using Shock Filter," *Society of Industrial and Applied Mathematics*, pp. 919-940, 1990.

[43] Wang Z, Gupta S, Bovik A C, Markey M K Sampat M P, "Complex Wavelet Structural Similarity: A New Image Similarity Index," *IEEE Transactions On Image Processing*, vol. 18, no. 11, pp. 2385-2401, 11 November 2009.

[44] Zhu Q Y, Siew C K Huang G B, "Extreme learning machine a new learning scheme of feedforward neural networks," in *Proc. IJCNN*, 2004.