

# **ApproxVision: Approximating the Image By Exploiting the Limitations of Human Visual System**

Dinesh Buswala

A Thesis Submitted to  
Indian Institute of Technology Hyderabad  
In Partial Fulfillment of the Requirements for  
The Degree of Master of Technology



भारतीय प्रौद्योगिकी संस्थान हैदराबाद  
Indian Institute of Technology Hyderabad

CANDLE research lab  
Department of Computer Science & Engineering

June 2018

## Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.



\_\_\_\_\_  
(Signature)

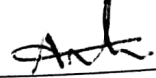
\_\_\_\_\_  
(Dinesh Buswala)

CS16MTECH11005

(Roll No.)

## Approval Sheet


This Thesis entitled ApproxVision: Approximating the Image By Exploiting the Limitations of Human Visual System by Dinesh Buswala is approved for the degree of Master of Technology from IIT Hyderabad




Antony Prasad, Examiner  
Dept. of Computer Science & Eng  
IITH



Dr. Bheemasjuna Reddy, Examiner  
Dept. of Computer Science & Eng  
IITH



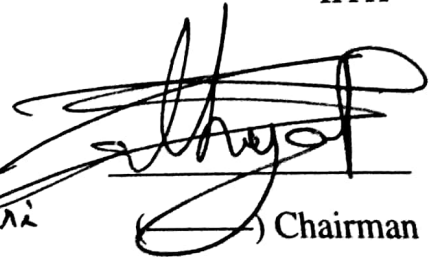
Dr. SPARSH, MITTAL, Adviser  
Dept. of Computer Science & Eng  
IITH

Dinesh has no co-adviser  


( ) Co-Adviser

Dept. of of Computer Science & Eng  
IITH

Dr. Sathya Peri



Chairman  
Dept. of Computer Science & Eng  
IITH

## **Acknowledgements**

First and foremost, I have to thank my parents (Mr. Raju Buswala and Mrs. Lajja Buswala) for their love and support throughout my life. Thank you both for giving me the strength to reach for the stars and chase my dreams. My sisters (Ms. Karishma Buswala) and My elder brother (Mr. Yogesh Buswala) deserve my wholehearted thanks as well.

I would like to sincerely thank my supervisor, Dr. Sparsh Mittal, for his guidance and support throughout this study, and especially for his confidence in me. I would also like to thank Dr. Sathya Peri, Dr. Saurabh Joshi and Dr. Srijith P. K. for serving as a members of my thesis committee during my stage 1 and stage 2. Their comments and questions were very beneficial in my completion of the manuscript and especially at interview time. I learned from their insight a lot. I extend my special thanks to B. Tulja Vamshi Kiran, Gaurav Garg, Mohd. Saalim, Sumanta Patro, Debashish Mishra and Mukesh Kumar Giluka for their inputs and help with my thesis. Lastly, I thank all the faculty of IIT Hyderabad.

# Dedication

I dedicate this thesis to my parents, my brother, my sister and the people around the world who believe that  
hard work conquers all.

## **Abstract**

Approximate computing has recently emerged as a promising approach to the energy-efficient design of digital systems. Approximate computing relies on the ability of many systems and applications to tolerate some loss of quality or optimality in the computed result for saving energy and performance enhancement.

In image processing, applications impose high energy consumption in loading and accessing the image data in the memory. Fortunately, most image processing applications can tolerate approximation in processing. The quality of service (QoS) of image processing applications depends upon the human visual system. The Human Visual system has some limitations like weak peripheral vision and not able to distinguish the difference between the quality of the original image and processed image when PSNR value is greater than 30 dB. These limitations give us a hint that instead of approximating the entire image we should take the peripheral part of the image because the human eye has the lower peripheral vision and high center vision and at the same time processed image has PSNR value greater 30 dB to gain the excellent quality of an image. Leveraging these facts we proposed one approximate computing technique that will save energy without sacrificing the QoS. We will approximate only the peripheral part of the image, and in the peripheral region, we change lower bits in each pixel because the contribution of lower bits in a pixel is less compare to higher order bits in a pixel. The proposed technique will take care of the limitations of the human visual system to approximate the images.

# Contents

Declaration . . . . .	ii
Approval Sheet . . . . .	iii
Acknowledgements . . . . .	iv
Abstract . . . . .	vi
<b>Nomenclature</b>	<b>vii</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Related Research Work . . . . .	4
1.2 Thesis Organization . . . . .	5
<b>2 Challenges and Opportunities</b>	<b>6</b>
<b>3 Proposed Solution</b>	<b>7</b>
3.1 Modification of the proposed solution . . . . .	9
<b>4 Experiment</b>	<b>11</b>
4.1 How to find the optimal partition and optimal threshold value . . . . .	11
4.2 Approximate only on the peripheral part . . . . .	14
4.3 Summary . . . . .	16
<b>5 Conclusion and Future Work</b>	<b>17</b>
5.1 Conclusion . . . . .	17
5.2 Future Work . . . . .	18
<b>References</b>	<b>19</b>

# Abbreviations

<b>Acronym</b>	<b>What (it) Stands For</b>
<b>PSNR</b> .....	Peak Signal to Noise Ratio
<b>SSIM</b> .....	Structural Similarity
<b>MSB</b> .....	Most Significant Bit
<b>LSB</b> .....	Least Significant Bit
<b>STT-RAM</b> .....	Spin Transfer Torque Random Access Memory
<b>SRAM</b> .....	Static Random Access Memory
<b>DRAM</b> .....	Dynamic Random Access Memory
<b>HVS</b> .....	Human Visual System
<b>AC</b> .....	Approximate Computing
<b>FP</b> .....	Floating Point
<b>CPU</b> .....	Central Processing Unit
<b>MLC</b> .....	Multi Level Cell
<b>dB</b> .....	Decibel



# Chapter 1

## Introduction

There are many applications where computation accuracy can be traded off to achieve better performance and energy efficiency. The techniques to accomplish this trade-off is called approximate computing. Approximation computing preferred in many different domains such as machine learning, image processing, and video processing. Different existing algorithms in these domains have been approximated by programmers to achieve better performance and save energy because these applications can bear some computational error. Image processing algorithms are good candidates for the approximation as random variation in results will not be noticeable by the user. For example, if we manipulate some lower bits in the pixel, then it doesn't impact much on overall values of the pixel. Even there is massive scope for approximation in some primitive data types (that are used to store real value data only) in applications where that much precision is not required. In image processing applications the quality of service depends upon the human visual system. There are many algorithms made to approximate the image to gain performance and save energy. These algorithms are not favorable to the characteristics of the human visual system.

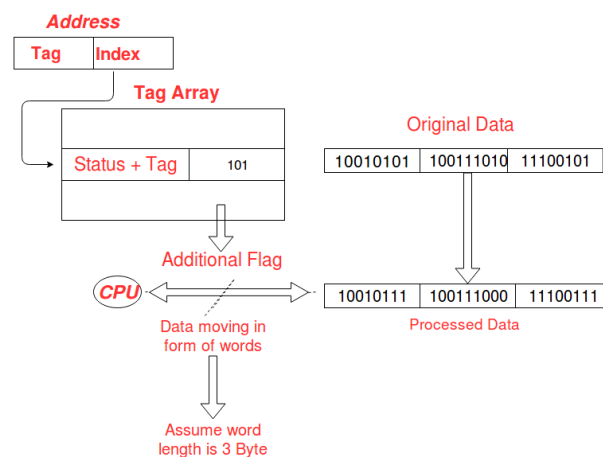


Figure 1.1: Illustration of the Proposed Approximate Storage approach



Figure 1.2: Peripheral vision basic functions

In this paper, we proposed the technique that favors human visual system, i.e., approximate the lower bits of the pixel. Our solution will divide each pixel into two part. We called them MSB part (contains higher order bits) and LSB part (contains lower order bits). After that, we count the number of ones in LSB part if most of the bits in LSB are one then we set one flag bit else (few bits are one) then we reset the flag bit. Keep this flag bit in tag field so that before accessing the data we read the status of this flag bit. While accessing the data read MSB as it and for LSB read the state of the flag bit if the flag bit is set then append all ones to MSB else if the flag bit is reset then append all zeros to MSB. In a broad sense, we can say that we are using the flag bit at a byte level granularity [1]. The figure 1.1 shows that how original data is processed when reading the original data based on the status of flag bit corresponding to each pixel.

We also extend this approach by understanding the limitations of the human visual system [2]. The human visual system suffers from low peripheral vision and high center vision. By taking this limitation, we can use the approximation limited to the outer area of the image instead of the complete picture.

Peripheral vision [3] is a part of the view that occurs only on the side gaze. The figure 1.2 explains how the human eye sees the standard image when it has deficient peripheral vision. That gives us motivation that we should approximate the pixel values that are in the outer area of the image because of low peripheral vision. Actually by approximating only the pixels that are on the outer part of the image we are reducing the error rate.

## 1.1 Related Research Work

Approximate computing helps us to gain performance and save energy by compromising the computational accuracy. This trade-off can be extended to increase the hardware lifetime by storing data approximately. Approximate storage helps us to improve the lifetime of the memory. In this paper [4] the researcher come up with the technique that will help to extend the lifetime of the memory by using two ways. The first allows errors in multi-level cells by reducing the number of programming pulses applied to write them. The second mechanism mitigates wear-out failures and extends memory endurance by mapping approximate data onto blocks that have exhausted their hardware error correction resources. The approximation can be made at arithmetic circuits also.

In this paper [5] the researcher explains the approximate arithmetic circuits like approximate full adders, multiple-bit approximate adders, approximate multipliers, and approximate logic synthesis. Approximate computing [6] can be used as an error-resilience of programs and users. The perceptual limitations of humans provide scope for approximate computing in visual and other computing applications. Similarly, many programs have noncritical portions, and small errors in these do not affect the quality of result significantly. For example, in [7] researcher note that in a 3D raytracer application, 98% of floating point operations and 91% of data accesses are approximable. Similarly, since the lower-order bits have smaller significance than the higher-order bits, approximating them may have only a minor impact on quality of result [8, 9, 10, 4].

However, there is an enormous scope of approximation in image data because changing the bits in the pixel does not make much difference in the quality of the image. In this paper [11], the researchers propose an approximate image processing scheme that improves system energy efficiency without upsetting image quality requirement of applications. There design consists of an approximate image storage mechanism that strives to only write the soft bits in MLC STT-MRAM main memory with small write current and a memory mode controller that determines the approximation of image data and coordinates across precise/approximate memory access modes.

## **1.2 Thesis Organization**

The rest of the thesis is organized as follows:

- In Chapter 2, we highlight the obstacle arises in the completion of our proposed solution and how we can make our observations more reliable.
- Chapter 3 presents proposed solution and modification in our proposed solution to gain more performance.
- Chapter 4 describes experimental results for the completion and justification of the proposed algorithm.
- Finally, Chapter 5 concludes the work done and highlights the future course of research work.

## Chapter 2

# Challenges and Opportunities

In our proposed technique, there are many possible combinations for partitioning the pixel into MSB and LSB. Each partition will introduce some error in the image. The combination which produces the least error is our desired partition. Finding the partition with least error is the challenging part of our algorithm.

### **Challenge I: How to find the optimal partition**

While dividing the pixel into MSB part and LSB part, we observed that there are many possible combinations exists like 3 Bits in LSB and 5 Bits in MSB, 4 Bits in LSB and 4 Bits in MSB, 5 Bits in LSB and 3 bits in MSB, and so on. Each combination inserts some error in the original image. The partition with least error, we called it Optimal partition value. Finding the optimal partition value is the most challenging part because it is the heart of our proposed solution. To see the error in the partitions, we initially used mean squared error [12] but it is not enough for deep inspection of each partition because it is not very well match to perceived visual quality. To make our observations more reliable for finding the optimal partition we use two other metrics such as PSNR and SSIM [13]. PSNR and SSIM value is the extension of the human visual system. These metrics ensure that the quality of partitions we get depends upon the human visual system.

### **Challenge II: How to find the optimal threshold value (number of ones in LSB to set or reset flag bit)**

After dividing the pixel into LSB and MSB then we need the count of the number of ones in LSB based on this count value flag bit is set or reset. There are many ones possible in the LSB part. If ' $m$ ' bits are there in LSB part, then there are ' $m$ ' possible count values exists, and each of them injects some error in the obtained image. Thus, error in the obtained images is introduced by both partition of MSB and LSB and the count value to set or reset the flag bit. Now our next challenge is to find the best count value to set or reset flag bit corresponding to optimal partition we called it as the optimal threshold value. The count value has a significant impact on PSNR value because by changing the LSB part to all ones, or all zeros modify the decimal equivalent value of the original value. Thus, it also plays a vital role in our observations.

## Chapter 3

# Proposed Solution

In processing environment, there are many applications in which some degree of variation or error permitted in the result of the computation. There are many valuable domains where approximation can significantly improve application performance, i.e., multimedia processing and gaming.

The baseline to compare any approximation technique for images is the human visual system. Most of the approximation techniques focus on the quality of the image obtained. However, the final quality of service depends upon the human visual system. The existing approximation techniques are not examining the limitation of the human visual system. There are many opportunities to approximate the image by considering the limitation of the human visual system. By using these limitations, one can approximate the image with the high-grade quality of service and that quality service is directly proportional to the limitation of the human visual system.

Peak signal-to-noise ratio often abbreviated PSNR, measured in decibels and computed between two images. This ratio is frequently used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image. The human visual system has one limitation like if the PSNR value of the processed image is greater than 30 dB, then the human eye cannot distinguish between the original one and processed one. So getting the higher PSNR would not make much difference in terms visual inspection by Human eye, leveraging these limitations we can process image faster without sacrificing the quality of the image. Also, the value of lower bits or least significant bits in pixel does not contribute that much to the quality of service compared to higher bits or most significant bits in pixels. So If we approximate the lower bits in pixels [5], then it does not impact much compare to higher bits in pixels. By considering these limitations, we proposed our algorithm to approximate the pixels.

Algo. 1 explains the proposed solution for each pixel. In our algorithm, we divide the pixel or byte into two parts we call them LSB (Least significant bit) part and MSB (a Most significant bit) part. The MSB part contains higher order bits whereas LSB part contains lower order bits. The higher order bits contribute more to the equivalent value in decimal [5, 14, 15]. After dividing the pixel value in MSB and LSB part. We count the number of ones in LSB part if most of the bits in LSB part are one then we can assume that all the bits in LSB part are one and set the flag bit corresponding to that pixel [1]. Otherwise (most of the LSB are not one) we reset the flag bit corresponding to that pixel. Now keep this flag bit in tag field to access it earlier, i.e., before reading the actual data. While reading the data, read-only MSB part and for LSB part use the status of the flag bit. If the flag bit is one, then append ones as the total number of bits in LSB part or if the flag bit is zero then append zeros as the total number of bits in LSB part. The whole solution depends upon the fact that

**Algorithm 1** ApproxVision

---

```
1: Each pixel has one flag bit associated with it
2: Divide the pixel or byte into two parts MSB and LSB
3: Count the number of bits in LSB part
4: if (Most of the bits in LSB are one) then
5:   'set' flag bit
   /**  
   indicates the majority of bits in LSBs are ones so we can assume all bits in LSB are one */
6: else ▷ very few bits of LSB are ones
7:   then 'reset' flag bit
   /**  
   indicates the majority of bits in LSB is Zeroes so we can assume all are zeroes */
8: end if
9: Keep this flag bit in the tag field
10: while reading the data, read MSB as it and for LSB read the status of flag bit if it is one then append all
    one to MSB or if it is zero then append all zero to MSB
```

---

changing the lower bits in a pixel does not impact much on its decimal equivalent value. To identify whether the block arrives for the first time in the memory or not we use the dirty bit. Dirty bit helps us to identify that the block is referred by CPU or not. If the block is referred by CPU, then we do not need to change the flag bit status. If the block is not referred by CPU, then we need to change the flag bit status. In a broad sense, we can say that we are using the flag bit at a byte level granularity [1].

The figure 1.1 explains how our algorithm works on the stored data. It shows that how original data is processed when reading the original data based on the status of flag bit corresponding to each pixel. In figure 1.1 we assume two things

1. Some arbitrary plane (let's say here 3 bits in LSB part and 5 bits in MSB part) to divide the pixel into MSB part and LSB part.
2. Some arbitrary value to set or reset the flag bit (let's say here if the number of ones in LSB part is more than or equal two then we set flag bit else reset the flag bit).

### 3.1 Modification of the proposed solution

The quality of service of our solution depends upon the PNSR value and SSIM. PNSR value is the expansion of the human visual system. By investigating the performance of the human optical system [2], we come to know the one limitation which is useful for us to improve the quality of service of our proposed solution, i.e., weak peripheral vision.

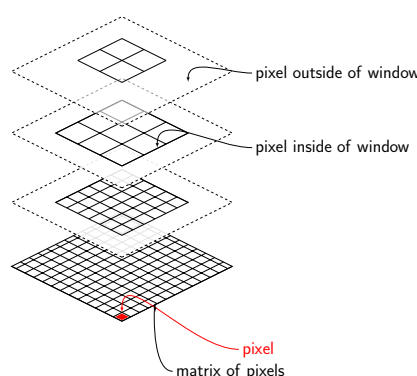


Figure 3.1: Growth of window dimensions

Peripheral vision or Side vision is the ability to see objects and movement outside of the direct line of sight. Peripheral vision is weak in humans (especially at distinguishing detail, color, and shape) because the density of receptor and ganglion cells in the retina is higher at the center and lowest at the edges. Human eye suffers from low peripheral vision and high center vision. The figure 1.2 explains how the human eye sees the standard image (as shown in figure 1.2 (a)). When it has the deficient peripheral vision, its focus on center part and for an outer part it does not discriminate between color and shape due to this it blurs peripheral part as shown in figure 2 (b).

In the algorithm 1, we approximate the entire image. However, when human eye sees a picture, it always focuses on the center of image compare to its peripheral area because the peripheral vision in the human eye is weak [3]. Using the above fact, we modified our proposed algorithm in such a way that always approximate the peripheral part of an image instead of the center part of the image. The center part of the image that is not approximated we call it the center window. The primary focus of human visual system on any image is the center of the image due to the high center vision and low peripheral vision.

The center window always formed in the center of the image and grows in length and width in both directions. There are many sizes possible for center window. Figure 3.1 shows the growth of center window on the 2-dimensional matrix of the pixel, from top to bottom we see that initially, the window contains limited pixels but as the window grows it covers the maximum portions matrix. If we approximate the pixels outside of center window using our proposed ApproxVision algorithm, then we can increase the quality of



the obtained image. Using the above modification, we are improving the quality of the image (or increasing the PSNR value). The above modification is the built on the limitation of the human visual system.

# Chapter 4

## Experiment

In the algorithm 1, two parameters are still unknown like how many bits in MSB part and LSB part (mention in step 2) and what is the number of ones to set the flag bit (mention in step 3). To complete the algorithm, we need to find these two parameters.

### 4.1 How to find the optimal partition and optimal threshold value

While dividing the pixel into MSB and LSB, we observed that there are many possible combinations exists like 3 bits in LSB and 5 bits in MSB, 4 bits in LSB and 4 bits in MSB, 5 bits in LSB and 3 bits in MSB, and so on. Each combination inserts some error in the original image. Finding the partition with least error, we called it as the Optimal partition value is the most challenging part because it is the heart of our proposed solution. To see the error in the partitions, initially we used mean squared error [12] but it is not enough for deep inspection of each partition because it is not very well match to perceived visual quality. To make our observations more reliable for finding the optimal partition we use two other metrics such as PSNR and SSIM [13]. These metrics ensure that the quality we get depends upon the human visual system.

After dividing the pixel into LSB and MSB, we need the count of the number of ones in LSB based on this count value flag bit is set or reset. There are many count values possible to set or reset the flag bit. If ' $m$ ' bits are there in LSB part, then there are ' $m$ ' possible count values exists, and each of them injects some error in the obtained image. The error in the processed images is introduced by both partition of MSB and LSB and the count value to set or reset the flag bit. Now our next challenge is to find the count value to set or reset flag bit corresponding to optimal partition for which the error is minimum, we called it as the optimal threshold value.

There are many possible combinations for MSB and LSB exists like 1 bit in LSB & 7 bits in MSB, 2 bits in LSB & 6 bits in MSB, 3 bits in LSB & 5 bits in MSB, and so on. Out of them, some combinations are not meaningful in the sense that the number of bits in LSB part is very less, e.g., 1 bit in LSB & 7 bits in MSB and 2 bits in LSB & 6 bits in MSB and our goal is to save the energy in LSB part. Due to this, we discard some of the combinations of MSB part and LSB part. The count of the number of ones in LSB part to set the flag bit is at least more than 50% of LSB part because it reduces the error rate in the obtained image and also it will simplify the math, without changing the result significantly. By considering that the count value is at least 50%, we discard some combinations of the count value. Thus, we have limited number of combinations of partitions and threshold value, i.e., only 14 combinations. Table 4.1 shows various combinations of partitions

Table 4.1: Combinations of partition and threshold value on one byte

Sr. No.	Combination Name
1	3 Bits in LSB Threshold is 1 Bits
2	3 Bits in LSB Threshold is 2 Bits
3	4 Bits in LSB Threshold is 2 Bits
4	4 Bits in LSB Threshold is 3 Bits
5	5 Bits in LSB Threshold is 2 Bits
6	5 Bits in LSB Threshold is 3 Bits
7	5 Bits in LSB Threshold is 4 Bits
8	6 Bits in LSB Threshold is 3 Bits
9	6 Bits in LSB Threshold is 4 Bits
10	6 Bits in LSB Threshold is 5 Bits
11	7 Bits in LSB Threshold is 3 Bits
12	7 Bits in LSB Threshold is 4 Bits
13	7 Bits in LSB Threshold is 5 Bits
14	7 Bits in LSB Threshold is 6 Bits

and threshold value on one byte (or pixel).

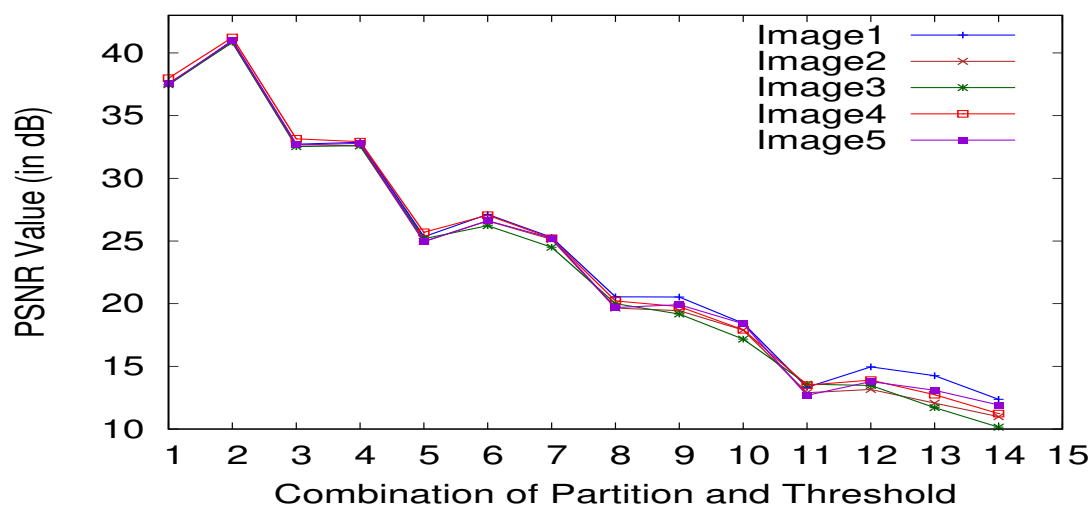


Figure 4.1: PSNR value for each combination of partition and threshold

To find the optimal partition of MSB part and LSB part and the optimal threshold value, we take images from a new General-100 dataset [16] that contains 100 bmp format images (with no compression). The size of the newly introduced 100 images ranges from  $710 \times 704$  (large) to  $131 \times 112$  (small). They are all of the good quality with clear edges but fewer smooth regions (e.g., sky and ocean), thus are very suitable for our experiment. Figure 4.1 shows various combinations of partitions and threshold value and their PSNR value on some images of General-100 dataset. Each point on the  $x$ -axis represents one combination of MSB and LSB part with different threshold values. Table 4.1 explains the combination corresponding to the point on the  $x$ -axis. Its serial number represents the point on the  $x$ -axis. The physical change in the quality of the image can be visualized in figure 4.2 for different combinations.

We see from figure 4.1 that the combination 5 bits in MSB and 3 Bits in LSB with threshold value more than or equal to 2 bit will give good PSNR value in all images. We also observe from figure 4.1 and figure 4.2



Figure 4.2: Image Quality for different partitions with different threshold value

that the count value to set or reset the flag bit has a significant impact on PSNR value. Thus, it also plays a vital role in our observations. The PSNR value in all images for the combination 5 bits in MSB and 3 Bits in LSB with threshold value more than or equal to 2 bit is magnificent than 34 dB which is our desired purpose because, beyond 34 dB of PSNR value, human image eye cannot distinguish between the original one and modified one. Thus, we can conclude that the optimal partition to divide the pixel into MSB and LSB is 5 bits in MSB and 3 bits in LSB, and the optimal threshold value is set or reset flag bit is when the number of ones in LSB is more than or equal to 2.

## 4.2 Approximate only on the peripheral part

In the above approach, we approximate the entire image. However, when human eye sees a picture, it always focuses on the center of image compare to its peripheral area because the peripheral vision in the human eye is weak. Using the above fact, we modified our proposed algorithm like always approximate the peripheral part of an image instead of the center part of the image. Now one obvious question arises:

*what is the size of the center part of the image?*

In the search for a suitable size of the center portion of the image. We define a quality of service on an image dimension with some general rules that apply to all images. We call center part of the image that is not approximate as the center window. Due to the weak peripheral vision in the human eye, we approximate entire image except for the center window of the image.

The window or center window always formed in the center of the image and grows in length and width in both directions as shown in figure 3.1. Thus the dimensions of the center window are the multiple of 4. There are many sizes possible for the center window. However, in the broad sense, there are only two variations possible for the center window these are maximum center window and minimum center window.

### Maximum Center Window:

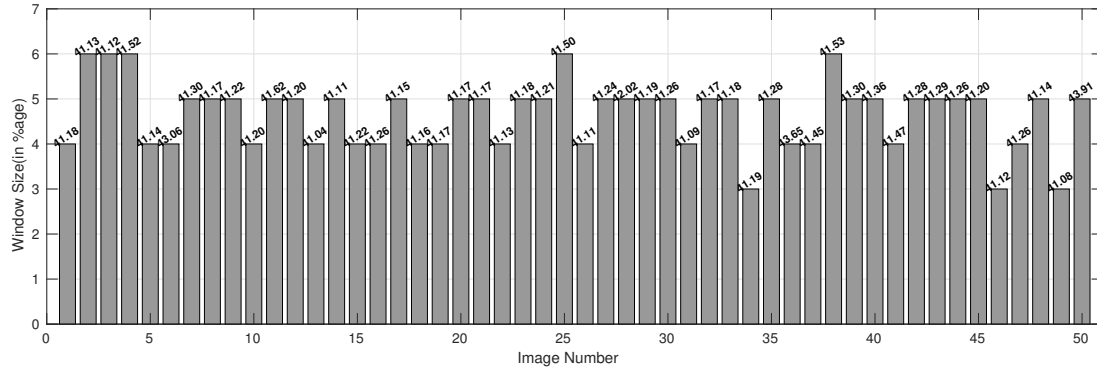
When the image dimensions are not same, i.e., heterogeneous, then the dimensions of the center window are nearly equal to the dimension of the image which is smaller. If the smaller dimension is not the multiple of 4, then the nearest multiple of 4 is considered and which is less than the smaller dimension of the image. Sometimes the center window covers the entire image if the dimensions of the image are same and are multiple of 4. That is the reason we called it the maximum center window.

### Minimum Center Window:

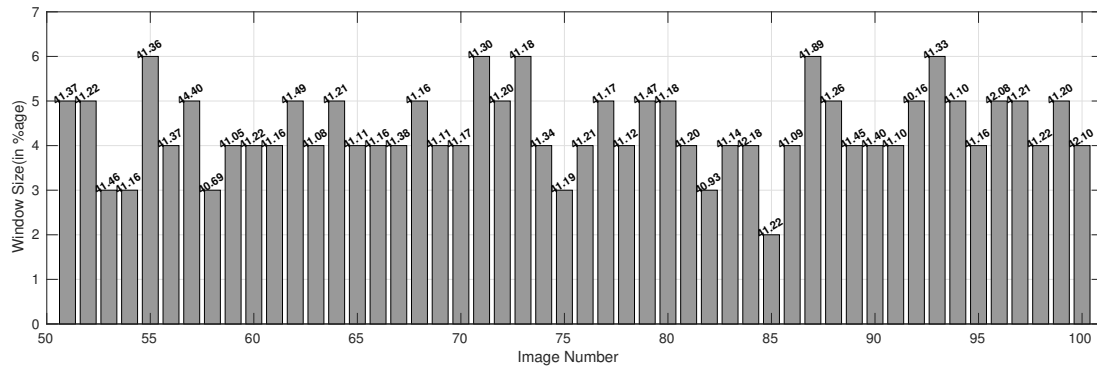
When the image dimensions are not same, then the center window is nearly equivalent to the smaller dimension of the image. By consider the size of the center window roughly equal to the size of the smaller dimension of the image and try to shrink the window size in the same way it grows and make it multiple of 4. Then the size of the center window becomes minimum. We call it the minimum center window. However, there are furthermore possibilities to shrink the window size but the size becomes tiny and center window contains very less number of pixels.

On our dataset, the size of the maximum center window varies from image to image and covers the maximum of 99% on some images and minimum of 67% on some images. And the size of the minimum center window covers the maximum of 7% on some images and minimum of 3% on some images. Thus, it is useless if we pick the maximum window size for our algorithm because sometimes it generates the same image with minor changes in some pixels. Therefore we are using the minimum center window approach for our algorithm.

After completing our algorithm with modifications, we are ready to run it on an entire General-100 dataset. Figure 4.3 (a) and 4.3 (b) shows the obtained PSNR value by using the minimum center window. The



(a)



(b)

Figure 4.3: ApproxVision with Minimum Center Window

value on the top of bar graph represents the obtained PSNR value, value on y-axis represent the minimum center window size and the value on x-axis represent the image serial number. We observe that the PSNR value in all images is more than 40 dB which is quite high for the human eye to distinguish between the quality of the original image and the obtained image (from our algorithm with modification).

### 4.3 Summary

Table 4.2 summarizes our experimental evaluations. The proposed algorithm (ApproxVision) with 3 bits in LSB and 5 bits in MSB as optimal partition (first row in Table 4.2) and greater than or equal to 2 bits in LSB as optimal threshold (second row in Table 4.2) and minimum center window (third row in Table 4.2), would be the best choice to achieve a high-quality image. The technique is useful regarding approximate storage also. However, regarding the quality of the image with more opportunity to approximate more pixels, the proposed algorithm with the minimum center window is the most excellent decision (last row in Table 4.2), which can cover 7% (maximum) portion of the entire image that is not altered.

Table 4.2: Summary of experimental evaluations

Experiment Name	Result
Partition of MSB and LSB (optimal)	3 Bits in LSB and 5 Bits in MSB
Threshold value in LSB (optimal)	$\geq 2$ Bits in LSB
Minimum Center Window (Size)	Minimum 2% and Maximum 6%
Maximum Center Window (Size)	Minimum 48% and Maximum 99%
Quality Obtained (PSNR value in dB)	Minimum 40.16 (dB) and Maximum 43.91 (dB)

# Chapter 5

## Conclusion and Future Work

### 5.1 Conclusion

In this paper, we proposed an algorithm (ApproxVision) for approximating the image to save energy without sacrificing the quality of the image. The usage is not limited to save energy, however it can also be useful for approximate storage of data. The proposed algorithm ApproxVision exploits the limitations of the human visual system. It uses the two limitations of the human visual system to approximate the image. First, lower bits in the pixel does not contribute much to the value of the pixel and second, human eye suffers from weak peripheral vision and strong center vision. The algorithm needs some parameters like the partition of MSB part and LSB part, threshold value in LSB part, and center window size for completion. These parameters can be determined by the heuristic way only with some general mathematics and probability.

The proposed algorithm ApproxVision with 3 bits in LSB and 5 bits in MSB as optimal partition (first row in Table 4.2) and greater than or equal to 2 bits in LSB as optimal threshold (second row in Table 4.2) and minimum center window (third row in Table 4.2), would be the best choice to achieve a high-quality image. Experimental results show that the maximum quality of image we get is 43.91 dB and minimum quality of image we get is 40.16 dB, both maximum and minimum quality we get is very high. All the pixels are changed except some pixels in the center part. The pixels that are not changed is almost 7% of the image.



Table 5.1: ApproxVision on Primitive Data-type

Data-type	Original Value	50% of MSB and 50% of LSB	75% of MSB and 25% of LSB
Float	6.880524635	6.875	6.880493164
Double	6.88052456664526	6.88052368164062	6.8805245666299

## 5.2 Future Work

The proposed algorithm ApproxVision is applied to image data only. The purpose of ApproxVision is to approximate the data and approximate storage of data for saving the energy. Some primitive data type requires large memory requirement like float and double, and also consume lots of energy while processing. These data types are used to store real value data. However, between two real values there exist infinite numbers of real values. Thus, these data type used some already existing approximation algorithm [17] to store the precision.

We can also approximate the value stored in the data type by using our proposed algorithm ApproxVision. But our approach also reduces some precision. To get the glance of how a float or double value is change by using the ApproxVision algorithm. To apply our algorithm we developed a pintool on top of PIN [18], a widely used dynamic binary instrumentation framework developed by Intel. The pintool models our ApproxVision algorithm on the basic data-types. Table 5.1 shows how original value is changed based on the different combinations of MSB part and LSB part (threshold is fixed, i.e., set flag bit if the number of ones is more than or equal to 50% of the LSB part). Here some combinations are left to explore. The validation of approximate value is application dependent. Applications that exhaustively use real value are the perfect candidate to check the validation of approximate value. By using our approximation algorithm (ApproxVision) in primitive data type, we can save energy for at least one byte which will be beneficial for applications which mostly use real values [19].

# References

- [1] J. Jung, Y. Nakata, M. Yoshimoto, and H. Kawaguchi. Energy-efficient Spin-Transfer Torque RAM cache exploiting additional all-zero-data flags. In *Quality Electronic Design (ISQED)*, 2013 14th International Symposium on. IEEE, 2013 216–222.
- [2] J. Smythies. A note on the concept of the visual field in neurology, psychology, and visual neuroscience. *Perception* 25, (1996) 369–371.
- [3] H. Strasburger, I. Rentschler, and M. Jüttner. Peripheral vision and pattern recognition: A review. *Journal of vision* 11, (2011) 13–13.
- [4] A. Sampson, J. Nelson, K. Strauss, and L. Ceze. Approximate storage in solid-state memories. *ACM Transactions on Computer Systems (TOCS)* 32, (2014) 9.
- [5] J. Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *Test Symposium (ETS)*, 2013 18th IEEE European. IEEE, 2013 1–6.
- [6] S. Mittal. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)* 48, (2016) 62.
- [7] H. Esmailzadeh, A. Sampson, L. Ceze, and D. Burger. Architecture support for disciplined approximate programming. In *ACM SIGPLAN Notices*, volume 47. ACM, 2012 301–312.
- [8] K. Cho, Y. Lee, Y. H. Oh, G.-c. Hwang, and J. W. Lee. eDRAM-based tiered-reliability memory with applications to low-power frame buffers. In *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM, 2014 333–338.
- [9] S. Ganapathy, G. Karakonstantis, A. Teman, and A. Burg. Mitigating the impact of faults in unreliable memories for error-resilient applications. In *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015 102.
- [10] A. Ranjan, S. Venkataramani, X. Fong, K. Roy, and A. Raghunathan. Approximate storage for energy efficient spintronic memories. In *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015 195.
- [11] H. Zhao, L. Xue, P. Chi, and J. Zhao. Approximate image storage with multi-level cell STT-MRAM main memory. In *Computer-Aided Design (ICCAD)*, 2017 IEEE/ACM International Conference on. IEEE, 2017 268–275.
- [12] Wikipedia contributors. Mean squared error — Wikipedia, The Free Encyclopedia. [https://en.wikipedia.org/w/index.php?title=Mean\\_squared\\_error&oldid=838879031](https://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=838879031).

- [13] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, (2004) 600–612.
- [14] X. Chen, L. Yang, R. P. Dick, L. Shang, and H. Lekatsas. C-pack: A high-performance microprocessor cache compression algorithm. *IEEE transactions on very large scale integration (VLSI) systems* 18, (2010) 1196–1208.
- [15] G. Pekhimenko, V. Seshadri, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry. Base-delta-immediate compression: practical data compression for on-chip caches. In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*. ACM, 2012 377–388.
- [16] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision*. Springer, 2016 391–407.
- [17] R. W. Mason and C. A. Heikes. Common format for encoding both single and double precision floating point numbers 1993. US Patent 5,268,855.
- [18] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. In *Acm sigplan notices*, volume 40. ACM, 2005 190–200.
- [19] S. Winkler. Perceptual distortion metric for digital color video. In *Human Vision and Electronic Imaging IV*, volume 3644. International Society for Optics and Photonics, 1999 175–185.