AN EFFICIENT ALGORITHM FOR DISPARITY MAP COMPRESSION BASED ON SPATIAL CORRELATIONS AND ITS LOW-COST HARDWARE ARCHITECTURE

A Thesis

by

Mustafa Ghanim

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Electrical and Electronics Engineering

Özyeğin University
December 2021

AN EFFICIENT ALGORITHM FOR DISPARITY MAP COMPRESSION BASED ON SPATIAL CORRELATIONS AND ITS LOW-COST HARDWARE ARCHITECTURE

Advisor: Prof. H. Fatih Uğurdağ
Co-advisor: Dr. Özgür Taşdizen

Approved by:

Prof. H. Fatih Uğurdağ, Advisor
Department of Electrical and Electronics Engineering
Özyeğin University

Prof. Tanju Erdem
Department of Electrical and Electronics Engineering
Özyeğin University

Asst. Prof. Onur Demir
Department of Computer Engineering
Yeditepe University

Date Approved: 27 December 2021

To my family, friends, colleagues, and instructors

ABSTRACT

This thesis proposes a low-cost Disparity Map (DM) compression algorithm and its hardware architecture for high resolution and high frame rate applications, where limited computational capacity and power consumption make it a significant challenge to run these applications in real-time. The proposed algorithm uses spatial correlations between neighboring disparity values and has two variants. The first variant, Single Neighbor (SN), encodes the current disparity using its left neighbor, whereas the second variant, Multiple Neighbor (MN), benefits from left and upper neighbors. The performance of the proposed algorithm is evaluated on DMs that are obtained using the most common low-cost stereo matching techniques for state of- the-art benchmarks. When evaluating the performance of the proposed compression algorithm, a median filter is applied to smooth out the DMs. After the median filter, the proposed algorithm with respect to original DM sizes, obtains 48% and 56% savings in memory space on the average for its SN and MN variants, respectively. Transferring DMs to the memory within the System-on-Chip (SoC) and transmitting them between SoCs in a compressed format will reduce the memory bandwidth usage and save power. Implementation results show that the proposed architecture can support 1920x1080 resolution at 60 fps on a low-cost FPGA while consuming very low area and power.

ÖZETÇE

Bu tezde, Eşitsizlik (Derinlik) Görüntülerini (DM) sıkıştırmak için düşük maliyetli bir algoritmanın yanısıra, yüksek çözünürlük ve yüksek kare hızı gerektiren ugyulamalara uygun bir donanım mimarisi önerilmektedir. Özellikle bu uygulamaların gerçek zamanda çalıştırılabilmesi, hesaplama kapasitesi ve güç tüketimi açısından büyük bir zorluktur. Önerilen algoritma, komşu piksellerin arasındaki uzamsal korelasyonlarını kullanır ve iki varyantı vardır. Tek Komşu (SN), sol komşusunu kullanarak mevcut eşitsizlik değerini kodlarken, ikinci varyant, Çoklu Komşu (MN), sol ve üst komşulardan yararlanır. Önerilen algoritmanın performansı, son model DM veri setine bağlı bir şekilde, yaygın düşük maliyetli stereo eşleştirme teknikleri kullanılarak elde edilen DM'ler üzerinde değerlendirildi. Önerilen sıkıştırma algoritmasının performansını değerlendirirken, DM'leri aşırı gürültüden temizlemek için bir medyan filtresi uygulanır. Medyan filtresinden sonra önerilen algoritma, SN ve MN varyantları için sırasıyla ortalama olarak bellek alanında %48 ve %56 tasarruf elde etmektedir. DM'leri Yonga-üzeri-Sistem (SoC) içindeki belleğe aktarmak ve bunları SoC'ler arasında sıkıştırılmış bir biçimde iletmek, bellek bant genişliği kullanımını azaltacak ve güç tasarrufu sağlayacaktır.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. Özgür Taşdizen of ARM, UK for his effective co-supervising of my MS thesis and submitted manuscript (which is under review) with contributions that include: methodology, conceptualization, review, and editing. Furthermore, I would like to thank him for his patience with my pace in this project, especially during the design verification phase.

Dr. Özgür Taşdizen also spent a decent amount of time after working hours of his primary job at ARM (as a Principal GPU Architect) to schedule/appear on online meetings and communicate with me through email.

I would also like to express my sincere thanks to my supervisor Prof. H. Fatih Uğurdağ at Özyeğin University, who has been helping me in many ways since I joined Özyeğin University. Prof. H. Fatih Uğurdağ has helped me in getting accepted to the EE Dept. as a full-time teaching assistant with two scholarship packages, namely, teaching assistant stipend and fellowship.

Moreover, he was a great instructor in the hardware courses that I took: Digital Electronics & FPGA Design (EE562) and System on Chip Design (EE566). These courses have strengthened my Verilog RTL design and verification skills at a professional scale and also helped me complete the hardware design aspects of this thesis properly.

To give a simple example, Prof. Uğurdağ has taught me how to make RTL designs pass synthesis and implementation successfully without errors and dangerous warnings (such as transparent latches) in a systematic way, regardless of their complexities. Prof. Uğurdağ has also encouraged me to take two undergraduate Computer Science courses which are: Data Structures and Analysis of Algorithms. By taking these courses, I could expand my knowledge and skills of solving problems efficiently, developing new algorithms, and programming with

high-level languages such as C, C++, and Python.

I am also so grateful to my other colleagues in Özyeğin University and other professors as well. I was lucky to attend their courses, where an environment of high-quality engineering education always exists. I would like to mention two courses especially, which were useful for me while being introduced to the area of depth compression, Digital Image Processing by Prof. Tanju Erdem and Computer Vision by Asst. Prof. Furkan Kıraç.

Last but not least, I would like to thank all my family members, namely, my father, my mother, my brothers, and my sisters for their endless support to me during all of my studies in Turkey in the last seven years and specifically in the last three years as a graduate student at Özyeğin University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS:

ASIC Application Specific Integrated Circuits *p.* 35

AXI Advanced eXtensible Interface *p.* 30

BP Belief Propagation *p.* 4

BPP Bits-per-pixel *p.* 13

BRAM Block Random-access Memory *p.* 34

CRT Complete Rank Transform *p.* 3

CT Census Transform *p.* 3

DM Disparity/Depth Map *p.* 1

DRAM Dynamic Random-access Memory *p.* 1

EDA Electronic Design Automation *p.* 35

FF Flip-Flop *p.* 38

FIFO First In First Out *p.* 34

FPGA Field-programmable Gate Array *p.* 42

HDL Hardware Description Language *p.* 35

HSYNC Horizontal Synchronization Signal *p.* 29

I/O Input/Output *p.* 38

IP Intellectual Property *p.* 34

ISE Integrated Synthesis Environment *p.* 34

JPEG-LS Lossless/Near-lossless JPEG Algorithm *p.* 7

JPEG-LS MPO JPEG-LS Modified Predictor Only *p.* 14

LSB Least-significant Bit *p.* 32

MN Multiple Neighbor Compression Algorithm *p.* 9

MSB Most-significant Bit *p.* 32

MV-HEVC Multi-view High Efficiency Video Coding *p.* 5

PSNR Peak Signal-to-noise Ratio *p.* 5

RLE Run-length Encoding *p.* 18

RMSE Root-mean-square Error *p.* 57

RT Rank Transform *p.* 3

RTL Register-transfer Level *p.* 28

SAD Sum of Absolute Differences *p.* 4

SAIF Switching Activity Interchange Format *p.* 42

SGM Semi-global Matching *p.* 3

SN Single Neighbor Compression Algorithm *p.* 9

SoC System-on-Chip *p.* 1

TNS Total Negative Slack *p.* 40

VSYNC Vertical Synchronization Signal *p.* 29

WNS Worst Negative Slack *p.* 40

CHAPTER I

INTRODUCTION

This chapter explains the purpose of this thesis in detail. It gives the basics of disparity and depth concepts, most common disparity estimation methods, and preprocessing transforms. Furthermore, it includes the state-of-the-art works in the literature that are related to depth compression and their hardware implementations where available.

The works presented in Chapters 1 and 2, Sections 3.1 and 3.3, Chapter 4, and Section 5.3 were included in our journal submission [1].

1.1 Problem description

This work aims to introduce an area and power-efficient algorithm as well as its hardware architecture for Disparity Map (DM) compression. Such architectures are required when processing real-time, high resolution, and high frame rate video signals with limited hardware resources. Moreover, in today's System-on-Chips (SoCs) low power consumption is critical due to thermal constraints and battery life of mobile devices. Accessing DRAM may consume a significant portion of the total power in mobile SoCs. Thus, the proposed algorithm and its hardware architecture aim to reduce the total power consumption by decreasing the memory bandwidth. Depth information is essential for many computer vision applications such as augmented reality, 3D mapping, 3D movies, video surveillance, military applications, robotics, and self-driving vehicles. Depth information can be defined as the distance of the surfaces of scene objects to the viewpoint. The terms depth and disparity are related to each other. Binocular disparity describes the shift amount in coordinates of the same object between a stereo image pair [2].

The depth value depends on the disparity value and the camera calibration. Depth information can be estimated by imitating the human visual system with a

stereo camera pair. The third-dimension coordinate of an object can be extracted by using left and right images of the stereo pair, where one image will be the shifted version of the other. Disparity estimation is a process very similar to motion estimation, which is widely used in video compression [3]. Unlike the block matching for video compression, which is applied within a two-dimensional search window [4], block matching for disparity estimation implements a single dimensional search window along the horizontal axis.

In this work, horizontally rectified stereo image pairs are utilized as input for disparity estimation; i.e., it is assumed that considering disparities at x-axis only is sufficient to correctly estimate the output DM. Figure 1 shows the typical structure of a binocular stereo vision system, where the corresponding disparities of Point-1 and Point-2 are equal to:

$$Disparity(Point\text{-}1) = xL1 - xR1 \tag{1}$$

$$Disparity(Point\text{-}2) = xL2 - xR2 \tag{2}$$

Where $xL1$ and $xR1$ represent projection of Point-1 on the x-axis of left and right camera focal planes, respectively. Likewise, $xL2$ and $xR2$ represent projections of Point-2 on the x-axis. Because Point-1 is closer to the focal planes of the stereo camera pair, its disparity value will be larger when compared to the disparity value of Point-2. Depth values of Point-1 and Point-2 can be found using the following equations:

$$Depth(Point\text{-}1) = \frac{Baseline\,.\,Focal\ Length}{xL1 - xR1} \tag{3}$$

$$Depth(Point\text{-}2) = \frac{Baseline\,.\,Focal\ Length}{xL2 - xR2} \tag{4}$$

We can define the baseline as the distance between stereo camera pair lenses, whereas focal length of the lens is the distance between the lens and the image sensor when the subject is in focus.

It is important to preprocess stereo image pairs before the disparity estimation due to right and left images being exposed to different noise and brightness levels.

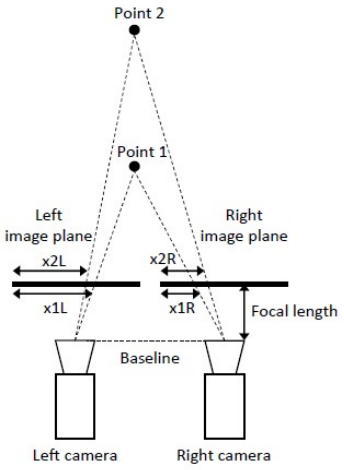


Figure 1: Binocular stereo vision system.

There are transforms presented in the literature to increase the tolerance against these negative effects.

These transforms are applied before the disparity estimation, and this work focuses on three widely used transforms in the literature: Rank Transform (RT), Census Transform (CT), and Complete Rank Transform (CRT).

RT applies a filtering window, where the center value of the filtering window gets replaced by the total number of samples within the filtering window that are smaller than the center value [5]. CT outputs a string of $(N.M-1)$ bits, where N and M are the width and the height of CT's filtering window. In CT, window locations that have smaller or equal gray-scale intensity values than the center value are flagged as '0' and remaining locations with larger intensity are flagged as '1'. Thus, the output of CT is a bit string. CRT [6] is a modified version of the RT. Unlike CT and RT, which output a single value for the center location of the filtering window, CRT outputs a window for each input disparity value. CRT is morphologically invariant; meaning that the output image is invariant to any monotonically increase in its gray-scale intensity. Figure 2 illustrates a 3 x 3-windowing example for these three transforms. Other than the local block matching methods, there are semi-global and global methods proposed in the literature for disparity estimation. Complexities of these methods increase from local methods to global methods. Semi-Global Matching (SGM) [7], global algorithms

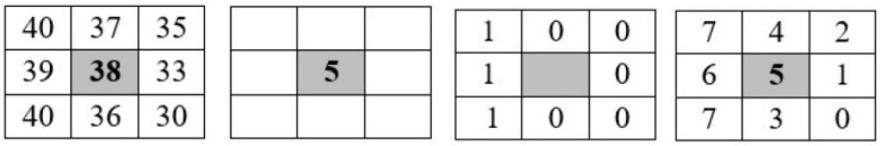


Figure 2: Preprocessing transforms applied to stereo pairs. (a) 3 x 3 Intensity window. (b) RT. (c) CT. (d) CRT.

such as Maximum Likelihood Stereo [8], Belief Propagation (BP) [9] produce more accurate DMs than the local methods. The proposed compression algorithm can be applied to DMs obtained by any disparity estimation method.

In this thesis the performance of the proposed algorithm is evaluated on DMs that are generated by local methods. Because the proposed compression method is claimed to be low-cost, it is expected to work in conjunction with a low-cost method such as local block matching.

Local block matching algorithms divide the input frame into rectangular blocks and search for similar blocks in the reference frame. Each block from one frame (left or right) is searched in the other frame within given the search window. The block in the search window giving the minimum error value is chosen as the match. A preprocessing transform is applied to input stereo image pairs to increase the accuracy of the matching process. The error criteria in this search process can vary based on the type of the preprocessing transform. RT and CRT apply Sum of Absolute Differences (SAD). Depending on the input image resolution, increasing the block size generally leads to a better prediction accuracy, but it also increases the hardware complexity. Eq. (5) shows the SAD metric:

$$SAD = \sum_{i=1}^{M} \sum_{j=1}^{N} |I_R(i,j) - I_L(i,j)| \tag{5}$$

In this equation, I_R, I_L, i, j, M, and N represent intensity values for right and left grayscale stereo images, pixel coordinates, and maximum search window limits on x and y axes, respectively. CT uses Hamming distance for calculating the error between two bitstrings obtained from left and right images. As shown in Figure 3, which demonstrates the calculation of the Hamming distance between

two example bit strings, using Hamming distance as the error criteria requires an XOR operation between input bitstrings followed an accumulation operation.

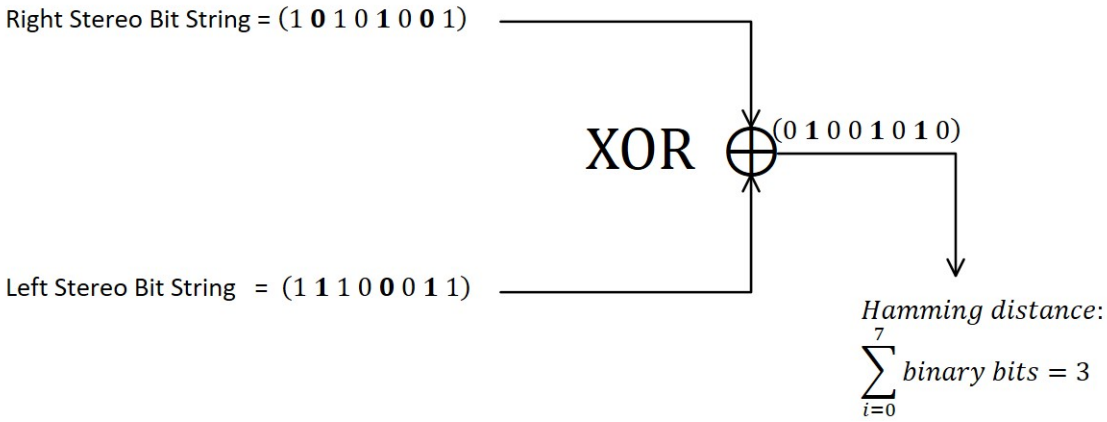


Figure 3: Calculation of Hamming distance between two bit strings of 8-bit wide.

1.2 Literature review

Accessing temporal data when processing high resolution video streams leads to increased DRAM bandwidth, which is becoming the main contributor to the total dynamic power consumption of an SoC. Thus, it is crucial to compress DMs for real-time applications requiring the disparity (or depth) data of the previous frame(s) on a thermally or battery constrained device.

A wide range of papers have been reported in the literature about Multi-View extension of High Efficiency Video Coding (MV-HEVC) and 3D-HEVC [10]. These papers generally present a trade-off between coding efficiency and computational complexity [11–13]. For example, in [11], computational complexity of 3D-HEVC is significantly reduced with a slight decrease in the coding efficiency. This is accomplished by applying static decision trees, which were built with data mining and machine learning. Likewise, in [12] early termination mode decision, adaptive search range motion estimation, and fast disparity estimation approaches are presented to reduce the complexity of 3D-HEVC by compromising a small loss in PSNR and a small increase in the bitrate. In [13], data analysis and clustering based approach is used to skip depth modeling modes in 3D-HEVC where possible. HEVC can obtain much higher compression rates, but it is very costly to

implement in hardware even though there can be some optimizations to reduce its computational complexity as proposed in [11–13].

There are less complex algorithms proposed in the literature targeting disparity/depth compression [14, 15]. These are lossy compression algorithms obtaining higher compression results when compared to the proposed lossless algorithm in this thesis. Their computational complexity is lower when compared to HEVC but still higher when compared to the proposed algorithm in this thesis. These papers on disparity/depth compression do not consider the target platform running the algorithm. Therefore, a direct comparison with the proposed hardware architecture cannot be made.

In addition, there are some algorithms that use multiple neighbor pixel information and sparse analysis for DM coding. In [16], a depth-coding algorithm is demonstrated by modeling each image block as a graph. Moreover, a 4-neighbor connectivity is considered for each pixel at most. As for [17], sparse predication is used; estimated and residual disparities are encoded for the left image-based DM. The estimated value depends on so called regression vectors which contain; top, top-left, left, down-left, down, down-right, right, and top-right neighbors in a 9-long vector for the full template form, and top, left, top-left, and top-right neighbors for the causal template form. These two forms are used according to the pre-determined conditions. Although that these two works used neighboring disparity relationships for depth compression, they are lossy methods and more complicated due to higher number of complex arithmetic operations included. To illustrate further, graph-based algorithm cost function includes taking logarithm, division, and six multiplication operations which are done iteratively for a series of adjacency matrices. The sparse prediction-based method in [17], the encoder needs one multiplication operation per pixel. Furthermore, the sparse predictor calculates the least squares which requires one subtraction, one multiplication, and an ordering operation for each corresponding regression vector. As a result,

finding the disparity value of a single pixel in [17] requires 16 additions, 20 subtractions, and 4 divisions due to gradient context calculations. This requirement results with a more complex hardware implementation consuming larger silicon are when compared with the proposed hardware architecture in this paper.

There are papers proposing algorithms where one of the images in the stereo pair is predicted from the other image [17], and right-view DM is predicted from the left-view DM [18]. This operation is not the same as disparity/depth compression. As a result, it can be said that to the best of available knowledge there is no lossless and efficient DM compression algorithm proposed in the literature with its hardware architecture. Therefore, it is decided to compare both the compression performance and hardware implementation results of the proposed algorithm against LOw COmplexity LOssless COmpression (LOCO-I) algorithm.

LOCO-I is the core of JPEG-LS [19] image compression standard. JPEG-LS has a lossless compression mode which can be tested for disparity compression. JPEG-LS algorithm consists of predictor, context modeling, and adaptive error correction. Moreover, it has two compression modes: run mode that uses a run coder, and regular mode that uses Golomb coder. Thus, its computational complexity is higher than the proposed algorithm. The predictor part of LOCO-I/JPEG-LS is called as the median edge detector, which uses left, top, and top-left neighbors of the current disparity for detecting horizontal or vertical edges and predicting the current disparity value. The predicted median edge value, $\tilde{d}(x,y)$ is shown in Eq. (6).

$$\tilde{d}(x,y) = \begin{cases} [d(x-1,y), d(x,y-1)]_{\min}, \\ \textit{if } d(x-1,y-1) \geq [d(x-1,y), d(x,y-1)]_{\max} \\ [d(x-1,y), d(x,y-1)]_{\max}, \\ \textit{if } d(x-1,y-1) \leq [d(x-1,y), d(x,y-1)]_{\min} \\ d(x-1,y) + d(x,y-1) - d(x-1,y-1), \ \textit{otherwise} \end{cases} \tag{6}$$

In [20], a raster-scan order based encoder and decoder couple is proposed. The proposed compression approach is based on prediction and residual coding like

JPEG-LS and supports near-lossless and lossless modes. A fully pipelined architecture for lossless JPEG-LS encoder is implemented in [21]. In [22], hardware implementation of a real-time video transmission system based on hierarchical prediction followed by variable length coding is presented. As discussed in Chapter 5.3, implementing the proposed low-cost algorithm proposed encoder and decoder hardware architectures in this thesis consume a smaller silicon area and support higher clock frequencies when compared with hardware implementation results reported in [20–22].

The rest of this thesis is organized as follows. Chapter 2 explains the proposed algorithm, and Chapter 3 presents the compression results. Chapter 4 describes encoder and decoder hardware architectures implementing the proposed algorithm, and Chapter 5 presents the hardware implementation results. Chapter 6 concludes this thesis.

CHAPTER II

PROPOSED ALGORITHM

Advantages of compressing DMs for mobile devices operating in real time motivated us to research an efficient algorithm targeting low-cost hardware implementation. In this research, we focused on correlations between neighboring disparity values. Benefiting from the small variance in neighboring disparity values, especially if they belong to a background object in the scene, we proposed a lossless compression algorithm with two variants: Single Neighbor (SN) and Multiple-Neighbor (MN). SN method uses only the left neighbor. whereas MN uses the left neighbor and neighbors from upper line, which requires a line buffer in the hardware implementation. When processing DMs, a 3 x 3 median filter is widely used in academia and industry to eliminate the outliers. Therefore, we evaluated the performance of both methods with and without a median filter as a preprocessing step. This chapter clarifies all of SN, MN, and median filter use in this work.

2.1 Disparity compression based on single neighbor

SN method is defined in Eq. (7), where $d(x, y)$ is the current disparity value, $d(x-1, y)$ is the left neighbor's disparity value, Δ is the difference between them, δ is the compression threshold, and $\tilde{d}(x, y)$ is the encoded disparity value. If the absolute difference between neighboring values is smaller than or equal to δ, the current disparity value gets compressed by encoding the difference value to the bitstream else the current disparity value is written to the bitstream.

$$\Delta = d(x, y) - d(x-1, y)$$
$$\tilde{d}(x, y) = \begin{cases} \Delta, \text{if } |\Delta| \leq \delta \\ d(x, y), \ \textit{otherwise} \end{cases} \tag{7}$$

We analyzed the compression performance using fixed length and merged header approaches. In the fixed length header approach, SN method includes a single bit of header for each disparity value encoded in the bitstream. If this header bit indicating to the disparity state is set as uncompressed, then the decoder reads the next N bits from the payload as the disparity value. Else if the disparity state is set as compressed, then the decoder reads the Δ from the payload. If the compression threshold is not zero, then the sign bit of Δ is included in the payload. In the merged header approach, there is no separate bit showing the compressed/uncompressed state. The header and payload are merged and encoded together. For example, when δ is 1, all the available conditions are encoded with 2 bits: "uncompressed, compressed and equal, compressed with +1 difference, and compressed with -1 difference". Table 1 shows the required number of bits to encode a disparity value depending on the compression state and various compression thresholds.

Table 1: Number of bits in the bitstream for SN method.

Compression state	δ	Number of bits in fixed length header	Number of bits in merged header
Uncompressed	0	$N+1$	$N+1$
Compressed	0	1	1
Uncompressed	1	$N+1$	$N+2$
Compressed	1	3	2
Uncompressed	3	$N+1$	$N+3$
Compressed	3	4	3

2.2 Disparity compression based on multiple neighbors

MN method uses the fixed length header approach. MN method uses 2 bits in the fixed length header to select between three neighboring values and the original disparity value. These neighboring values, ΔL, ΔT, and ΔTR represent the difference of the current disparity to its left, top, and top-right neighbors and shown in Eq. (8), Eq. (9) and Eq. (10), respectively. Selection process of the MN

method is shown in Eq. (11).

$$\Delta L = d(x, y) - d(x-1, y) \tag{8}$$

$$\Delta T = d(x, y) - d(x, y-1) \tag{9}$$

$$\Delta TR = d(x, y) - d(x+1, y-1) \tag{10}$$

$$\tilde{d}(x, y) = \begin{cases} \Delta\mathrm{L}, \ \textit{if} \ |\Delta\mathrm{L}| \leq \delta \\ \textit{else} \ \Delta\mathrm{T}, \ \textit{if} \ |\Delta\mathrm{T}| \leq \delta \\ \textit{else} \ \Delta\mathrm{TR}, \ \textit{if} \ |\Delta\mathrm{TR}| \leq \delta \\ d(x, y), \ \textit{otherwise} \end{cases} \tag{11}$$

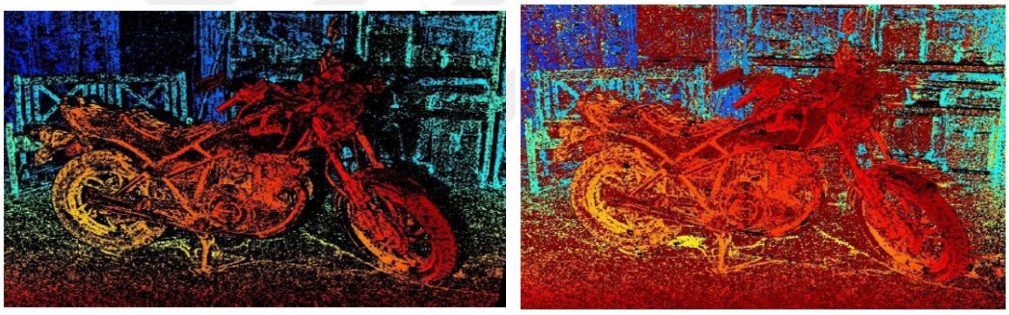
In the bitstream, fixed length header information is followed by the payload, which consist of the sign bit and magnitude bits when $\delta > 0$. Table 2 shows the required number of bits in the bitstream for various compression thresholds. If the N-bit disparity value gets compressed by the MN method, its size in the bitstream reduces to the total number of header and payload bits, which is 2, 4, and 5 bits for $\delta = 0$, $\delta = 1$, and $\delta = 3$, respectively.

Table 2: Number of bits in the bitstream for MN method.

Compression state	δ	Number of bits
Uncompressed	0	$N + 2$
Compressed	0	2
Uncompressed	1	$N + 4$
Compressed	1	4
Uncompressed	3	$N + 5$
Compressed	3	5

2.3 Median filter

Block matching disparity estimation process is prone to errors resulting with outlier disparity values. Using a median filter, these outliers can be detected and replaced by the median value of the filtering window. Therefore, a 3 x 3 median filtering is a commonly used to enhance the quality of DMs [23]. Taking this state-of-the art processing step into account, we have smoothed DMs with a 3 x 3 median filter before compressing them with the proposed algorithm. Figure 4(a) shows a block matching-based DM in jet colormap array format, where warm colors represent higher disparities. The black pixels in the DM represent the incorrectly estimated disparity locations as if the disparity values were zero in these locations. As it can be seen in Figure 4(b), application of a 3 x 3 median filter significantly decreases these outliers and generates a better DM.



(a) (b)

Figure 4: Application of a median filter on a DM. (a) Orginal DM. (b) Median filtered DM.

CHAPTER III

ALGORITHM RESULTS

This chapter presents the experimental results for the proposed compression methods and compares them with JPEG-LS. Moreover, it includes adaptive versions of the proposed algorithms, quantization effect on DM compression, and compression performance on high-resolution RGB and grayscale images dataset.

3.1 SN and MN compression methods

We have evaluated performance of proposed methods using the rectified Middlebury 2014 dataset [24]. Figure 5 shows 10 benchmark stereo image pairs from [24] used in our simulations and obtained space savings for each of these benchmarks for various compression thresholds with and without the median filter. The resolutions of the benchmark stereo image pairs range from 1848 x 2724 to 2000 x 2964. DMs are obtained from these stereo images using block matching after applying common transforms (CT, RT, CRT) for various window sizes. Results presented in Figure 5 show the average space savings (in percentage) of individual simulations for DMs, which are obtained after applying these three transforms. Space savings ratio presented in this section is calculated as follows:

$$Space\ Savings\ = 1 - \frac{Compressed\ Data\ Size}{Uncompressed\ Data\ Size} \tag{12}$$

There are also different metrics to show compression performance in the literature such as, bits-per-pixel (BPP) and compression ratio which can be easily related to space savings as shown below:

$$Compression\ Ratio = \frac{Uncompressed\ Data\ Size}{Compressed\ Data\ Size} \tag{13}$$

$$BPP = \frac{N}{Compression\ Ratio} = N \cdot (1 - Space\ Savings) \tag{14}$$

Where N is the total number of bits allocated for one original disparity/pixel value without compression. For example, when $N = 8$ and total frame size without compression is 5 MB, after compression it is 1 MB. We find space savings, compression ratio, and BPP are equal to: 0.8 (or 80%), 5 (or 5:1), and 1.6 bits per pixel respectively.

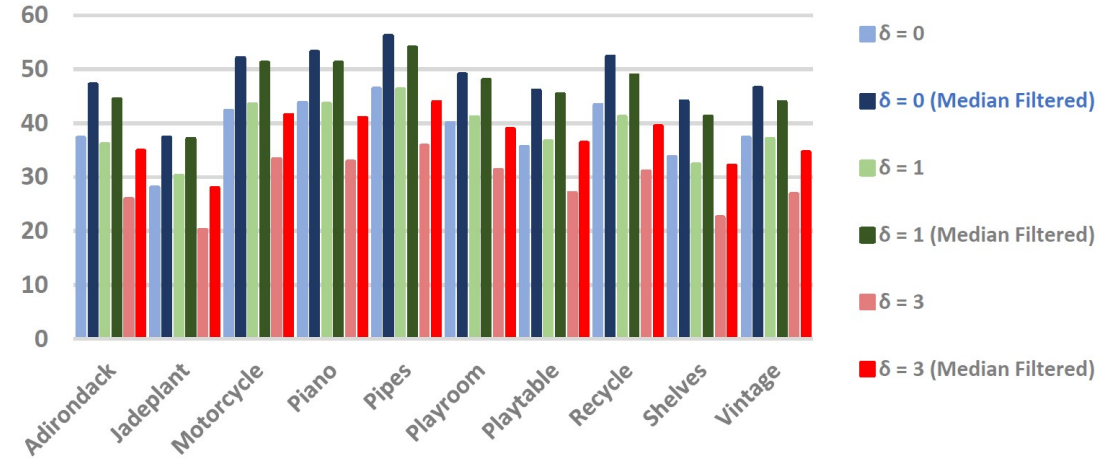
Table 3: Details of the used dataset for DMs.

DM name	Resolution (width x height)	Disparity range of its ground truth
Adirondack	2880 x 1988	0 - 243
Jadeplant	2632 x 1988	0 - 588
Motorcycle	2964 x 2000	0 - 240
Piano	2820 x 1920	0 - 217
Pipes	2960 x 1924	0 - 271
Playroom	2800 x 1908	0 - 294
Playtable	2724 x 1848	0 - 265
Recycle	2864 x 1924	0 - 221
Shelves	2952 x 2000	0 - 201
Vintage	2912 x 1924	0 - 696

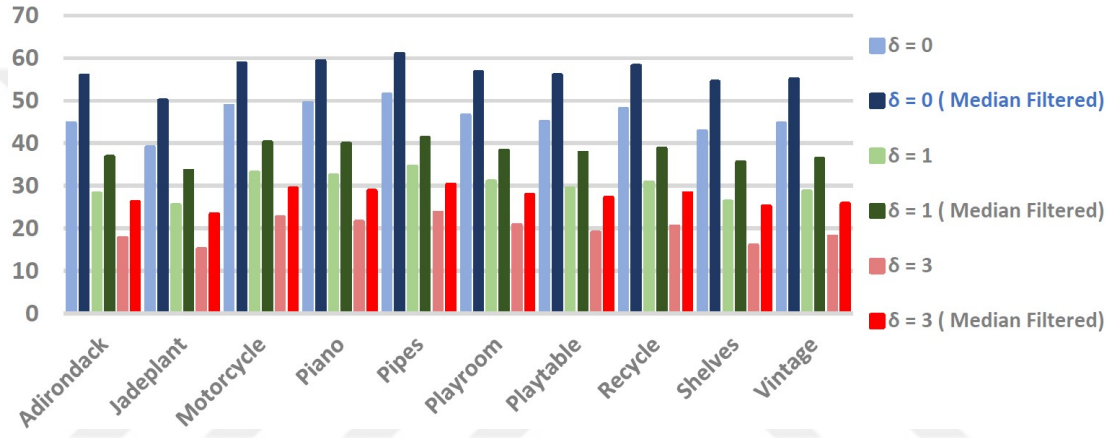
When evaluating the performance of the proposed compression methods, we also analyzed the performance of JPEG-LS using [25]. We compressed DMs running the JPEG-LS in its lossless mode, and we also compared compression results obtained by its predictor only. We converted the fixed-predictor of JPEG-LS into a lossless encoder that is very similar to the proposed MN method. We called this encoder JPEG-LS Modified Predictor Only (JPEG-LS MPO). The only difference between the proposed MN method and the JPEG-LS MPO is that the proposed method can use the top-right neighbor's value instead of the predicted value calculated by the JPEG-LS MPO as shown in Eq. (15).

$$\tilde{d}(x,y) = \begin{cases} d(x-1,y) + d(d,y-1) - d(x-1,y-1), \ \ if \ : \\ |d(x-1,y) + d(x,y-1) - d(x-1,y-1)| \leq \delta \\ d(x,y), \ \ otherwise \end{cases} \tag{15}$$

Table 4 shows space savings obtained by the proposed SN and MN methods, JPEG-LS MPO, and lossless mode of JPEG-LS for different transform types, block matching sizes, and various δ values with and without applying a 3 x 3 median



(a)



(b)

Figure 5: Space savings in [%] (a) SN results. (b) MN results.

filter. Results presented for SN method are based on the merged header approach, which performs slightly better than the fixed header approach. As expected, reducing the variance between neighboring disparity values by applying a median filter improves the compression performance. MN method, which requires a line buffer in its hardware implementation, outperforms SN method. Selecting $\delta = 0$ obtains the highest space savings for the MN method. In this table, we also see that the proposed MN method outperforms JPEG-LS MPO, which also requires a line buffer in its hardware implementation. Although, its hardware implementation would be much more complex, the lossless mode of JPEG-LS obtains the least space savings. This is because JPEG-LS is developed for image compression, and it is not fit for purpose when compressing DMs.

We have also evaluated the performance of the proposed methods on the ground truth DMs included in [24]. The same benchmarks listed in Figure 5 are used for this evaluation. Converting the 32-bit floating-point ground truth DMs to a fixed-point representation with different number of fractional bits, we have prepared Table 5. The fixed-point representation includes 9 bits for the integer part except for "Jadeplant" and "Vintage" benchmarks, which require 10 bits for the integer part and for the "Shelves" benchmark, which requires 8 bits for the integer part. Increasing the total number of bits to represent the disparity value by increasing the bits spent for the fractional part reduces the chance of neighboring disparity values to be identical and leads to smaller space savings. This table shows that the proposed MN method obtains better results when compared with JPEG-LS MPO. Therefore, we can conclude that independent of the quality of the input DMs, either obtained by low-cost local block matching algorithms or by more complex algorithms, which could approximate near ground truth quality, proposed MN method outperforms JPEG-LS MPO. Because ground truth DMs represent the exact disparity values, they are free from outliers. As a result, ground truth DMs are suitable for image compression algorithms and in this comparison the lossless mode of JPEG-LS encoder obtains best results. In Table 5, the fractional part of the floating disparity is quantized in uniform levels between zero and $max(2^{-b})$, where b represents the number of allocated bits for the subpixel precision. For instance, when $f = 2$ the possible fractions of a subpixel disparity are $i.00$, $i.25$, $i.50$, $i.75$, where i represents the integer part.

Table 4: Space savings results (%) for local block matching based DMs.

Pre-process transform and block matching size	SN		MN		JPEG-LS MPO		Lossless JPEG-LS	
	$\delta = 0$		$\delta = 0$		$\delta = 0$			
	$\delta = 1$		$\delta = 1$		$\delta = 1$			
	$\delta = 3$		$\delta = 3$		$\delta = 3$			
	Without median	With median	Without median	With median	Without median	With median	Without median	With median
7 x 7 RT + 5x5 block	29.82	41.16	40.29	51.81	37.42	47.03	7.52	39.54
	28.61	38.50	24.24	33.57	20.62	29.22		
	18.36	28.94	13.76	23.08	10.16	19.12		
7 x 7 RT + 7x7 block	40.17	49.58	50.60	58.39	47.67	54.10	24.32	49.39
	39.64	46.98	33.24	39.11	30.04	35.43		
	29.37	37.02	22.28	28.02	19.18	24.66		
7 x 7 CRT + 5x5 block	56.14	58.63	64.00	65.43	60.60	61.30	50.97	58.06
	54.25	56.11	43.46	44.50	40.66	41.34		
	43.89	45.96	31.81	32.89	29.22	30.06		
7 x 7 CRT + 7x7 block	61.02	62.78	66.81	67.71	63.69	64.01	58.77	63.40
	58.19	59.46	45.30	45.94	42.93	43.28		
	47.59	48.99	33.44	34.11	31.28	31.73		
7 x 7 CT + 1x1 block	1.93	25.47	5.98	37.11	5.90	33.29	-16.55	29.53
	9.38	27.95	2.00	24.73	-2.30	20.60		
	0.77	21.36	-5.14	16.99	-9.77	10.10		
Average space savings	**37.81**	**47.52**	**45.55**	**56.10**	**43.06**	**51.95**	**25.01**	**47.98**
	38.01	**45.80**	**29.65**	**37.58**	**26.39**	**33.97**		
	27.99	**36.45**	**19.24**	**27.03**	**16.02**	**23.77**		

Table 5: Space savings results (%) for ground truth DMs.

Subpixel precision (bits)	SN ($\delta = 0$)	MN ($\delta = 0$)	JPEG-LS MPO ($\delta = 0$)	Lossless JPEG-LS	PSNR (dB)
0	80.95	75.69	76.39	88.95	59.05
1	76.41	77.61	75.95	87.16	65.72
2	67.14	77.11	74.87	85.02	71.23
3	51.29	72.53	71.70	83.03	77.77

3.2 Run-length analysis for SN algorithm

Run-length Encoding (RLE) analysis is made for the proposed SN method to evaluate its performance. RLE has a simple compression idea of serial and sequential data by encoding it as the number of consecutive equal (compressible) values. Eq. (16) explains the RLE algorithm with sequential symbols (x).

$$\begin{aligned} &\textit{Sequential Data}: \ x_1, x_2, x_2, x_2, x_2, x_2, x_2, x_2, x_3, x_3, x_4, x_4, x_4, x_4, x_5, x_5, x_6 \\ &\textit{Encoded Data}: \ \mathbf{1}, x_1, \mathbf{7}, x_2, \mathbf{2}, x_3, \mathbf{4}, x_4, \mathbf{2}, x_5, \mathbf{1}, x_6 \end{aligned} \tag{16}$$

In Eq. (16), the bold text represents the count of each equal data consecutively. Thus for a 8-bit data and assuming that generally the count will not exceed 255 (if such rare case occurs, lossless compression can be forced be lossy). The space savings for RLE in the previous example is equal to:

$$\textit{Space Savings } (RLE) = (1 - \frac{12 \cdot 8}{17 \cdot 8}) \cdot 100\% = 29.41\%$$

Whereas the space savings of SN with $\delta = 0$ is equal to:

$$\textit{Space Savings } (SN, \delta = 0) = (1 - \frac{66}{17 \cdot 8}) \cdot 100\% = 51.47\%$$

The analysis at Table 6 shows that applying RLE instead of SN would not be more efficient in terms of the compression performance since average data count per pixel is < 2, i.e. the generated DMs using local block matching algorithm do not have row edges densely.

Table 6: RLE analysis with different compression thresholds (δ).

DM type/Average consecutive and compressible data count	$\delta = 0$	$\delta = 1$	$\delta = 3$
Unfiltered local block matching based DMs	1.52	1.64	1.67
Median filtered local block matching based DMs	1.59	1.71	1.74
Ground truth DMs (0-bit fraction precision)	1.93	1.99	1.99
Ground truth DMs (1-bit fraction precision)	1.87	1.87	1.87
Ground truth DMs (2-bit fraction precision)	1.76	1.76	1.76
Ground truth DMs (3-bit fraction precision)	1.63	1.63	1.63

3.3 Line adaptive compression

We have investigated adaptive approaches to improve the compression performance of SN and MN methods. However, these approaches are found to provide little benefit compared to their implementation complexity. Firstly, we allowed selecting an independent compression threshold (δ) for each line of the DM and applied this mechanism to both SN and MN methods. We looked at how often different δ values are selected, and results show us that there is little benefit of this approach since δ is predominantly selected as 0. Secondly, we looked dividing each line into multiple slices and assigning an individual δ per slice. Eq. (17) shows the additional bits required by this approach.

In this equation, F is the total number of extra bits required by this approach, H and W are frame height and width, respectively, S is the number of horizontal slices per line, and B is the maximum number of bits required to represent δ.

$$F = H \cdot W \cdot S \cdot B \ (bits) \tag{17}$$

Compared to its implementation complexity, this approach did not result with a significant improvement in compression performance either. The hardware implementation of the adaptive approach has either do a preprocessing on each line to find the optimal δ value or process each line for different δ values in parallel and then decide for the optimum δ value.

The former approach will increase the latency and decrease the throughput of the hardware architecture. The latter approach can maintain the same throughput and latency values of the non-adaptive approach, but it will significantly increase

the silicon area of the encoder hardware architecture due to parallel processing. For the latter approach, the silicon area can be multiplied with the number of different δ values, on which encoder blocks will be working in parallel.

We have also investigated an adaptive approach selecting between SN and MN variants for each line and for each slice. For smooth regions of a DM, i.e., for constant backgrounds, SN approach can perform better than the MN approach due to consuming less header bits. For other parts of the DM, where there are horizontal edges in the DM, MN can perform better due to looking for correlations between multiple neighbours including neighbours from upper line.

Although this approach can obtain minor improvements in average compression results, due to its hardware implementation cost it is excluded from the proposed hardware architecture. Table 7 shows average space savings results for these adaptive approaches.

We have analysed the frequency of selecting different δ values. For the unfiltered DMs compressed by the SN method, the ratios for selecting δ of 0, 1, and 3 are 57.88%, 36.09%, and 6.03%, respectively. For the median filtered DMs compressed by the SN method, the ratios for selecting δ of 0, 1, and 3 are 61.15%, 31.90%, and 6.95%, respectively.

For the unfiltered DMs compressed by the SN method, the ratios for selecting δ values of 0, 1, and 3 are 57.88%, 36.09%, and 6.03%, respectively. For the median filtered DMs compressed by the SN method, the ratio for selecting δ of 0, 1, and 3 are 61.15%, 31.90%, and 6.95%, respectively.

Since the frequency of selecting the δ of 3 is much less likely for both unfiltered and median filtered DMs, we reduce B from 2 bits to 1 bit and investigated the scenario of selecting δ of only 0 and 1. This limitation in adaptive δ values resulted with a slight increase in the performance compression as shown in Table 7. For the unfiltered DMs compressed by the MN method, the ratios for selecting δ of 0, 1, and 3 are of 94.78%, 5.03%, and 0.20%, respectively.

$\delta = 3$ is selected extremely rare, this is due to higher number of header bits

required to realize the MN encoder.

For the adaptive SN/MN approach, we selected a fixed δ of 0, because we noticed that allowing other values for δ does not improve the results. For unfiltered DMs, the ratios for selecting SN and MN methods are 34.52% and 64.48%, respectively. For median filtered DMs, this ratio becomes 61.93% for the SN approach and 38.07% for the MN approach. Finally, considering their benefit-cost ratios, we decided not to include any of the investigated adaptive mechanisms in the hardware implementation.

Table 7: Average space savings results (%) for local block matching based DMs using adaptive approaches.

Slices per line	δ-adaptive SN ($\delta = 0, 1, 3$)		δ-adaptive MN ($\delta = 0, 1, 3$)		δ-adaptive SN ($\delta = 0, 1$)		δ-adaptive MN ($\delta = 0, 1$)		Adaptive SN/MN ($\delta = 0$)	
	Without median	With median	Without median	With median	Without median	With median	Without median	With median	Without median	With median
1	39.79	48.27	45.60	**56.09**	39.79	48.27	45.61	**56.09**	45.77	56.37
2	39.94	48.39	45.63	56.08	39.95	48.39	45.64	56.09	45.84	56.44
4	40.18	48.60	45.67	56.07	40.20	48.61	45.69	56.08	45.95	56.54
8	40.39	48.79	45.71	56.04	40.42	48.82	45.74	56.08	46.06	56.64
16	40.61	49.01	**45.74**	56.00	40.68	49.07	45.80	56.07	46.20	56.76
32	40.82	49.23	45.70	55.90	40.95	49.33	**45.83**	56.04	46.31	56.83
64	40.99	49.42	45.56	55.67	41.23	49.61	45.81	55.94	46.42	56.89
128	41.12	49.58	45.23	55.22	41.56	49.94	45.73	55.76	46.54	56.93
256	41.19	49.70	44.61	54.44	42.02	50.42	45.60	55.50	46.86	57.16
512	**41.24**	**49.79**	43.43	53.00	42.83	**51.23**	45.34	55.03	**47.36**	57.58
1024	39.68	48.37	42.03	51.13	**42.87**	51.21	45.78	54.98	47.22	**57.68**

3.4 Lossy SN/MN compression with quantization

Uniform quantization can be done on the processed DM in order to decrease the disparity range during compression. Thus, increasing spatial correlations and compression ratio of the DM frames. Moreover, quantization itself is a lossy compression method that has a trade-off between compression performance which increases as the quantization (reduction) factor (Q) increases, and the quality of the resulted DMs. Eq. (18) demonstrates space savings of lossy quantization combined with the proposed compression methods. Where N is the number of original data bits, $\hat{N}$ is the number of quantized data bits, CB is the number of the compressed bitstream bits, H is DM height, and W is DM width.

$$Space\ Savings = 1 - \frac{CB}{H \cdot W \cdot (\hat{N} + log_2 Q)} \tag{18}$$

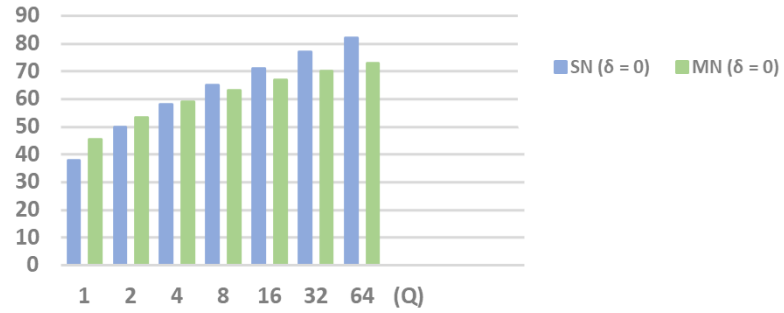
$$\hat{N} = N - log_2 Q \tag{19}$$

For example, when applying 7-bit quantization to 8-bit DM, then $Q = 2$ and without any SN or MN compression, the decoder can estimate the original DM by inverting the quantization as:

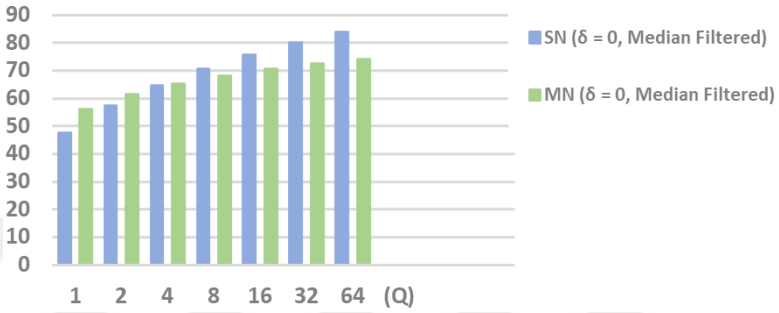
$$\hat{DM} = 2 \cdot DM_q$$

Where $\hat{DM}$ is the restored DM, and DM_q is the quantized DM. Since the original DM disparity range is [0, 255], and the quantized DM disparity range is [0, 127], a flooring or ceiling operation can be done to quantize disparities that are not divisible by 2 (odd numbers). Such operations cause restoration of the DM with losses.

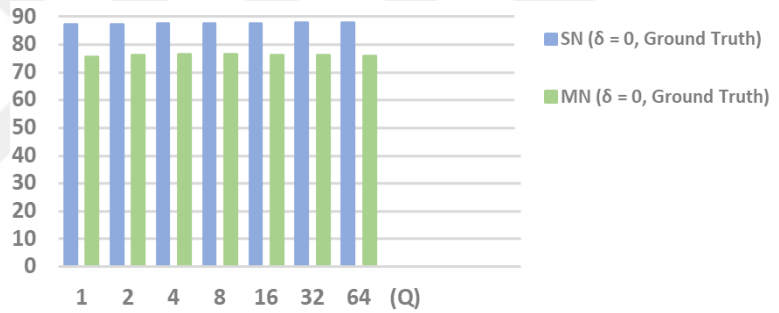
As it is shown in Figure 6, $Q = 1$ is the lossless case which is not different from the originally proposed SN and MN methods. As Q increases, generally space savings increase. However, for ground truth DMs, the denominator in Eq. (18) does not have a significant effect on the numerator when changing the value of Q due to high spatial correlations (and low CB) in such DMs.



(a)



(b)



(c)

Figure 6: Space savings of uniform quantization applied with the proposed compression algorithms. (a) Unfiltered local block matching DMs. (b) Median filtered local block matching DMs. (c) Integer ground truth DMs.

Figure 7 illustrates PSNR-Q graph which has a negative relationship since as Q increases, DM quality gets lower. It is obvious that for the three types of the tested DMs: local block matching based DMs (unfiltered), local block matching based DMs (median filtered), and original ground truth DMs, PSNR-Q graphs take very similar shapes and values.

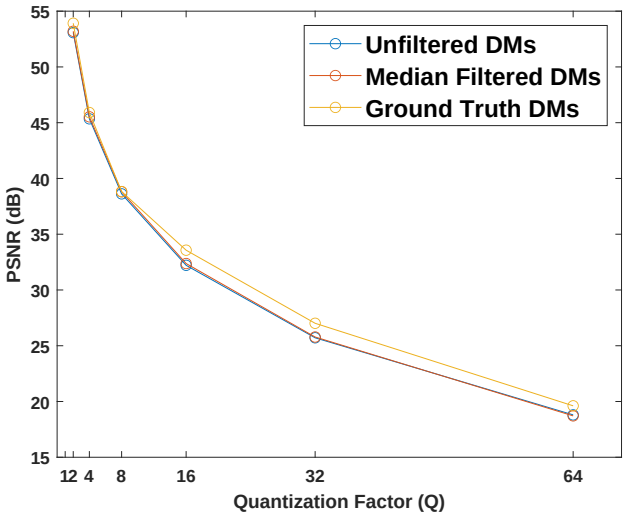


Figure 7: PSNR results corresponding to DM quantization.

3.5 Lossless compression of RGB and grayscale images

Although that this thesis focuses on DM frames. It is a good idea to see what happens when its application area is extended into other high-resolution images such as RGB (multiple-channel) and grayscale images. The used dataset includes images that have average resolutions approximately equal to 4K: an average resolution of 3585 x 2798 for RGB images, and an average resolution of 3546 x 2745 for grayscale images [26]. Table 8 shows lossless compression results for 14 images using different lossless compression algorithms. Furthermore, Table 9 includes the corresponding results for 15 grayscale images. Except the proposed algorithms of SN and MN, other results are given as reported in [26]. The proposed methods originally consider 1-channel images such as DMs. However, for multiple-channel images such as RGB, SN and/or MN is applied for each color channel separately without complicated interventions. The overall space savings or BPP is averaged over all channels. According to the calculated results, MN is still better than SN in terms of compression performance. Nonetheless, both of SN and MN compression results are much lower than other lossless algorithms with higher complexities. Still, the proposed low-cost algorithms are more beneficial when it comes to implementations with limited hardware resources, low power, and high processing speed (frequency) are required.

Table 8: Lossless compression algorithm results on high-resolution 8-bit RGB images.

Image/algorithm	JPEG-LS	JPEG2000	HDPhoto	SN ($\delta = 0$)	MN ($\delta = 0$)
artificial	0.77	0.98	1.85	1.93	2.60
big_building	3.81	3.56	3.83	7.78	7.44
big_tree	4.31	4.39	4.65	8.05	7.92
bridge	4.53	4.53	4.61	8.36	8.56
cathedral	3.77	3.75	3.97	7.51	7.07
deer	5.22	5.32	5.40	8.59	8.94
fireworks	1.52	1.78	2.56	3.89	3.77
flower_foveon	2.03	2.14	2.61	5.91	4.64
hdr	2.46	2.53	2.95	6.60	5.34
leaves_iso_1600	4.63	4.49	4.71	7.96	8.08
leaves_iso_200	3.92	3.69	4.02	7.59	7.49
nightshot_iso_100	2.20	2.43	2.93	5.7	4.74
nightshot_iso_1600	4.20	4.39	4.61	7.40	7.18
spider_web	1.81	1.90	2.47	6.03	4.71
Average BPP	**3.23**	**3.28**	**3.66**	**6.66**	**6.32**
Average space savings (%)	**59.65**	**59.04**	**54.30**	**16.70**	**21.00**

Table 9: Lossless compression algorithm results on high-resolution 8-bit grayscale images.

Image/algorithm	JPEG-LS	JPEG2000	HDPhoto	SN ($\delta = 0$)	MN ($\delta = 0$)
artificial	0.80	1.19	2.49	1.95	2.61
big_building	3.59	3.65	3.88	7.62	7.14
big_tree	3.73	3.80	3.97	7.95	7.52
bridge	4.15	4.19	4.32	8.29	8.34
cathedral	3.57	3.70	3.91	7.66	7.12
deer	4.66	4.58	4.74	8.47	8.60
fireworks	1.47	1.65	2.61	3.94	3.73
flower_foveon	2.04	2.19	2.72	6.01	4.47
hdr	2.18	2.34	2.80	6.13	4.83
leaves_iso_1600	4.49	4.67	4.80	8.17	8.28
leaves_iso_200	3.82	4.08	4.30	7.63	7.48
nightshot_iso_100	2.13	2.30	2.83	5.76	4.60
nightshot_iso_1600	3.97	4.03	4.19	7.92	7.60
spider_web	1.77	1.90	2.51	5.84	4.48
zone_plate	7.43	5.75	7.25	8.96	9.86
Average BPP	**3.50**	**3.49**	**3.92**	**7.17**	**6.72**
Average space savings (%)	**56.27**	**56.40**	**51.05**	**10.40**	**16.03**

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 8: Examples of the tested RGB and grayscale images. (a,e): Artificial. (b,f): Big building. (c,g): Big tree. (d,h): Bridge.

CHAPTER IV

PROPOSED HARDWARE ARCHITECTURES

We proposed hardware architectures for the median filter, SN, and MN encoders/decoders. Proposed hardware architectures are designed and verified using Verilog RTL. 3 x 3 median filter is included in the encoder design since it is an essential step to smooth out the DMs generated by local block matching disparity estimation methods. Proposed hardware architectures are designed for $\delta = 0$, because this setting obtains the best compression results. Setting $\delta = 0$ also reduces the hardware complexity leading to minimum silicon area consumption. Proposed hardware architectures are designed to operate in raster-scan order. Raster-scan order is widely used in displays, it starts processing pixels from leftmost side of the top row (horizontal line) and ends when it reaches the rightmost pixel of the bottom row as shown in Figure 9. The implemented hardware architectures are configurable and can support any resolution and disparity range. The throughput of the encoder architecture is one disparity value per clock cycle. The implemented design supports 8-bit disparity range resulting in 1-bit SN compressed, 9-bit SN uncompressed, 2-bit MN compressed, and 10-bit MN uncompressed data in the bitstream.

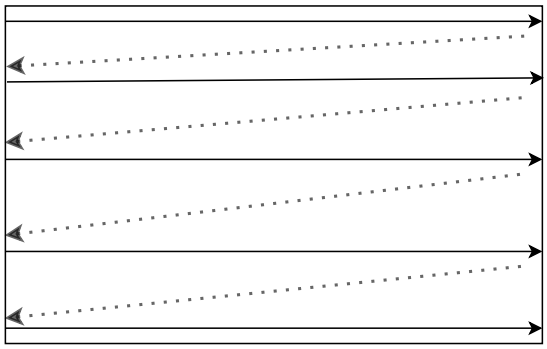


Figure 9: Raster-scan order in a display.

4.1 Design of 3×3 median filter, SN, and MN encoders

Figure 10 illustrates the proposed 3 x 3 median filter and SN encoder hardware block diagrams. Furthermore, Figure 11 illustrates the proposed hardware block diagram of MN encoder only, supposing its input is median filtered.

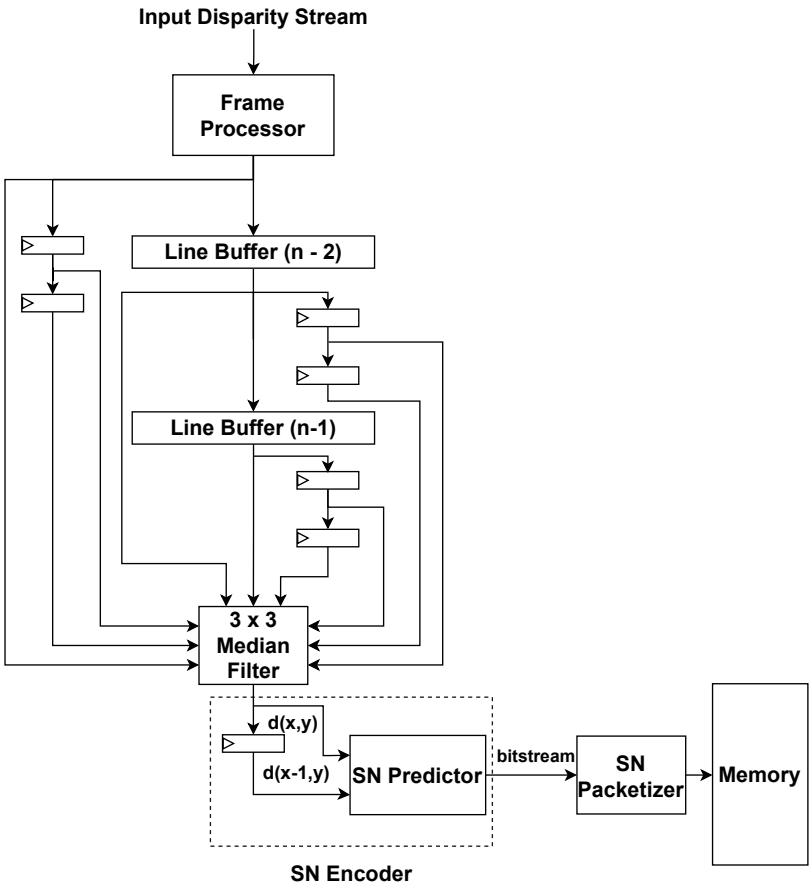


Figure 10: Block diagram of a 3 x 3 median filter connected to the proposed SN encoder.

Frame processor module in Figure 10 generates the vertical and horizontal synchronization signals based on the input resolution. Those signals are generally referred to as VSYNC and HSYNC. The first one indicates that refreshing the current frame is required to the next frame, and the later one indicates that refreshing the current frame row (line) is required. Output of the frame processor module is stored in two-line buffers to generate the previous two lines for the 3 x 3 median filter. The values read from line buffers as well as the value from the input disparity stream are registered to generate the left neighbors on these lines

for the 3 x 3 median filter. The SN encoder shown in Figure 10 registers the input disparity value, $d(x, y)$, and uses it as the left neighbor of the current disparity, $d(x-1, y)$, in the next clock cycle.

The MN encoder shown in Figure 11 stores the current line of the input disparity stream in a line buffer. Reading the values from the line buffer, the MN encoder compares the current disparity value with its left, $d(x-1, y)$, top, $d(x, y-1)$, and top right, $d(x+1, y-1)$, neighbors. Both of encoder and decoder architectures can work with arbitrary sized data words. In this proposal, they are designed to operate on 128-bit data words, which makes them suitable to transfer data to and from the memory via an SoC bus protocol like AXI.

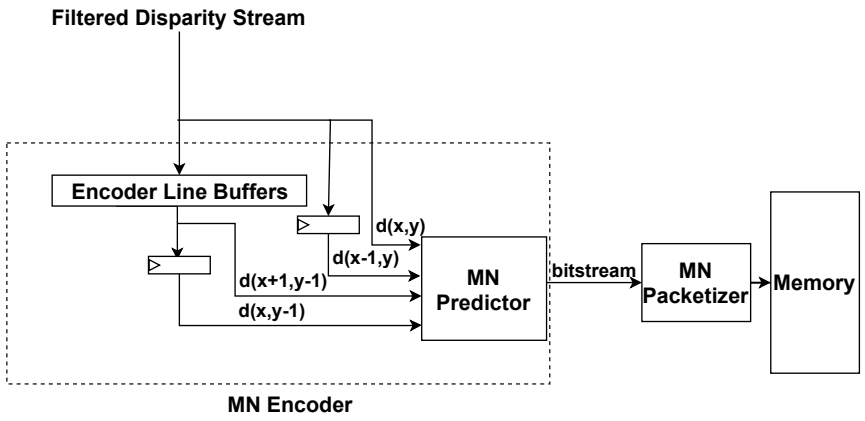


Figure 11: Block diagram of the proposed MN encoder.

The packetizer module shown in Figure 10 and Figure 11 is responsible for generating 128-bit memory words from the compressed bitstream received from the encoder module. Generated bitstream is efficiently packed into memory words without wasting any bits. If the received stream for the current disparity from the encoder does not align with the 128 bits boundary of the current memory word, all the available bits in the current memory word will be filled with the least significant bits of the received stream and the next memory word will start with the remainder of the received stream. To show how the combinational circuit of the 3 x 3 median filter in Figure 12 works, suppose we have 9 inputs that are:

$in_1, in_2, in_3, in_4, in_5, in_6, in_7, in_8, in_9$ in which the ordered list of them in ascending order is: $in_1, in_4, in_3, in_7, in_8, in_2, in_6, in_9, in_5$. Thus, the median value is equal to in_8 and if we can show that using the flow of Figure 12, it should be enough as a sort of *Proof by Example*.

$$Sorter\text{-}1(MAX) = in_2,\ Sorter\text{-}1(MED) = in_3,\ Sorter\text{-}1(MIN) = in_1 \quad (20)$$

$$Sorter\text{-}2(MAX) = in_5,\ Sorter\text{-}2(MED) = in_6,\ Sorter\text{-}2(MIN) = in_4 \quad (21)$$

$$Sorter\text{-}3(MAX) = in_9,\ Sorter\text{-}3(MED) = in_8,\ Sorter\text{-}3(MIN) = in_7 \quad (22)$$

$$Sorter\text{-}4(MIN) = in_2 \quad (23)$$

$$Sorter\text{-}5(MED) = in_8 \quad (24)$$

$$Sorter\text{-}6(MAX) = in_7 \quad (25)$$

$$Sorter\text{-}7(MED) = in_8 \quad (26)$$

We can see clearly that answer of Eq. (26) correctly matches the expected median value which is equal to in_8 according to the ordered list previously. As it can be seen from Figure 12, in Verilog RTL we need to design the combinational circuit of *Sorter* module and make the proper connections between its 7 instances. The logic design of the *Sorter* requires covering 6 different possible conditions that result in 6 different output combinations, MAX, MED, and MIN.

Table 10: Combinations of input comparisons and their ordered results.

If condition	MAX	MED	MIN
$in_1 \leq in_2 \leq in_3$	in_3	in_2	in_1
$in_1 \leq in_3 \leq in_2$	in_2	in_3	in_1
$in_2 \leq in_1 \leq in_3$	in_3	in_1	in_2
$in_2 \leq in_3 \leq in_1$	in_1	in_3	in_2
$in_3 \leq in_1 \leq in_2$	in_2	in_1	in_3
$in_3 \leq in_2 \leq in_1$	in_1	in_2	in_3

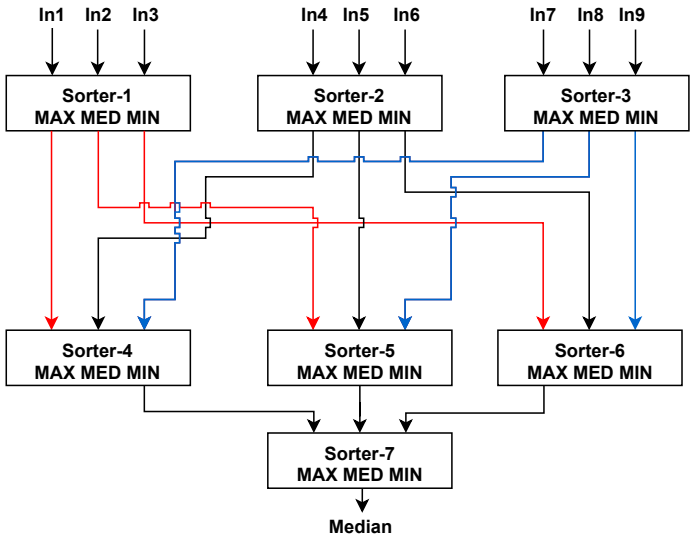


Figure 12: Sorting network of the implemented 3 x 3 median filter.

4.2 Design of SN and MN decoders

For SN decoder, decoding operation starts form the least-significant-bit (LSB) part of the current memory word and terminates at the most-significant-bit (MSB) part of the last memory word for each frame. The value of the header bit determines the compression state, and increase amount in the processed bits index of the memory word. As explained in the encoder design, last bits of any memory word's MSB part contain portion of the current uncompressed data (P_R) if the remaining bits in the memory word (R_{bits}) are less than disparity bits, which are equal to 8 for this design. In such case, after a new memory word is fetched, R_{bits} many bits from its LSB part are stored into another register called P_L. The final decoded disparity value is equal to the sum of left-shifted P_L and P_R. It is guaranteed

that two portions are combined without incorrect intersections in bit locations. A similar hardware design is implemented for MN decoder. The main differences between two decoders are, in MN case decoded disparities are buffered for each frame row to be selected when a compression condition is satisfied according to the 2-bit header value. Furthermore, the MN decoder needs additional logic to take care of keeping correct memory word indexing if the memory word length is odd which is not common in digital systems. However, if such design is required, the header bits can be split between two consecutive memory words.

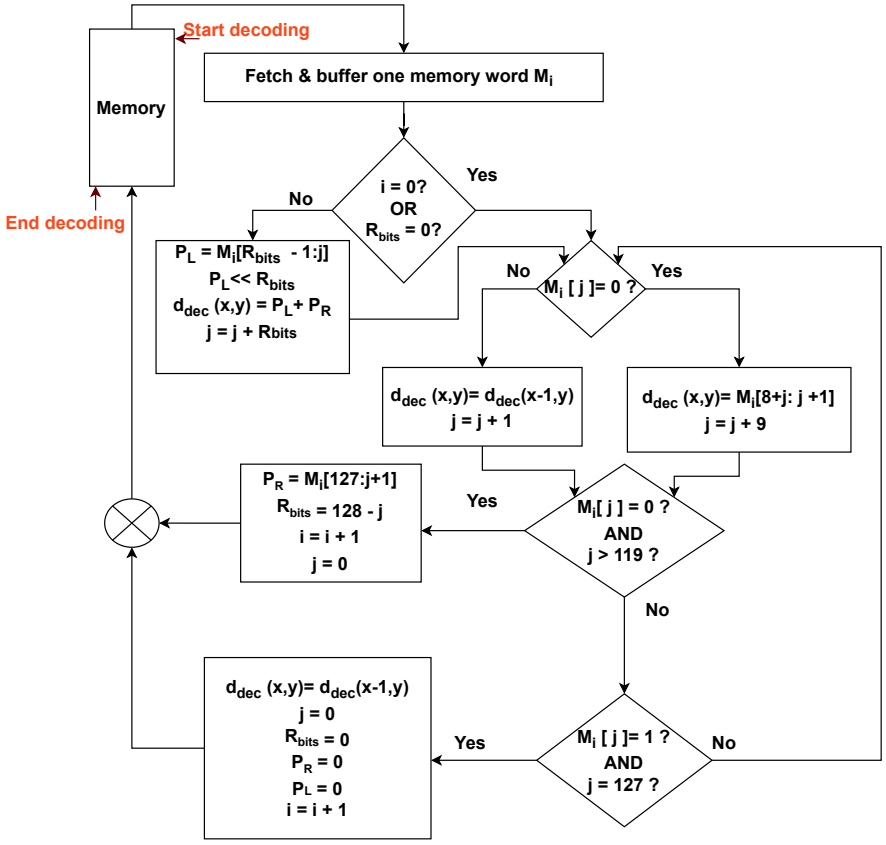


Figure 13: Flowchart of the designed SN decoder.

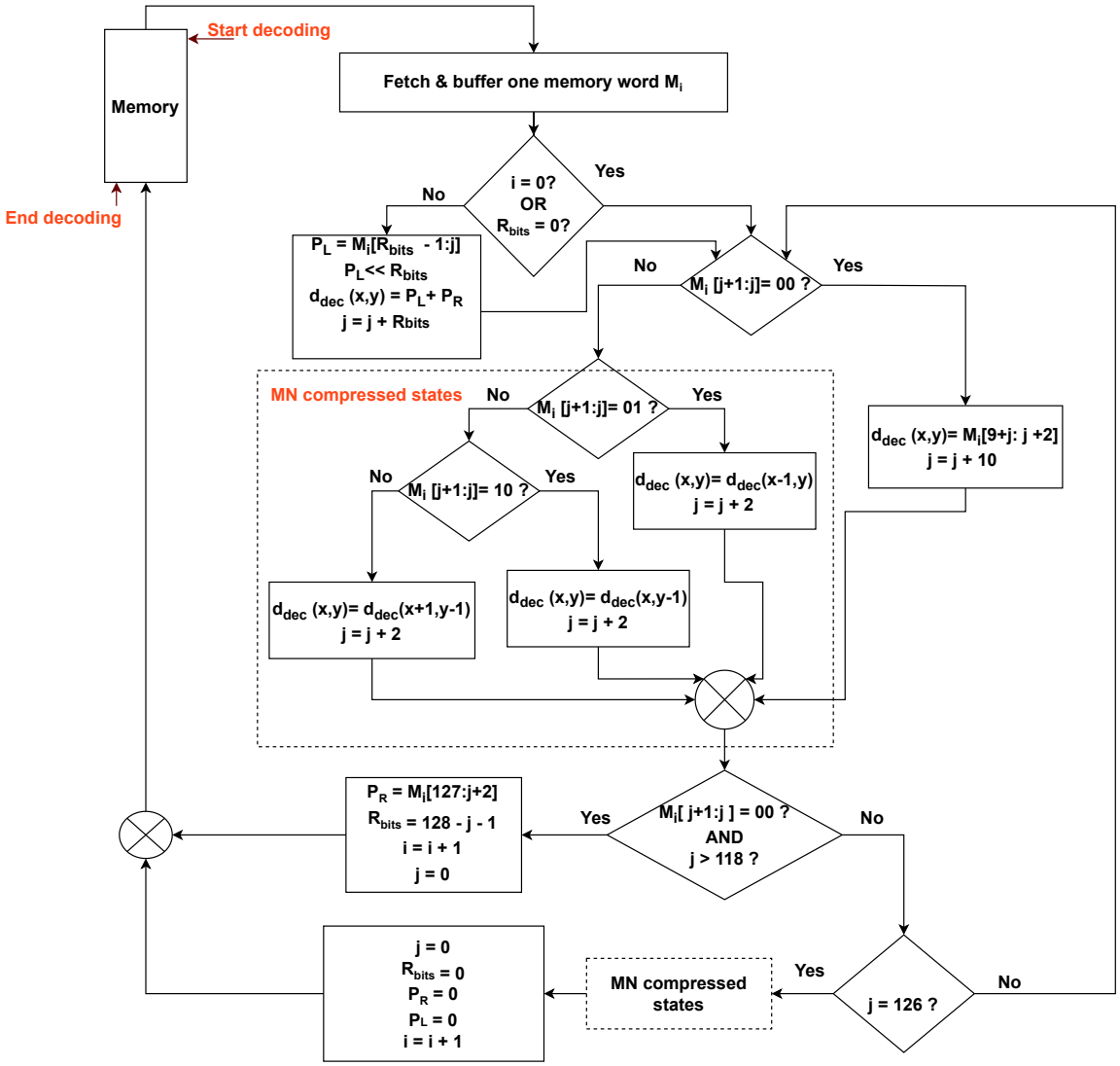


Figure 14: Flowchart of the designed MN decoder.

Note that MN decoder requires a line buffer to decode a compressed disparity that is equal to its top or top-right neighbors. Line buffers can be synthesized and implemented using different options: IP core of BRAM generator , IP core of FIFO generator, and distributed RAM [27]. IP cores are generally provided by the FPGA vendor such as Xilinx and Intel (Altera) with their software suits such as ISE, Vivado, and Quartus. IP cores require only setting correct parameters of the BRAM/FIFO such as memory words width and depth, as well as if the BRAM has to be single-port or dual-port. After that, connections between the IP core

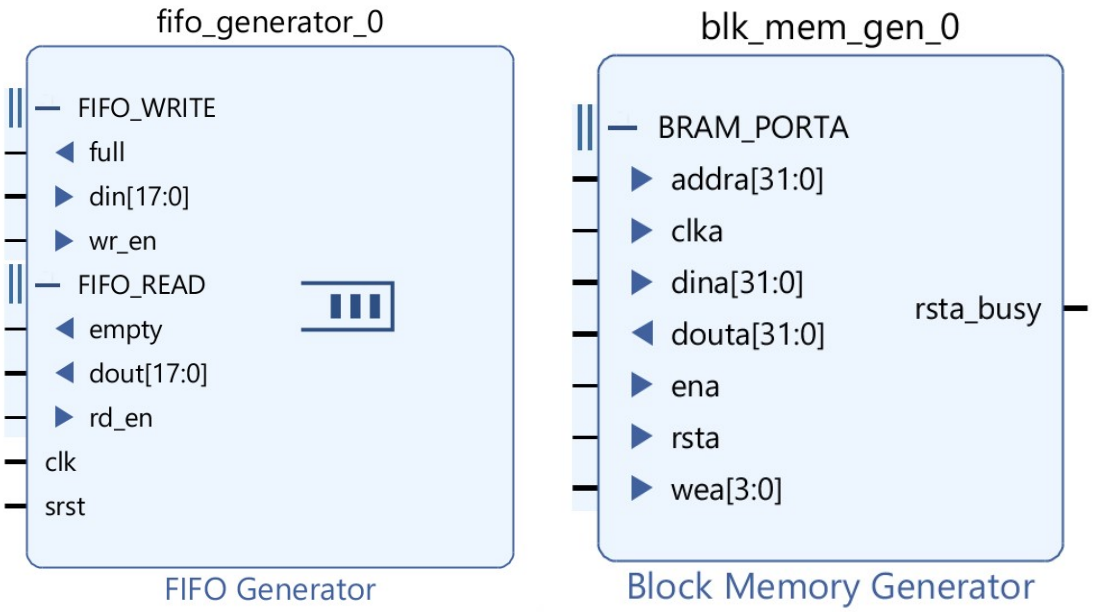
and original design modules have to be completed using an IP wrapper. However, distributed RAMs are preferred in this project for Xilinx FPGA since:

- They are easier to control and change using Verilog or VHDL RTL.
- No other IP cores are included in this design, so designing all modules in a codable RTL form seems more systematic.
- Some IP cores such as FIFO generator cannot support all depth values in which it would contradict with our claim of supporting any frame resolution.
- Implementation runtime of distributed RAMs is generally less than when IP-based modules exist.
- This thesis targets low-cost FPGAs, however, it can also emerge into low-cost ASICs. In that case, implemented distributed RAMs by an HDL are more compatible with ASIC EDA tools.

Important: The used Verilog RTL for distributed RAMs at [27] is mapped after FPGA implementation process into BRAM in most of the cases. However, when different implementation strategy, different width and depth values, or different operations are included with the original buffer another distributed RAM called LUTRAM can also be used. Such smaller memory cells that are based on the flexible structure of LUT exist in many FPGAs within a plentiful quantity.

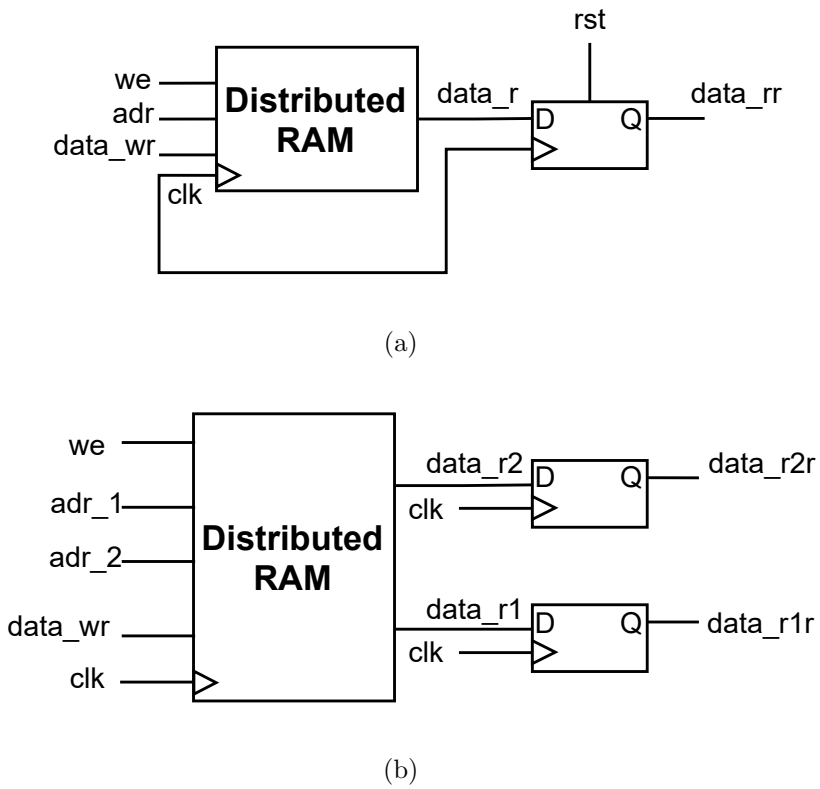
Figure 16 includes the block diagrams of the implemented distributed RAMs in which we have preferred using a single-port RAM for both median filter and MN encoder designs. Since its cost and area is lower compared to the dual-port RAM especially when considering ASIC and SoC architectures.

However, for MN decoder we have decided to implement the line buffer using a dual-port RAM since the MN decoder hardware algorithm requires a complex state machine. Thus, its total latency and complexity are reduced by using a dual-port RAM that accesses to multiple neighbor disparities more quickly and easily. Table 11 explains I/O signals shown in Figure 16.



(a) (b)

Figure 15: Xilinx FPGA BRAM and FIFO IP cores.



(a)

(b)

Figure 16: Distributed RAMs. (a) Single-port RAM with false synchronous read and resetable data read. (b) Dual-port RAM with false synchronous read.

Table 11: Descriptions of the use signals in the implement distributed RAMs.

Port/signal abbreviation	Description
clk	System clock
rst	Active-high system reset
we	Write enable
data_wr	Data to be written into a single-port RAM
data_r	Data to be read from a single-port RAM
data_rr	Register output of data_r
adr	Single-port RAM read/write address
adr_1	Dual-port RAM primary read/write address
adr_2	Dual-port RAM secondary read/write address
data_r1	Dual-port RAM read data corresponding to adr_1
data_r1r	Register output of data_r1
data_r2	Dual-port RAM read data corresponding to adr_2
data_r2r	Register output of data_r2

CHAPTER V

HARDWARE RESULTS

This chapter includes the technical properties of the used FPGAs such as their available resources and software environment. Verification methods with MATLAB models are also explained. Moreover, hardware implementation results of area, timing, and power are highlighted.

5.1 FPGA specifications

The specifications of the used FPGA devices for this thesis are illustrated in Table 12. As it can be clearly seen, when the process technology gets smaller, more logic elements can be fit in the target FPGA. Moreover, higher operating frequency can be achieved. As for BRAM space, considering 1920 x 1080 (1080p) based 8-bit uncompressed depth frame, both of 18 Kb and 36 Kb BRAMs are sufficient to buffer frame line. However, higher resolutions such as 3840 x 2160 (4K) with 8-bit for each depth value require 36 Kb BRAM. 6-input LUT is generally more advantageous than 4-input LUT since it can process complex operations with higher number of inputs and reduce the critical path. Where critical path can be defined as the logic path that has the maximum delay in the design.

Table 12: Specifications of the used FPGAs.

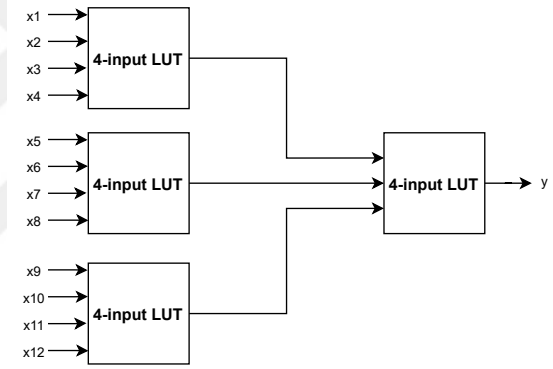
Device part	Device family	Process technology	Software suit	I/O	LUT	BRAM	FF
XC7S50FGGA484	Spartan-7	28 nm	Xilinx Vivado	484	32600 (6-input)	75 (36 Kb/18 Kb)	65200
XC6SLX45	Spartan-6	45 nm	Xilinx ISE	316	27288 (6-input)	348 (18 Kb/9 Kb)	54576
XC4VLX15	Virtex-4	90 nm	Xilinx ISE	240	122800 (4-input)	48 (18 Kb)	122800

Eq. (27) shows multiple XNOR operations between 12 input x and outputs y.

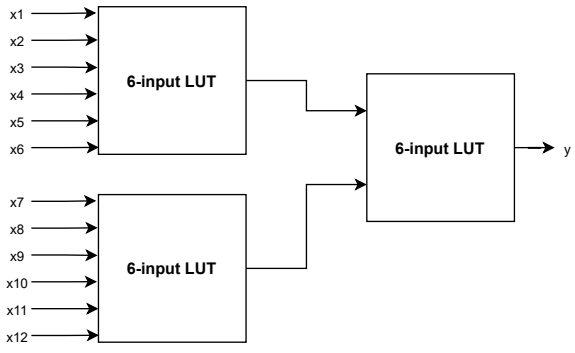
$$y = \sum_{i=0}^{n=12, i<n} x_i \odot x_{i+1} \tag{27}$$

Such operations can be done using typical XOR gates which consist of NOR, OR, and AND gates. However, FPGA tends to map such operations with LUT

as shown in Figure 17 where both cases of using 4-input LUT and 6-input LUT appear. As it can be understood from Figure 17, a 6-input LUT is more advantageous than 4-input LUT since it can handle more inputs to complete the same function. Thus, it generally ends with less number of logic stages and area. Moreover, if the number of logic stages to implement the same function reduces, the corresponding function total delay reduces as well. Even if the number of logic stages of both 4-input LUT and 6-input LUT is the same, 6-input LUT can still be more efficient in terms of timing performance since most probably its final stage will have less fan-in which is the number of its connected inputs. When the fan-in of a gate increases, its input capacitance increases and the total propagation delay increases as well.



(a)



(b)

Figure 17: Multiple logical operations using different LUT architectures. (a) Using 4-input LUTs. (b) Using 6-input LUT.

5.2 Implementation and verification methodologies

Xilinx Vivado supports implementation strategies that try to optimize specific factors such as area, power, and timing performance. *"Performance_ExtraTimingOpt"* strategy is selected for implementation since the proposed architectures are low-cost and their functionality already consume small silicon area. Thus, focusing on optimizing maximum operating frequency would be critical in this project. Similar implementation strategy is used with Xilinx ISE too. Worst Negative Slack (WNS) and Total Negative Slack (TNS) are two important concepts for timing analysis of Xilinx tools. It is very important to have a tight timing constraint that pushes timing optimization into its best achievable limits. For Xilinx Vivado, the maximum clock frequency F_{max} can be calculated as:

$$F_{max} = \frac{1}{P - WNS} \qquad [Hz] \tag{28}$$

Where P is the target clock period in the timing constraint. When WNS is negative, timing target fails and the implementation does so. Since it shows that the critical path delay is longer than the required clock pulse width, hold, or setup time with such amount. TNS is the sum of delays of all failing paths and it is equal to zero when the implementation is successful.

Figure 18 demonstrates the verification steps used to make sure that the proposed lossless compression methods match with the implemented hardware designs. It is very hard to debug a hardware deign testbench mismatches if the design has multiple signals and modules. Thus, for both SN and MN verification, first the implemented 3 x 3 median filter is checked to see if it outputs the same DM that MATLAB model does. Both of *MN Model* and *SN Model* include actually separated models of each which are:

- *Bitstream Encoder*: Responsible for verifying that the compressed bitsream output of the hardware design is as expected before the packetizing operation in a BRAM starts.

- *Memory Words Generator*: Generates the corresponding 128-bit memory

words of the compressed bitstream that includes 1-bit and 9-bit disparities. It must match with the packetizer output of the hardware encoders.

- *Memory Words Decoder*: Its input is the compressed memory words of the hardware design, and its output must be the compressed DM without any incorrect pixel value.

- *Decoder Model*: It checks that both hardware decoders output the expected DMs correctly.

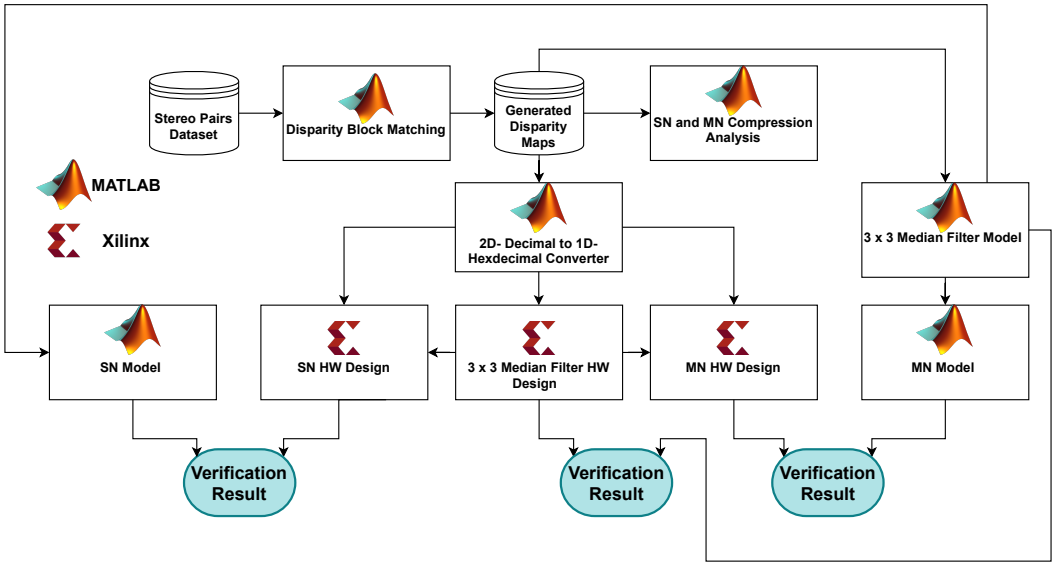


Figure 18: Verification of the implemented hardware designs with MATLAB models.

MATLAB can deal with images easily in their 2D structure and in any format. However, for hardware testbench it is required to read and write the DMs in a serial 1D format with resepect to raster-scan order.

5.3 Utilization, timing, and power results

Table 13 shows the implementation results of proposed hardware architectures for various low-cost FPGA device families. To allow a fair comparison with the papers reported in the literature, Table 13 presents results for low-cost FPGAs manufactured with 28 nm, 45 nm, and 90 nm process technologies. As it can be seen in Table 13, the proposed hardware architectures consume a much smaller silicon area while supporting a higher throughput when compared to the related work reported in the literature. When implemented on a 28 nm FPGA device, MN encoder and decoder can support 1920 x 1080 resolution at 87 fps and 89 fps, respectively. BRAM usage reported in this table is less for the 28 nm FPGA technology, because its BRAM size is twice the size of BRAM in 45 nm and 90 nm FPGA technologies.

The proposed hardware architecture for the SN method has only a single clock cycle latency since it registers the current disparity value and uses it as the left neighbor of the next disparity value. Because the MN method requires neighboring disparity values from the upper line, the latency of its hardware architecture is equal to the number of clock cycles required to process a single line of the input frame.

Table 14 shows the estimated dynamic power consumption of the proposed hardware architectures at 125 MHz and at their maximum clock frequencies. 125 MHz is the operating frequency to support 1080p resolution at 60 fps. Gate level simulations for the 28 nm FPGA implementation are run on the "Adirondack" benchmark image, which closely represents the average compression rate of the whole benchmark suit. This accurate power estimation is done after generating a Switching Activity Interchange Format (SAIF) during a post-implementation

simulation of the corresponding DM in the testbench, and it helps to calculate the required static probability (toggle rate) of each internal and external signal in the design. Thus, such simulation stores large files and can take longer runtime depending on the frame size, signals number, design complexity, and hardware capabilities of the simulation device. However, it guarantees upgrading dynamic power confidence level from "Low" or "Medium" into "High" which refers into a realistic analysis.

In a system with DDR4 DRAM, the proposed encoder and decoder couple can save 20 mW of power. The dynamic power consumption of DDR4 DRAM is given as 51 pJ/bit [28]. Based on this figure, the power consumed when writing the DM to the memory and reading it back from the memory once per frame for 1080p resolution at 60 fps is calculated as 102 mW. Using the MN method together with the median filter, the memory bandwidth reduces to 43.93%, which results with 45 mW of power consumption when accessing the memory. Adding the power consumed by the encoder and the decoder itself (27 and 8 mW), the total power consumption of the proposed method can be calculated as 80 mW. If the proposed hardware encoder and decoder architectures access a DDR3 DRAM, which consumes 64 pJ/bit [28], then the estimated power savings at the system level will be 33 mW. These calculations are based on fully utilized DRAM bandwidth. Memory bandwidth utilization in real systems will be less than ideal. Therefore, the actual power savings of the proposed hardware architectures is expected to be higher with reduced DRAM bandwidth utilization. The given power savings are calculated using the following equation:

$$P_S = P_{DRAM} - (P_{DRAM} \cdot \frac{100 - SS}{100} + P_E + P_D) \qquad \text{[W]} \tag{29}$$

Where P_S is the saved dynamic power, P_{DRAM} is the dynamic power of the used DRAM without compression, SS is space savings in [%], P_E and P_D are the encoder and decoder dynamic powers (including median filter) respectively.

Table 13: Implementation results.

Design	FPGA technology	LUT*	BRAM**	Maximum frequency (MHz)
Median filter + SN encoder	28 nm	1664	1	185
	45 nm	1899	2	160
	90 nm	2928	2	174
Median filter + MN encoder	28 nm	1838	2	168
	45 nm	1999	3	162
	90 nm	3859	3	173
SN encoder	28 nm	1056	-	255
	45 nm	1282	-	161
	90 nm	2100	-	182
MN encoder	28 nm	1605	1	180
	45 nm	1414	1	168
	90 nm	2822	1	164
SN decoder	28 nm	382	-	200
	45 nm	418	-	155
	90 nm	1515	-	171
MN decoder	28 nm	405	1	185
	45 nm	466	1	144
	90 nm	2210	1	150
Palaz et al. [20]	45 nm	4272	14	154
Papadonikolakis et al. [21]	90 nm	2596***	6	102.6
Inatsuki et al. [22] (encoder)	N.A	4284	-	100
Inatsuki et al. [22] (decoder)	N.A	7314	-	100

* For 28 nm FPGA, it includes utilized small number of LUTRAMs too (≤ 5).
** For 28 nm FPGA, odd results such as 1 and 3 are actually equal to 0.5 and 1.5 respectively. Since one BRAM in this FPGA contains two 18 Kb RAMs. However, showing BRAM results with fractions can somehow lead to undesired misunderstandings .
*** Given in slices.

Table 14: Dynamic power consumption results.

Design	Operating frequency (MHz)	Dynamic power (mW)
Median filter + SN encoder	125	25
	185	32
Median filter + MN encoder	125	27
	168	33
SN encoder	125	15
	255	23
MN encoder	125	21
	180	27
SN decoder	125	6
	200	7
MN decoder	125	8
	185	10

CHAPTER VI

CONCLUSION

In this thesis, we proposed an efficient disparity compression algorithm and its hardware architecture. The proposed algorithm is based on spatial correlations and obtains significant space savings when evaluated with a state-of-the-art benchmark suite. A detailed analysis of the proposed algorithm with various options is presented and shown that the proposed algorithm can efficiently work with well-known local disparity estimation methods based on block matching. The proposed algorithm is suitable to be implemented using low-cost silicon devices in real-time, which makes it highly advantageous for consumer electronics devices.

The motivation to conduct research in this area is the fact of depth information becoming essential to many consumer electronics applications including mobile phones, security cameras, augmented reality devices, drones, and robots. Consumer electronics devices using the depth information are becoming increasingly popular in our daily lives. Using the depth information foreground background classification can be made and based on this information autonomous drones and robots can avoid collisions. For example, a vacuum cleaner navigating autonomously in our homes makes use of the depth information. Depth information is also used for entertainment purposes. For example, when taking pictures with mobile phones, bokeh effect can be added using the depth information. Running augmented reality applications in mobile phones also relies on depth maps to align the rendered artificial content with the real world. All these consumer electronics devices require low-cost solutions to turn ideas into a product and take part in our daily lives.

The presented encoder and decoder hardware architectures are implemented using a low-cost FPGA device. The implementation results show that the encoder

and decoder architectures can support high resolution and frame rates and consume smaller area when compared with similar hardware architectures found in the literature. Implementation results also include power estimation analysis. Based on this analysis, the proposed encoder and decoder couple is proven to save power at system level, which makes it highly attractive for mobile devices by offering increased battery life and simplifying the effort to meet thermal constraints.

As for possible future works in this area, reducing the implemented line buffers size into half or even quarter saves a large portion of the silicon area, where the resulted new space savings (for depth compression) can be compensated. Such hardware implementation would also reduce the consumed dynamic power, since less number of line buffer pixel (disparity) values are being read/written. Moreover, when powerful ASIC EDA tools such as Cadence or Synopsys are available, different technology libraries can be used for ASIC implementation. Thus, emphasizing new silicon area, power, and timing results will be possible for the customized consumer electronic perspective.

APPENDIX A

MIDDLEBURY STEREO DATASET

This appendix shows the used stereo pairs to generate DMs using block matching algorithm, as well as each ground truth corresponding to each pair [24]. Please notice that original images with high-resolutions as indicated in Table 3 are down-sampled into a resolution of 500 x 750 using MATLAB to produce a thesis PDF file with proper size.

(a) (b) (c)

Figure 19: Adirondack. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 20: Jadeplant. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 21: Motorcycle. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 22: Piano. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 23: Pipes. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 24: Playroom. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 25: Playtable. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 26: Recycle. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 27: Shelves. (a) Left image. (b) Right image. (c) Ground truth DM.

(a) (b) (c)

Figure 28: Vintage. (a) Left image. (b) Right image. (c) Ground truth DM.

APPENDIX B

DMs USING BLOCK MATCHING

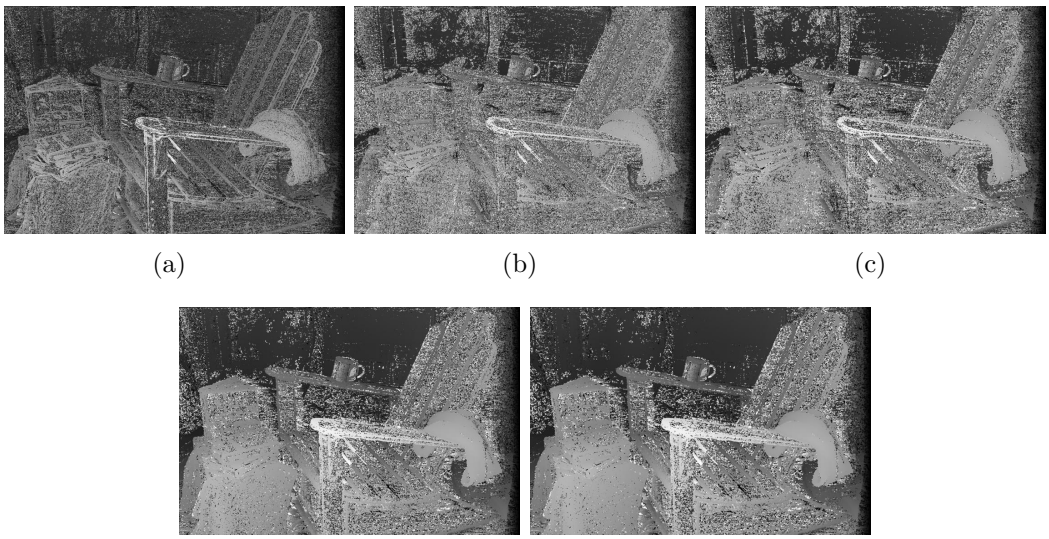
The generated DMs using block matching algorithm are displayed in this appendix with the same resolution used in Appendix A. Due to high error rate in those DMs compared to their ground truths, it is not easy for a human eye to distinguish between median filtered and unfiltered DMs. Thus, we include only postprocessed (filtered) DMs in this Appendix. Table 15 illustrates quality differences between orginal DMs dataset and generated DMs using local block matching algorithm. As it can be obviously seen from Table 15, the generated DMs are significantly noisy due to some factors which are:

1. The selected disparity range for generating DMs [0, 255] is generally lower than maximum disparity values in most ground truth DMs.

2. The selected window size for both preprocessing transforms and disparity matching is not large enough for better matching performance. Increasing window size would increase the complexity of the used cost functions such as SAD and Hamming distance.

3. The used algorithm is local, which means it does not have a specific cost function that considers multiple regions (or parameters) of each stereo image while deciding the best possible disparity value of a pixel.

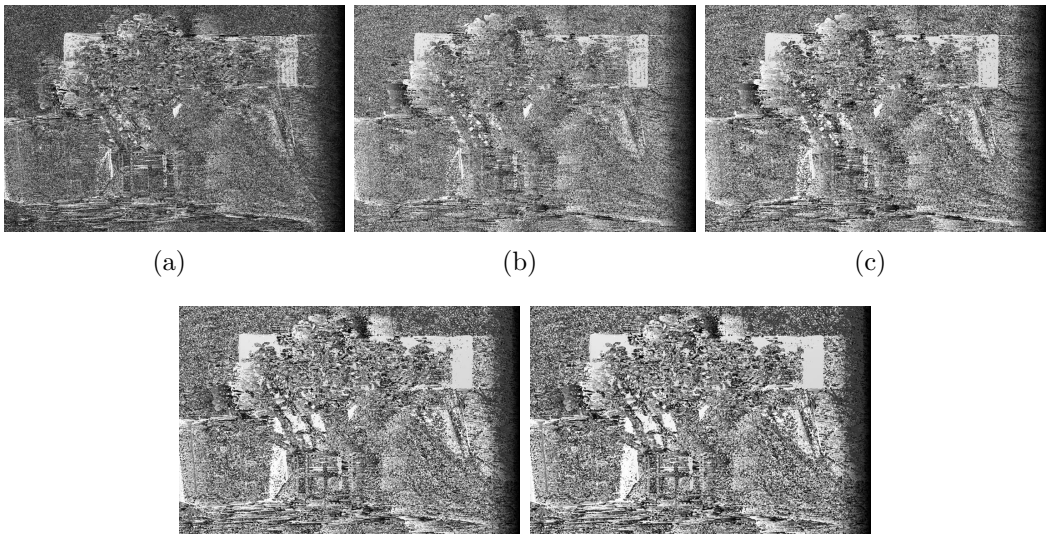
Nonetheless, this thesis has focused more on DM compression rather than the generation of DM itself. Since using a complex and global algorithm with a low-cost DM compression can be considered as a contradiction in the area of application. Moreover, generating large number of high-resolution DMs using algorithms such as SGM consumes an enormous amount of runtime in MATLAB.



(a) (b) (c)

(d) (e)

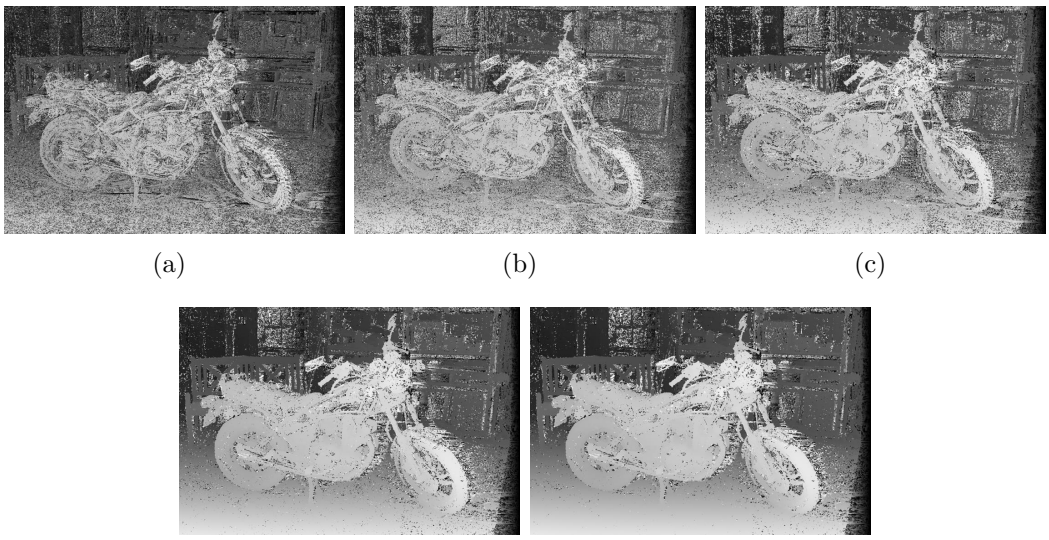
Figure 29: Adirondack. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

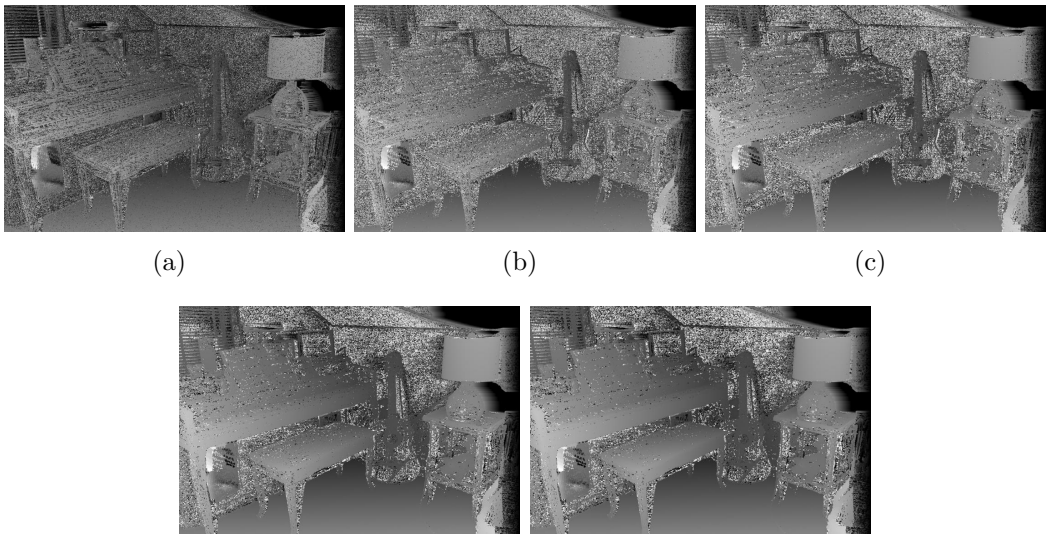
Figure 30: Jadeplant. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

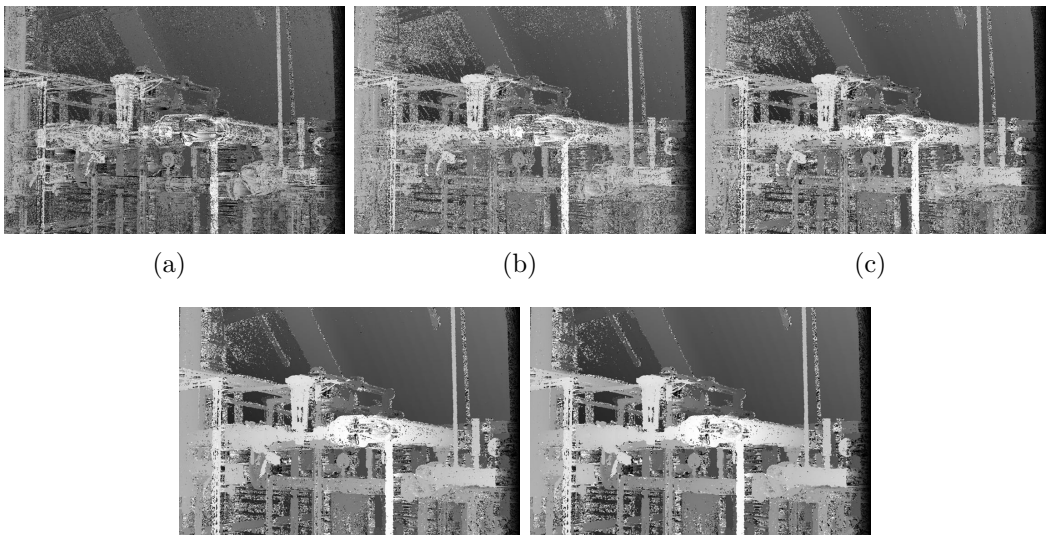
Figure 31: Motorcycle. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

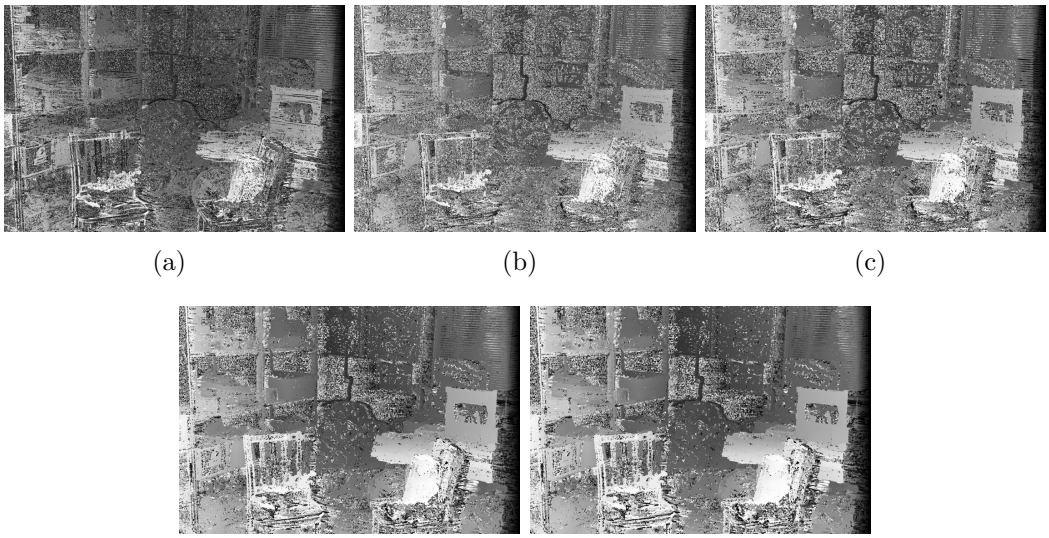
Figure 32: Piano. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

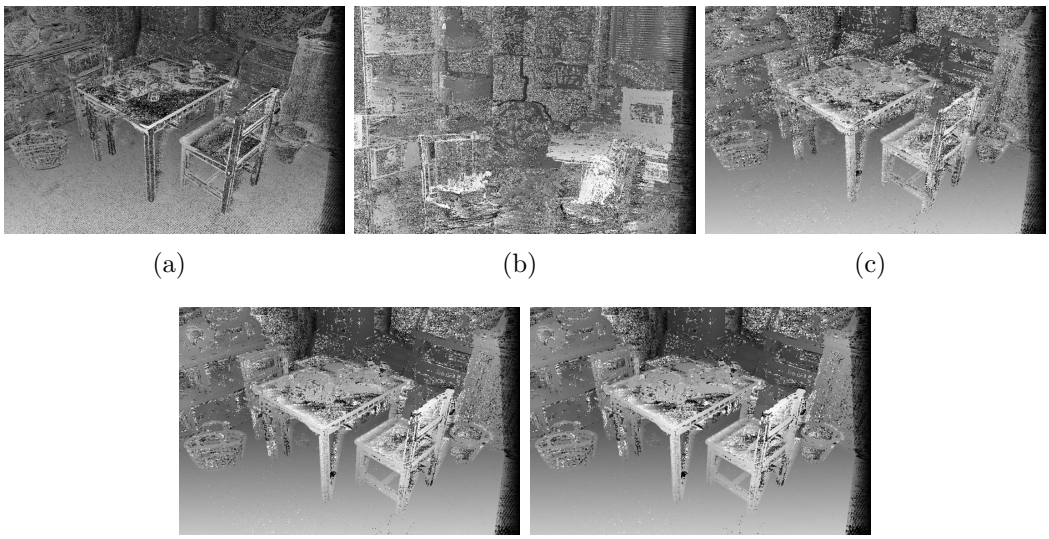
Figure 33: Pipes. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

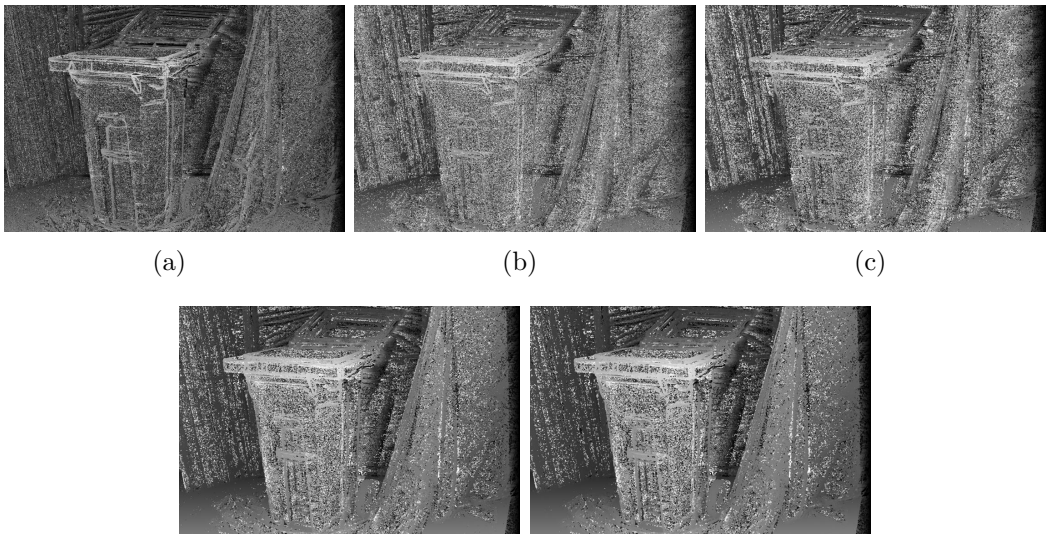
Figure 34: Playroom. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

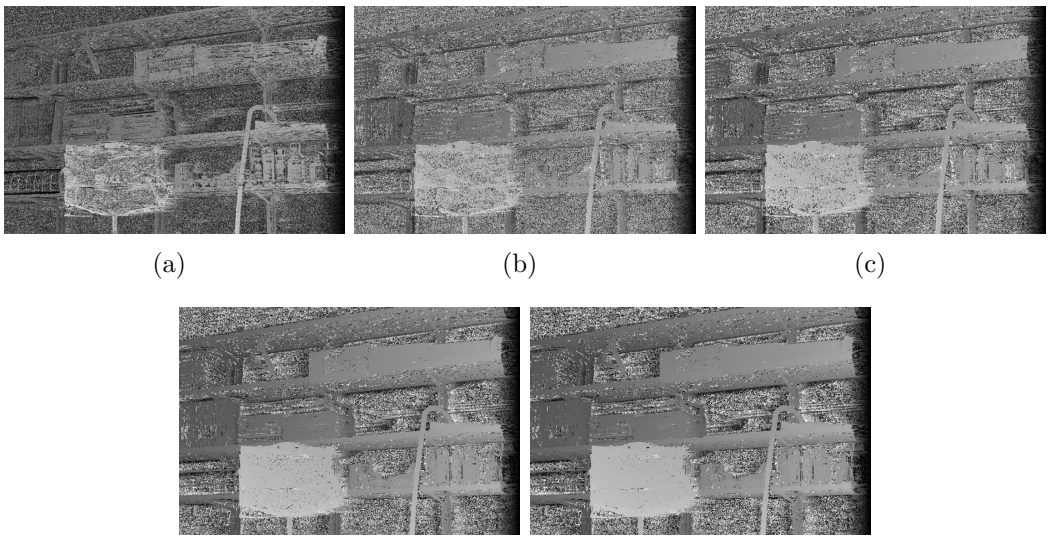
Figure 35: Playtable. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

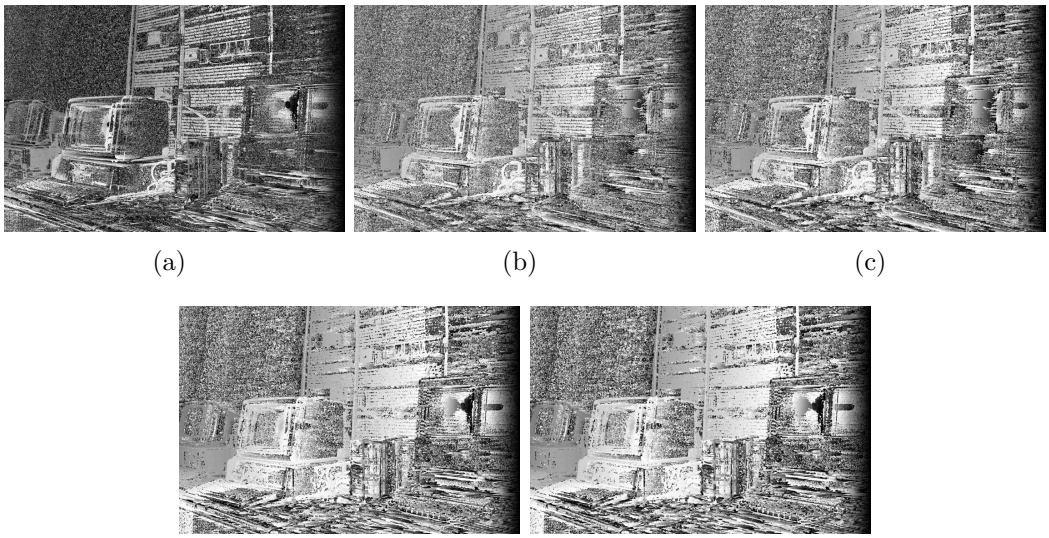
Figure 36: Recycle. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

Figure 37: Shelves. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.



(a) (b) (c)

(d) (e)

Figure 38: Vintage. (a) 7x7 CT + 1x1 search window. (b) 7x7 RT + 5x5 search window. (c) 7x7 RT + 7x7 search window. (d) 7x7 CRT + 5x5 search window. (e) 7x7 CRT + 7x7 search window.

Table 15: Different quality scores of the generated DMs using local block matching algorithm compared to orginal ground truth DMs.

Preprocessing transform	Search window	Median filter	PSNR (dB)	RMSE	*Accuracy (%)	**Accuracy (%)
7 x 7 RT	5 x 5	No	9.93	103.47	26.05	29.75
7 x 7 RT	7 x 7	No	10.27	100.13	31.36	34.91
7 x 7 CRT	5 x 5	No	11.30	90.88	41.26	45.21
7 x 7 CRT	7 x 7	No	11.57	88.75	44.13	48.00
7 x 7 CT	1 x 1	No	9.48	110.52	20.46	24.43
7 x 7 RT	5 x 5	Yes	10.82	96.09	28.89	32.98
7 x 7 RT	7 x 7	Yes	10.84	95.54	33.12	36.88
7 x 7 CRT	5 x 5	Yes	11.57	88.92	42.13	46.19
7 x 7 RT	7 x 7	Yes	11.75	87.44	44.67	48.63
7 x 7 CT	1 x 1	Yes	10.83	99.54	26.10	30.74

* Considers absolute disparity differences that are larger than 5 as incorrect.
** Considers absolute disparity differences that are larger than 10 as incorrect.

REFERENCES

[1] M. Ghanim, O. Tasdizen, and H. F. Ugurdag, "An efficient algorithm for disparity map compression based on spatial correlations and its low-cost hardware architecture." Manuscript submitted for publication, Oct., 2021.

[2] N. Qian, "Binocular disparity and the perception of depth," *Neuron*, vol. 18, no. 3, pp. 359–368, 1997. https://doi.org/10.1016/S0896-6273(00)81238-6.

[3] I. Hamzaoglu, O. Tasdizen, and E. Sahin, "An efficient H.264 intra frame coder system," *IEEE Trans. Consum. Electron.*, vol. 54, no. 4, pp. 1903–1911, 2008. https://doi.org/10.1109/TCE.2008.4711252.

[4] O. Tasdizen, A. Akin, H. Kukner, and I. Hamzaoglu, "Dynamically variable step search motion estimation algorithm and a dynamically reconfigurable hardware for its implementation," *IEEE Trans. Consum. Electron.*, vol. 55, no. 3, pp. 1645–1653, 2009. https://doi.org/10.1109/TCE.2009.5278038.

[5] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, pp. 151–158, 1994. https://doi.org/10.1007/bfb0028345.

[6] O. Demetz, D. Hafner, and J. Weickert, "The complete rank transform: A tool for accurate and morphologically invariant matching of structures.," in *Proc. Br. Mach. Vis. Conf. (BMVC)*, pp. 50.1–50.12, 2013. https://doi.org/10.5244/C.27.50.

[7] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, 2007. https://doi.org/10.1109/TPAMI.2007.1166.

[8] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs, "A maximum likelihood stereo algorithm," *Comput. Vis. Image Underst.*, vol. 63, no. 3, pp. 542–567, 1996. https://doi.org/10.1006/cviu.1996.0040.

[9] Z. Xie, S. Chen, and G. Orchard, "Event-based stereo depth estimation using belief propagation," *Front. Neurosci.*, vol. 11, p. 535, 2017. https://doi.org/10.3389/fnins.2017.00535.

[10] G. Tech, Y. Chen, K. Müller, J. R. Ohm, A. Vetro, and Y. K. Wang, "Overview of the multiview and 3D extensions of high efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 35–49, 2015. https://doi.org/10.1109/TCSVT.2015.2477935.

[11] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Fast 3D-HEVC depth map encoding using machine learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 3, pp. 850–861, 2019. https://doi.org/10.1109/TCSVT.2019.2898122.

[12] Q. Zhang, M. Chen, X. Huang, N. Li, and Y. Gan, "Low-complexity depth map compression in HEVC-based 3D video coding," *EURASIP J. Image Video Process.*, vol. 2015, no. 1, pp. 1–14, 2015. https://doi.org/10.1186/s13640-015-0058-5.

[13] H. Hamout and A. Elyousfi, "Fast depth map intra coding for 3D video compression-based tensor feature extraction and data analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 1933–1945, 2019. https://doi.org/10.1109/TCSVT.2019.2918770.

[14] S. Wang, L. Yu, and S. Xiang, "A low complexity compressed sensing-based codec for consumer depth video sensors," *IEEE Trans. Consum. Electron.*, vol. 65, no. 4, pp. 434–443, 2019. https://doi.org/10.1109/TCE.2019.2929586.

[15] L. F. Lucas, K. Wegner, N. M. Rodrigues, C. L. Pagliari, E. A. da Silva, and S. M. de Faria, "Intra-predictive depth map coding using flexible block partitioning," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4055–4068, 2015. https://doi.org/10.1109/TIP.2015.2456509.

[16] S. Lee and A. Ortega, "Adaptive compressed sensing for depthmap compression using graph-based transform," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, pp. 929–932, 2012. https://doi.org/10.1109/ICIP.2012.6467013.

[17] I. Tabus and P. Astola, "Sparse prediction for compression of stereo color images conditional on constant disparity patches," in *Proc. True Vision—Capture, Transmiss. Display 3D Video. (3DTV) Conf.*, pp. 1–4, 2014. https://doi.org/10.1109/3DTV.2014.6874766.

[18] M. Zamarin and S. Forchhammer, "Lossless compression of stereo disparity maps for 3D," in *Proc. IEEE Int. Conf. Multimed. Expo Work. (ICMEW)*, pp. 617–622, 2012. https://doi.org/10.1109/ICMEW.2012.113.

[19] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, 2000. https://doi.org/10.1109/83.855427.

[20] O. Palaz, H. F. Ugurdag, O. Ozkurt, B. Kertmen, and F. Donmez, "RImCom: raster-order image compressor for embedded video applications," *J. Signal Process. Syst.*, vol. 88, no. 2, pp. 149–165, 2017. https://doi.org/10.1007/s11265-016-1211-9.

[21] M. E. Papadonikolakis, A. P. Kakarountas, and C. E. Goutis, "Efficient high-performance implementation of JPEG-LS encoder," *J. Real-Time Image Process.*, vol. 3, no. 4, pp. 303–310, 2008. https://dx.doi.org/10.1007/s11554-008-0088-7.

[22] T. Inatsuki, M. Matsuura, K. Morinaga, H. Tsutsui, and Y. Miyanaga, "An FPGA implementation of low-latency video transmission system using lossless

and near-lossless line-based compression," in *Proc. IEEE Int. Conf. Digit. Signal Process. (DSP)*, pp. 1062–1066, 2015. https://doi.org/10.1109/ICDSP.2015.7252041.

[23] K. Mühlmann, D. Maier, J. Hesser, and R. Männer, "Calculating dense disparity maps from color stereo images, an efficient implementation," in *Proc. IEEE Work. Stereo Multi-Baseline Vision (SMBV)*, vol. 47, pp. 30–36, 2001. https://doi.org/10.1109/SMBV.2001.988760.

[24] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, pp. 31–42, 2014. https://doi.org/10.1007/978-3-319-11752-2_3.

[25] M. Grabner, "jpeg_encode.zip," *MATLAB File Exchange Center*, 2021. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/46424-jpegls_encode-zip. Accessed: Nov. 20, 2021.

[26] S. Garg, "Lossless compression software for camera raw images." [Online]. Available: http://imagecompression.info/lossless/, 2015. Accessed: Nov. 20, 2021.

[27] Xilinx, "Xilinx synthesis technology (XST) user guide." [Online]. Available: https://shorturl.at/glpsH, n.d. Accessed: Nov. 20, 2021.

[28] T. Schmitz, "The rise of serial memory and the future of DDR," *Xilinx UltraScale Devices. White Pap.*, pp. 1–9, 2015. [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp456-DDR-serial-mem.pdf. Accessed: Nov. 20, 2021.

[29] H. Li and K. N. Ngan, "Learning to extract focused objects from low dof images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 11, pp. 1571–1580, 2011. https://doi.org/10.1109/TCSVT.2011.2129150.

VITA

Name: Mustafa Ghanim

E-mail: ghanim.mustafa@gmail.com

Education

Özyeğin University, Turkey *September 2019 - Present*

MS in Electrical and Electronics Engineering. GPA: 4.00/4.00

Research Areas: FPGA, Efficient Hardware Architectures, Image/Video Processing, Computer Vision and AI, Digital Signal Processing.

Thesis : An Efficient Algorithm for Disparity Map Compression Based on Spatial Correlations and Its Low-cost Hardware Architecture.

Istanbul Technical University, Turkey *August 2015 - August 2019*

BS in Electronics and Communications Engineering. GPA: 3.71/4.00

Ranked in Faculty High Honor List.

Thesis : Design and Implementation of FPGA based Quadrotor Controller.

Work Experience

Özyeğin University, Turkey

Teaching and Research Assistant: *September 2019 - Present*

- Research Assistant at nEMESysLab (Embedded Systems/Microelectronic Lab)
- Teaching Assistant in courses of Physics Lab and Digital Systems.

AntSis Electronics, Turkey

Research and Development Engineering Intern: *June 2019 - August 2019*

- FPGA based systems simulation and verification.
- Video processing with Microblaze using Vivado Block Design and SDK.
- Projects debug with VHDL and FMC chip integration on FPGA Ultra-scale Boards.

Arçelik Cooking Appliances, Turkey

Research and Development Engineering Intern: *July 2018 - August 2018*

- Electrical safety tests for induction hobs.
- Design and simulation of power electronic circuits.
- Performance and power optimization (R&D).
- Involved in the initial phase of the development of next Gen induction hobs.

ITU Embedded Systems Labs, Turkey

Research and Development Engineering Intern: *June 2017 - July 2017*

- C/C++ embedded Linux programming.
- Python OpenCV based programming.
- Embedded systems communications.
- Face recognition with Intel Galileo.

Technical Skills

Programming Languages	C, C#, C++, Python, Verilog/VHDL, Assembly
Software & Tools	Latex, Excel Solver, MATLAB, Simulink, HDL Coder
	ISE/Vivado, Linux, HFSS, LTspice, Cadence

Projects

Digital Hardware and Embedded Systems:

- Design and real-time verification of 128-bit multiplication operations through UART on Spartan-3E.
- Mouse, keyboard, 7-segment display, and VGA control on Spartan-3E.
- Design of 16-bit RISC processor by semi-custom ASIC tools with different memory technologies and verifying the design on FPGA.
- Implementation of division, multiplication, and cryptography-based functions for high-performance processors.
- Conductivity-based fruit detecting embedded system on Arduino.
- Parallel-serial-parallel communication system with Picoblaze soft-core processor on FPGA.
- Bidirectional and multi-step 32-bit input rotating program at Picoblaze assembly environment (8-bit).
- Micro-controller based ultra-sonic radar design.

Analog Circuits:

- Design and simulation of Op-Amp with high-gain and high-bandwidth.

Machine Learning and Signal Processing:

- Colorizing gray scale images using Conventional Neural Networks.
- Sound separation using Non-negative Matrix Factorization.
- Music transcription with machine learning.
- Image-based gender and emotional state classification with machine learning.
- Voice-Activity Detection with learning from filter bank features.

- Multi-object classification based on SIFT and BoW methods.
- Economic dispatch and environmentally friendly unit commitment optimization for multiple electric generators.
- Momentum detection for S&P 500 stock market.

RF Engineering:

- Ultra-wide band microstrip antenna design.

Languages

- Arabic: Native
- English: Working proficiency, IELTS Academic average score: 7.00
- Turkish: Fluent
- German: Beginner (A1.1)

Academic Achievements

Awards: Anemone Culture Association recognition award (1st rank in bachelor studies, 2017).

Publications: Akçay, L., Göncü, E., Esen, M. M., Uslu, S., Ghanim, M., Yolcu, N. B., Öztürk, H. S., Sagman, M. B., Hüner, Y., Karakaslı, G., Gayretli, M. G., & Yalçın, B. Ö. (2019, October). Farklı CMOS süreçlerinde RISC işlemci tasarımı ve ASIC gerçeklemesi [Design and implementation of RISC processor with different CMOS technologies]. Poster presented at ITU Processor Design Workshop, Istanbul. doi: http://dx.doi.org/10.13140/RG.2.2.34502.83524.

Under Review: Ghanim, M., Tasdizen, O., Ugurdag, H. F. (2021). *An efficient algorithm for disparity map compression based on spatial correlations and its low-cost hardware architecture.* Manuscript submitted for publication.

Service: Manuscript reviewer at *Digital Signal Processing: A Review Journal.*