Mitigating the Performance and Quality of Parallelized Compressive Sensing Reconstruction Using Image Stitching

Mahmoud Namazi
mnamazi@ucsb.edu
University of California, Santa Barbara

Hosein Mohammadi Makrani
hmohamm8@gmu.edu
George Mason University

Zhi Tian
ztian1@gmu.edu
George Mason University

Setareh Rafatirad
srafatir@gmu.edu
George Mason University

Mohamad Hosein Akbari
mohammad.akbari@student.sharif.ir
Sharif University of Technology

Avesta Sasan
asasan@gmu.edu
George Mason University

Houman Homayoun
hhomayou@gmu.edu
George Mason University

ABSTRACT

Orthogonal Matching Pursuit is an iterative greedy algorithm used to find a sparse approximation for high-dimensional signals. The algorithm is most popularly used in Compressive Sensing, which allows for the reconstruction of sparse signals at rates lower than the Shannon-Nyquist frequency, which has traditionally been used in a number of applications such as MRI and computer vision and is increasingly finding its way into Big Data and data center analytics. OMP traditionally suffers from being computationally intensive and time-consuming, this is particularly a problem in the area of Big Data where the demand for computational resources continues to grow. In this paper, the data-level parallelization of OMP through blocking is examined. Traditionally blocking has been used to accelerate the performance of OMP reconstruction for big data image analytics. However, as we show in this work, blocking, particularly in the form of vectorizing, introduces significant error in terms of PSNR and SSIM index in the reconstruction quality. In response, we deploy the concept of stitching to recover the lost accuracy. We further examine the influence of the level of blocking and amount of stitching (overlap between each block) with regard to reconstruction time and reconstructed image quality. While stitching boosts up the image reconstruction accuracy significantly, the object detection count results show anywhere from 11.84% to 140.54% improvement, depending on the cases being compared, it introduces significant overhead with regard to reconstruction time. To address the overhead, we deploy hardware accelerated base solutions. Given the emergence of hardware accelerators in data centers and for big data analytics in form of FPGAs, our solution effectively utilizes this resource to enhance the performance overhead of stitching by 25%. We show the minimum block size required for an FPGA speed-up.

KEYWORDS

OMP; Compressive Sensing; Acceleration; Big Data; Cloud Computing; Data Centers

ACM Reference Format:
Mahmoud Namazi, Hosein Mohammadi Makrani, Zhi Tian, Setareh Rafatirad, Mohamad Hosein Akbari, Avesta Sasan, and Houman Homayoun. 2019. Mitigating the Performance and Quality of Parallelized Compressive Sensing Reconstruction Using Image Stitching. In *Great Lakes Symposium on VLSI 2019 (GLSVLSI '19), May 9–11, 2019, Tysons Corner, VA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3299874.3317991

1 INTRODUCTION

In recent years, Orthogonal Matching Pursuit (OMP) has emerged as one of several methods used to reconstruct sparse signals from random measurements [1]. OMP is one of several methods utilized in Compressive Sensing (CS), a novel set of techniques which enable the reconstruction of sparse signals sampled at sub-Nyquist rates [2]. By requiring fewer samples, CS requires sensors to run for the less time and use less energy, a great benefit in applications such as MRI imaging [3], radar, and 360° cameras. OMP serves as one of many reconstruction algorithms in CS, others include basis pursuit and total-variation minimization.

CS is used in a wide variety of applications where signals need to be processed with as few samples as possible. By projecting the captured signal to a lower dimensional space, chosen by the measurement matrix, it allows one to sample the signal at a lower rate than what the Shannon-Nyquist Theorem would predict. Additionally, it allows the signal to be compressed to larger degree. The trade-off is that decompressing the signal takes a significant amount of time and computational power. One such application is in 3D 360° cameras [4]. In this application, signals need to be high resolution and hence require a wide bandwidth. Both higher compressibility and faster imaging times are needed. By requiring fewer samples and doing both the compression and sampling in a single step, compressive sensing fulfills two highly sought-after

GLSVLSI '19, May 9–11, 2019, Tysons Corner, VA, USA

ACM ISBN 978-1-4503-6252-8/19/05…$15.00
https://doi.org/10.1145/3299874.3317991

criterion in this application. Current issues that exist are that reconstruction can be time-consuming and require large amounts of memory. Currently, however, faster imaging times can be acquired through the parallelization of compressive sensing.

In cloud computing and big data analytics, CS has been used to reduce active memory and speedup analytics algorithms in a scalable fashion [5]. In many types of big data queries, particularly high-value statistics, approximate results can suffice. CS can be used to query high-dimensional sparse data, essentially providing a compressed snapshot of the data. Since CS can go below the Shannon-Nyquist frequency with regard to sampling rates, this leads to less active memory, IO, and traffic volume. However, faster sampling times are needed as data center volumes continue to grow. Additionally, as data center memory volumes continue to grow, this also places pressure on CS from the aspect of memory usage, as the dictionary, a matrix which is used to compress the original data, grows with the size of the data.

Although OMP, in the CS context, has the advantage of requiring fewer samples, the drawback is that CS reconstruction is time-consuming and computationally expensive. There are trade-offs between OMP and other CS reconstruction techniques. Another algorithm, l_1-minimization, which OMP serves as an approximation for, is better in terms of accuracy, but is even more time-consuming and difficult to implement.

To address the challenge of slow reconstruction time, this paper first examines the blocking method for OMP. In blocking, an image signal is split into approximately equal-sized block. Each block is then sampled and reconstructed separately. Once reconstruction is complete, the blocks are then put back together in order to recreate the original image. One form of blocking is called vectorizing, this is where vertical columns of pixels are used as the block instead of square-like sections of the image. One unique contribution of this paper is in clarifying the relationship between block size and reconstruction time, as well as reconstruction quality. We show that while blocking improves performance with regard to reconstruction time, it introduces reconstruction quality loss, in response to which we introduce stitching. In stitching, an image is blocked such that there is overlap between blocks, in order to create redundancy and, in theory, improve image quality after reconstruction. The differences between blocking, vectorizing, and stitching are further clarified in Figure 1.

In our analysis, we examine the trade-off between reconstruction quality and reconstruction time when stitching and blocking are implemented. Last, we implement a parallelized version of the OMP algorithm [6] on an FPGA and examine the minimum block size where an FPGA would be faster than a CPU. To the best of our knowledge, this is the first paper that thoroughly analyzes blocking, including vectorizing, and deploys stitching as a response to the reduction of reconstruction quality incurred by blocking. By clarifying the impact of blocking and stitching, we hope to clear the way for faster and better-quality CS-based OMP image reconstruction.

2 BACKGROUND AND RELATED WORK

Orthogonal Matching Pursuit (OMP) is an iterative greedy algorithm, which allows for the sparse approximation of a function, given an over-complete dictionary [7]. OMP can be used to perform signal recovery from random measurements [1]. The basic theory that lies behind compressive sensing and the one OMP is focused on solving is Equation 1.

Figure 1: The concept of blocking is illustrated on the left where the the image is divided into blocks which can be processed separately. Stitching is illustrated by extending each solid line into the dotted lines, creating overlap with the adjacent block. Vectorizing is illustrated on the right, the blocks become rectangular rather than square.

$$y = \phi x \tag{1}$$

In this equation, ϕ is a matrix of dimension M x N, where M < N is the number of measurements and N is the size of the signal, x. Multiplying ϕ and x yields y which is referred to as the measured signal and is of length M. It is the projection of ϕ onto x. The goal of compressive sensing is to recover x, given y and ϕ. According to compressive sensing, if y has a sparse representation recovery of the original signal is possible, even if the sampling rate is lower than that called for by the Shannon-Nyquist Theorem. Reconstruction involves finding the best solution to the undetermined system of linear equations created by $y = \phi x$. The computational complexity of this problem comes most significantly from solving the least square problem. Previous works, as described below, have tried to solve this problem through a variety of methods and can be divided into algorithm improvements [8, 9], parallel architecture implementations [10, 11], and hardware/software co-design [6, 12, 13, 9].

Kulkarni et.al. [6] present a parallelization where the OMP algorithm is split into three kernels: dot product, sort, and least squares. Each of these kernels is then parallelized on an FPGA in order to reduce execution time. The design is implemented on a Xilinix Virtex 7 FPGA. It is shown that for a 128x128 image, the proposed reconfigurable architecture gives a 1.8x to 2.67 speedup. Alternative parallelization methods, such as blocking, are not proposed in this paper, although vectorizing (processing each column of the image separately) is used. In [14], Kulkarni and Mohsenin carry out an analysis of OMP on 7 different implementations. They design and implement a custom 61-core solution which gives a 24x speedup when compared to CPU implementations.

In [10], OMP is implemented on an Application Specific Instruction-set Processor (ASIP). This implementation targets portable devices where operation is expected to be over an extended period of time with limited energy resources. The paper shows trade-offs in power and performance. The CS processor, which has sub-V_t latch-based memories, operates at 0.37 V with a clock frequency of 100 KHz

and total power of 288nW. A 62x speedup is achieved versus OMP running on a baseline low-power processor.

In [12], OMP is implemented on a GPU. Fujimoto's matrix-vector multiplication algorithm is adopted for the projection module and the matrix-inverse-update algorithm is adopted for the least-squares module. It was shown that a 40x speedup can be obtained with a GTX480 GPU on an Intel Core i7 CPU. This paper does not explore blocking or stitching as alternative parallelization methods.

Sermwuthisarn et.al. [8] present an analysis of the parallelization of OMP with blocking. They complete reconstruction in MATLAB using blocking and stitching (where the blocks overlap) on 64 x 64 size images. This implementation only includes 8 x 8 blocks, so no analysis is done on differing block size and differing amounts of overlap between blocks in stitching. A total of 3 images are analyzed, making it difficult to draw any conclusive results. Additionally, no hardware alternatives are presented in this work.

In [13], OMP is implemented using Q-R decomposition for the matrix inverse kernel. A new algorithm is implemented for finding the inverse square root of fixed-point numbers. The entire OMP algorithm is implemented on a Xilinix FPGA Virtex-5 and used to reconstruct a 256-length signal. At the time of publication, this implementation was 2.4-times faster than the state-of-the-art implementation. This implementation includes advances in the algorithm, as well as a hardware implementation in order to best optimize the reconstruction time. Similarly, Yang et al. [9], present an optimized OMP algorithm where Cholesky decomposition is used to solve the LS problem in a highly parallel-friendly manner. Furthermore, they implement their algorithm on a GPU implementation and achieve a 38x speedup as compared to using a CPU with LAPACK for solving the LS problem.

Rabah et al. [11] present an algorithm where Cholesky decomposition and a modified Newton-Raphson method are used to solve the linear problem, which although not novel, is then optimized for hardware implementation through Simulink. The design is implemented on a Virtex-6 FPGA and achieves the best seen results with a reconstruction time of 0.34 μs and PSNR of 38.9 dB with a signal length of 1024, all while having a reconfigurable architecture.

In this work, we primarily analyze the consequences of changes in the OMP algorithm, through blocking and stitching. Examining varying block sizes, varying levels of overlap between blocks, and their effect on the reconstruction time and image quality has not been previously examined and is the primary contribution of this article. This paper also presents the minimum block size required in order for an FPGA to provide an improvement in time over a CPU.

3 ALGORITHM

The OMP algorithm requires two inputs, a measurement matrix (ϕ) and a measured signal (y). At each iteration (t), the dot product of the transpose of ϕ and y is taken in order to find the column in ϕ most correlated with y. The set of most correlated columns form a matrix which is used to obtain a new estimate for the signal x_t. After k iterations, we arrive at a final approximation for the signal. The algorithm is explained step-by-step in Algorithm 1. In the context of compressive sensing, ϕ is formed by the matrix multiplication of a sensing matrix and a representation matrix, which are used to sample the image and transform the sampled matrix to a sparse domain.

Algorithm 1 OMP Reconstruction Algorithm

1: Initialize $R_0 = y$, $\phi_0 = \emptyset$, $\Lambda_0 = \emptyset$, $\Phi_0 = \emptyset$ and $t = 0$
2: Find Index $\lambda_t = max_{j=1...n}$ subject to $| < \phi_j R_{t-1} > |$
3: Update $\Lambda_t = \Lambda_{t-1} \bigcup \lambda_t$
4: Update $\Phi_t = [\Phi_{t-1}\ \phi_{\Lambda_t}]$
5: Solve the Least Squares Problem
$x_t = \min_x || y - \Phi_y\ x||^2$
6: Calculate new approximation: $\alpha_t = \Phi_t\ x_t$
7: Calculate new residual: $R_t = y - \alpha_t$
8: Increment t, and repeat from step 2 if t<k
After all the iterations, we can find correct sparse signals.

4 PROPOSED WORK

n this work, we seek to explore the data-level parallelization of OMP in order to assuage problems with high reconstruction time and large memory requirements, due to the size of the dictionary when attempting to reconstruct the entirety of an image at once. Rather than exploring the speeding up of subroutines and instruction-level parallelism, which has been previously explored on a variety of hardware [6, 10, 12], we explore the blocking of the input image (before sampling), i.e. data-level parallelism. First, we explore blocking. In blocking, the image is split into m non-overlapping blocks. Depending on m, the blocks may be square or rectangular. The case where the blocks are adjacent vertical columns is called vectorizing. The individual blocks are sampled separately, using a random Gaussian matrix as the measurement matrix. For the representation basis, we use discrete cosine wavelets. The blocks are then reconstructed sequentially, and the original image is rebuilt, with each block being placed in its original location. With stitching, overlap is introduced between blocks. We examine stitching with overlaps of 2 and 5 pixels between adjacent blocks. We then average over overlapping sections. After reconstruction of the individual blocks, the entire image is made by averaging over the overlapping sections between blocks. In theory, this should improve the reconstructed image quality by creating a level of redundancy which can be exploited to improve what may be a poor reconstruction in one of the blocks.

By blocking the image, we are creating a series of local l_1-minimization problems as opposed to a larger global one. There are no guarantees that solving the l_1-minimization problem for a small subsection of an image will give the same solution if we solved the global problem and then referred to the same subsection.

We investigate image quality through two measures. First, we examine the Peak Signal-to-Noise Ratio (PSNR) which is a measure of the ratio of maximum signal power to noise power, in decibels. In a sense, the PSNR is a local measurement since it measures how the highest intensity regions of the image compare to the amount of noise. Finally, we examine the SSIM index which is a method for measuring the similarity between two images. Since the SSIM index is calculated over the entire image, it gives a more blanket metric for the entirety of the image. This gives a more global metric for comparison of the reconstructed image to the original.

In order to further analyze the quality of each reconstruction, we count the number of objects in each image through region boundary tracing. We then carry out the same analysis on each reconstructed

image and compare the number of objects in the original image to the number of objects in the reconstructed image. We examine the percent difference of images in the reconstructed image versus the original and make comparisons between blocking and stitching.

Finally, we implement OMP on a Xilinix Zynq-7000 SoC and derive an equation for the number of cycles required to process an input vector with N components (in bytes). We find the number of blocks required to achieve a speedup for hardware acceleration compared to software.

5 IMPLEMENTATION RESULTS

5.1 Quality and Time Results

The experiments were carried out on a server-class Xeon E5-2670 (2.60Ghz) processor. Due to large RAM constraints, brought on by the measurement matrix and the reconstruction process, we constrained our biggest block size to 32 x 32 with 256 x 256 images. The OMP algorithm was implemented in C. We acquired a large image database from [15]. The quality and time and results were averaged over 10 trials of each of the images. The reconstruction time and quality metric results can be seen in Tables 1 and 2, respectively. Figure 2 plots both the time and PSNR results for each blocking configuration.

Table 1: Time Results

Number of blocks	Reconstruction Time (seconds)
64	120.280
256	9.508
1024	6.563
4096	4.492
64 (vectorized)	120.262
256 (vectorized)	9.472
1024 (vectorized)	6.532
4096 (vectorized)	4.471
64, 2 pixel overlap	197.25
256, 2 pixel overlap	23.14
256, 5 pixel overlap	79.32
1024, 2 pixel overlap	11.86

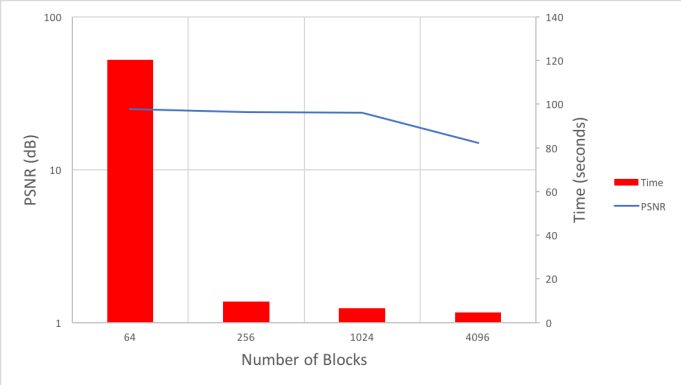

Figure 2: PSNR and Time vs. Number of Blocks

Table 2: Quality Results

Number of blocks	PSNR	SSIM Index
64	24.991	0.628
256	23.871	0.629
1024	23.591	0.660
4096	14.943	0.368
64 (vectorized)	22.792	0.462
256 (vectorized)	24.113	0.578
1024 (vectorized)	21.368	0.486
4096 (vectorized)	14.573	0.263
64, 2 pixel overlap	26.990	0.628
256, 2 pixel overlap	25.111	0.669
256, 5 pixel overlap	26.850	0.726
1024, 2 pixel overlap	25.904	0.629

Several trends are observed in the figure. It can be seen that as the number of blocks increases, the reconstruction time per block decreases exponentially. This is as expected, since less iterations of the OMP algorithm are needed to finish reconstruction. Additionally, Λ contains less columns, and so, steps 6 and 7, as outlined in Algorithm 1, require less computation. With larger block sizes, each step of the OMP algorithm takes progressively longer time as the size of Λ increases and more computation is required at each step.

Next, we observe interesting trends in the PSNR. It can be seen that as the block size decreases, the PSNR generally decreases. Although the changes are not as drastic as the change in reconstruction time per block, it can be seen that at 4096 blocks there is a large drop in the PSNR. We can observe that at these block sizes, a large amount of noise appears to be generated in the process of reconstruction. However, there is a very gradual drop going from a single block to greater block sizes. Vectorizing appears, in most of the cases, appears to give poor PSNR results. The only vectorizing case that has competitive results is the 256 case, this may be due to the fact that since the image itself is 256 x 256, features are scaled to the height of the image and therefore are more amenable to compression and reconstruction in this case.

The SSIM index trend follows the PSNR. In general, it appears that larger blocks give slightly better reconstruction results, with respect to quality. Although the SSIM index of 1024 blocks is the best, this can likely be attributed to it being particularly amenable to this dataset. Nonetheless, the general trend holds that less, larger blocks give better quality results.

Next, we examine the results from having overlap between blocks and averaging over these overlapping sections, referred to as stitching. The results can again be seen in Tables 1 and 2.

We observe that the reconstruction times increase significantly, which would be expected since each block is larger. Additionally, there is an averaging process which was not present before with blocking.

With regard to quality, it is clear that even a small amount of stitching increases the quality of the image reconstruction. This can be seen when comparing the stitching results to their blocking counterparts. This increase in quality is seen across both the PSNR and SSIM index results.

5.2 Object Detection Analysis

The experiments with regard to object detection were carried out separately from the quality and time experiment. These experiments were carried out on a Xeon E5-2670 processor, as well. Matlab was used to count the number of objects in each image. The results were averaged over 10 trials for each image in the database. The object detection count of the reconstructed image was compared to that of the original image and the percent difference was found. The percent difference was averaged over the image database. These results can be seen in Table 3 and Figure 3.

Table 3: Object Detection Results

Number of blocks, overlap	Percent Difference Object Count (%)
64	68.66
256	68.53
1024	59.83
4096	181.71
64 (vectorizing)	95.94
256 (vectorizing)	140.61
1024 (vectorizing)	160.84
4096 (vectorizing)	555.23
64, 2 pixel overlap	41.59
256, 2 pixel overlap	47.99
256, 5 pixel overlap	41.17
1024, 5 pixel overlap	42.26

It can be seen in Table 3, similar to the quality results, that a greater number of blocks leads to poorer reconstruction results when the reconstruction is compared to the original image. It is likely that the large number of blocks causes artifacts to be created between blocks when they are put in their proper position. These results further elucidate the relationship between the number of blocks and image quality. Vectorizing again has poor results, similar to the SSIM index results shown in Table 2. The stitching results appear to be much better. This is due to the averaging effect that occurs between blocks in stitching. The averaging improves bad reconstruction results in any one particular block and smooths over the artificially created objects and artifacts.

5.3 FPGA Implementation Results

There are three possible solutions for reducing the time overhead incurred by stitching in a cloud computing environment. These solutions include more CPU nodes, which is not cost effective, introducing GPUs, which can be power hungry in cloud usage, and finally FPGAs which have been shown to be most effective in the cloud computing domain, including use in Microsoft's Project Catapult. Here, we show a proof of concept using a particular Zynq platform, however, the analysis and methods shown can be applied to other FPGAs. The proposed FPGA implementation was carried out on a Zedboard development kit, using the Xilinx Zynq-7000 SoC. The measured signal input is the constricting factor in the algorithm, so this analysis is based on the number of bytes in this particular input. A variety of image sizes were reconstructed, with a maximum measured signal input of 40 bytes, which corresponds to a vector size of 10 components when each component is 32 bits.

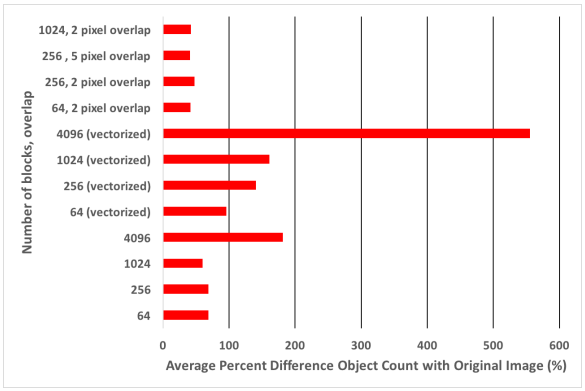

Figure 3: Impact of differing number of blocks and overlap on the percent difference in number of objects detected when compared to the original image

Through experiment, we observed that the DMA time is 500 - 530 cycles. The transfer time for each byte from the ARM processor to the FPGA, on this particular board, is 2 cycles. By adding together the DMA time, the transfer time, and the time it takes for the FPGA to do the relevant matrix multiplication operations, we get equation 2, where N is the total number of samples multiplied by the number of bytes it takes to represent each component in the input vector.

$$\text{Time (in cycles)} = \text{DMA Time} + 2N + (N/8) + N \quad (2)$$

Due to the limited amount of memory on the Zedboard (89.15% of LUT used when the measured signal was 40 bytes, as shown in Table 6), we can use this equation to extrapolate the reconstruction time of a larger measured signal.

Table 4: Resource utilization of HW+SW acceleration

Resources	Available	Utilization (%)
BRAM	280	77.35
DSP48	220	34.86
FF	106400	8.54
LUT	53200	89.15

By comparing this extrapolation to measured reconstruction times from running the algorithm solely on the ARM processor, we are able to approximate the measured signal size at which the hardware implementation outperforms the software implementation, with regard to reconstruction time. This can be seen in Figure 4.

Based on Figure 4, it can be seen that the break-even point occurs at approximately a measured signal size of 350 bytes for the Zynq platform, when the software implementation outperforms the hardware implementation with regard to reconstruction time. If we make assumptions on the size of each component in the measured dataset, we can derive the minimum size with regard to block size that would be required for a hardware speedup. For example, if we assume each component in the measured dataset has a size of 4 bytes, that they are sampled at a rate of 50%, then we would gain a hardware speedup at a granularity of 256 blocks or less for a 256 x 256 image. This speedup is restricted only to the algorithm-level

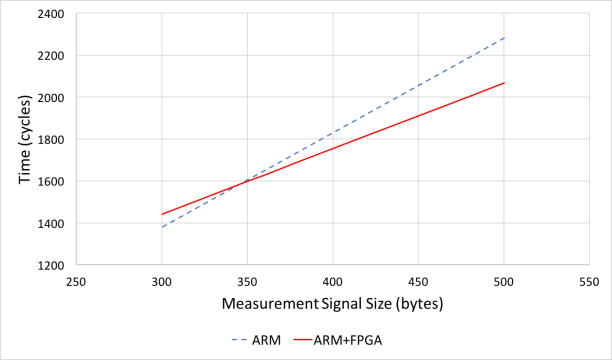

Figure 4: Impact of the size of the measurement signal input on time it takes for algorithm to complete

parallelization. At the data-level, there would also be a speedup proportional to the number of parallel instances involved as each block could be processed separately.

6 CONCLUSION

This paper presents the first thorough examination of blocking and stitching data-level parallelization methods for the OMP compressive sensing reconstruction algorithm, including varying block sizes and degrees of overlap. The accuracy and reconstruction time of the images are examined across different block sizes, as well as across different amounts of overlap between adjacent blocks in the case of stitching. The results indicate that while blocking reduces computational time when compared to reconstructing an entire image at once, it incurs a modest reduction in image quality. Vectorizing, a commonly used method for parallelization of OMP is shown to give poor global image quality results when compared to square blocking. It is shown that stitching improves PSNR by up to several decibels and the SSIM index by up to 15% when compared to blocking. Similarly, in the object count analysis, there is a 11.84% to 140.54% improvement, based on the cases that are compared. Finally, it is shown that parallel architectures can be utilized in order to reduce the overhead of stitching. While blocking gives large benefits with regard to reconstruction time and reduction in needed RAM, stitching is better from an image quality perspective. We show that this overhead can be reduced by up to 25% using FPGAs for blocks past a particular block size.

7 ACKNOWLEDGMENTS

This work was supported in parts by the National Science Foundation under grant CSR-1526913.

REFERENCES

[1] Joel A. Tropp and Anna C. Gilbert. 2007. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53, 12, 4655–4666. DOI: 10.1109/tit.2007.909108. https://doi.org/10.1109/tit.2007.909108.

[2] E.J. Candes, J. Romberg, and T. Tao. 2006. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52, 2, 489–509. DOI: 10.1109/tit.2005.862083. https://doi.org/10.1109/tit.2005.862083.

[3] Michael Lustig, David Donoho, and John M. Pauly. 2007. Sparse mri: the application of compressed sensing for rapid mr imaging. *Magnetic Resonance in Medicine*, 58, 6, 1182–1195. DOI: 10.1002/mrm.21391. https://doi.org/10.1002/mrm.21391.

[4] Ahmed Nabil Belbachir, Stephan Schraml, Manfred Mayerhofer, and Michael Hofstatter. 2014. A novel hdr depth camera for real-time 3d 360 degree panoramic vision. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. DOI: 10.1109/cvprw.2014.69. https://doi.org/10.1109/cvprw.2014.69.

[5] Jiaxing Zhang, Ying Yan, Liang Jeff Chen, Minjie Wang, Thomas Moscibroda, and Zheng Zhang. 2014. Impression store: compressive sensing-based storage for big data analytics. In *HotCloud*.

[6] Amey M. Kulkarni, Houman Homayoun, and Tinoosh Mohsenin. 2014. A parallel and reconfigurable architecture for efficient omp compressive sensing reconstruction. In *Proceedings of the 24th edition of the great lakes symposium on VLSI - GLSVLSI 14*. ACM Press. DOI: 10.1145/2591513.2591598. https://doi.org/10.1145/2591513.2591598.

[7] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. 1993. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*. IEEE Comput. Soc. Press. DOI: 10.1109/acssc.1993.342465. https://doi.org/10.1109/acssc.1993.342465.

[8] Parichat Sermwuthisarn, Supatana Auethavekiat, and Vorapoj Patanavijit. 2009. A fast image recovery using compressive sensing technique with block based orthogonal matching pursuit. In *2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. IEEE. DOI: 10.1109/ispacs.2009.5383863. https://doi.org/10.1109/ispacs.2009.5383863.

[9] Depeng Yang, Gregory. D. Peterson, and Husheng Li. 2012. Compressed sensing and cholesky decomposition on fpgas and gpus. *Parallel Computing*, 38, 8, 421–437.

[10] Jeremy Constantin, Ahmed Dogan, Oskar Andersson, Pascal Meinerzhagen, Joachim Rodrigues, David Atienza, and Andreas Burg. 2013. An ultra-low-power application-specific processor with sub-vt memories for compressed sensing. In *VLSI-SoC: From Algorithms to Circuits and System-on-Chip Design*. Springer Berlin Heidelberg, 88–106. DOI: 10.1007/978-3-642-45073-0_5. https://doi.org/10.1007/978-3-642-45073-05.

[11] Hassan Rabah, Abbes Amira, Basant Kumar Mohanty, Somaya Almaadeed, and Pramod Kumar Meher. 2015. Fpga implementation of orthogonal matching pursuit for compressive sensing reconstruction. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23, 10, 2209–2220.

[12] Yong Fang, Liang Chen, Jiaji Wu, and Bormin Huang. 2011. Gpu implementation of orthogonal matching pursuit for compressive sensing. In *2011 IEEE 17th International Conference on Parallel and Distributed Systems*. IEEE. DOI: 10.1109/icpads.2011.158. https://doi.org/10.1109/icpads.2011.158.

[13] Jerome L.V.M. Stanislaus and Tinoosh Mohsenin. 2012. High performance compressive sensing reconstruction hardware with qrd process. In *2012 IEEE International Symposium on Circuits and Systems*. IEEE. DOI: 10.1109/iscas.2012.6271921. https://doi.org/10.1109/iscas.2012.6271921.

[14] Amey Kulkarni and Tinoosh Mohsenin. 2015. Accelerating compressive sensing reconstruction omp algorithm with cpu, gpu, fpga and domain specific many-core. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. DOI: 10.1109/iscas.2015.7168797. https://doi.org/10.1109/iscas.2015.7168797.

[15] [n. d.] In http://decsai.ugr.es/cvg/dbimagenes/g256.php.