Mixed Precision $l_1$ Solver for Compressive Depth Reconstruction: An ADMM Case Study

Citation for published version:

Link:
Link to publication record in Heriot-Watt Research Portal

Document Version:
Peer reviewed version

Published in:
2021 International Workshop on Signal Processing Systems

Publisher Rights Statement:
© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

General rights
Copyright for the publications made accessible via Heriot-Watt Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
Heriot-Watt University has made every reasonable effort to ensure that the content in Heriot-Watt Research Portal complies with UK legislation. If you believe that the public display of this file breaches copyright please contact open.access@hw.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Mixed Precision $\ell_1$ Solver for Compressive Depth Reconstruction: An ADMM Case Study

Yun Wu, Andrew Michael Wallace  
*The School of Engineering and Physical Sciences  
Heriot-Watt University  
Edinburgh, UK  
Y.Wu, A.M.Wallace@hw.ac.uk

Andreas Aßmann, Brian Stewart  
*Imaging Division  
STMicroelectronics R&D Ltd.  
Edinburgh, UK  
andreas.assmann, brian.stewart@st.com

Abstract—Rapid reconstruction of depth images from sparsely sampled data is important for many machine learning applications, including robot or vehicle assistance or autonomy, which require low power LiDAR sensing for eye safety, and resource reduction for FPGA or solid state implementation, especially with constrained energy budgets. A new compressive depth reconstruction design approach is proposed using a compact ADMM solver for the LASSO problem, which varies the precision scaling in an iterative optimization process. Implementations on an FPGA architecture show over 55% savings in hardware resources and 78% in power with only minor reduction in reconstructed depth image quality compared to single float precision.

Index Terms—Mixed Precision, Compressive Sensing, Depth Reconstruction, Alternating Direction Method of Multipliers, Field-Programmable Gate Array

I. INTRODUCTION

3D LiDAR is used widely for 3D imaging and scene reconstruction [1], which are key requirements for autonomous systems. Increasing the laser power may improve performance when viewed against background illumination, subject to eye safety requirements, but the large volumes of raw signal data demand considerable hardware resources for memory storage and digital signal processing. This limits application when resources are constrained, e.g. for mobile devices with limited battery supply and physical space, drone scene mapping with SLAM [2], augmented reality [3] and mobile robotics [4].

Parallel compressive sensing (CS) enables high, 3D imaging frame rates with reduced laser power, memory use and logic cost with lower power consumption [5]. In [6], further resource savings were made by reducing the precision of CS accelerators using the Alternating Direction Method of Multipliers (ADMM) algorithm. However, in that case the precision is fixed and pre-determined through the whole execution cycle. In contrast, mixed [7] and automated precision tuning [8] are promising approaches to improve energy efficiency when employing iterative refinement. However, current mixed and auto-tuned precision algorithms are either built for an instruction-driven architecture, or only support floating-point above single precision. This limits the use of approximation applied to smaller scale, resource constrained applications.

In this work, an mixed precision framework is proposed, supporting both fixed-point and floating point arithmetic, for 3D LiDAR CS depth reconstruction using a parallel $\ell_1$ solver of the least absolute shrinkage and selection operator (lasso) problem. Specifically, by investigating hybrid precision scaling on the iterative ADMM solver in compressive depth reconstruction, considerable savings in power, logic and memory resources are achieved in comparison with a fixed precision solution.

The contributions are summarized as follows:

- A compact ADMM algorithm using iterative mixed precision implementation
- An mixed precision framework for accelerator generation on an FPGA
- A comparative study between constant and mixed precision in terms of depth reconstruction quality, implementation cost, and power consumption.

CS depth reconstruction based on convex optimization problem and the use of precision scaling are introduced in Section II. In Section III, the proposed framework is illustrated. The results are demonstrated in Section IV with a conclusion in Section V.

II. BACKGROUND

A. Parallel Depth Reconstruction

![Fig. 1: Parallel Depth Reconstruction [6]](image)

Depth reconstruction with a parallel processing pattern using Checkerboard Compressive Sensing (CBCS) is illustrated in Fig. 1 [6]. The sensing pixels are sampled with a pre-defined pattern, the sensing matrix, which maps the scene to
This parallel reconstruction process can be re-formulated as an environment.

Exploiting sparsity in the scene, the intensity scaled distance per pixel \(x_Q \in \mathbb{R}^n\) and the intensity per pixel \(x_I \in \mathbb{R}^n\), are recovered from smaller independent measured feature vectors, \(y_Q, y_I \in \mathbb{R}^m\) with an associated sensing matrix, \(A \in \mathbb{R}^{m \times n}\). This parallel reconstruction process can be re-formulated as a twin optimization problem as

\[
\begin{align*}
\alpha \min_{x_Q} & \left\{ \frac{1}{2} \| y_Q - A \cdot x_Q \|_2^2 + \lambda \| x_Q \|_1 \right\}, \\
\alpha \min_{x_I} & \left\{ \frac{1}{2} \| y_I - A \cdot x_I \|_2^2 + \lambda \| x_I \|_1 \right\},
\end{align*}
\]

where \(\| \cdot \|_2\) is the Euclidean norm, \(\lambda > 0\) is a regularization parameter to \(\| \cdot \|_1 = \sum_i |i|\) being the pseudo \(\ell_1\)-norm. The formulation in Eq.(1), also known as the lasso [9], has been widely adopted in image reconstruction and denoising and can be solved using a convex optimizer such as ADMM [10].

B. Mixed Precision and Tuning

Mixed precision combines the accelerated and less resource-intensive processing of lower precision with the greater accuracy of higher precision arithmetic. It is normally deployed for iterative refinement using floating point data formats [11], [12], for example using three levels of precision [7]. Customized precision tuning instruments have been employed based on specific input sets [8]. Such precision scaling has gained increasing interest for more computation-intensive applications, such as deep learning, saving computation and data storage [13] using both floating point

\[
\text{sign} \times \text{mantissa} \times 2^{127}\text{-exponent} \tag{2}
\]

and fixed point [14]

\[
\text{sign} \times (2^{\text{integer}} + 2^{\text{fraction}}). \tag{3}
\]

By adjusting the combination of mixed precision arithmetic at compile-time, static precision tuning is performed once during the system design phase [7]. Such a tuning process can be considered as part of wider hardware/software co-design approaches, profiling the application data and code to compare with empirical measurement before running the system. On the other hand, dynamic precision tuning is invoked multiple times during the execution of application in system at run-time, in which case the overhead of precision reconfiguration must be considered [8].

Our work develops a framework for CS depth reconstruction using the parallel ADMM solver, enabling mixed precision design of both the arithmetic data type and the binary bit width appropriate to the performance requirements of the sensing environment.

III. The Design Framework

A. Compact ADMM with Mixed Precision

In the general case of lasso, the standard ADMM implementation requires \(O(n)\) flops with a soft-threshold based on \(\lambda/\rho\) where \(\lambda = \tau \max(|A^T y|)\) with scaling factors \(\tau\) and \(\rho\). Since the sensing matrix \(A\) is a fixed, user-determined, pattern, it is feasible to pre-compute partial iterative parameters for the dual variable update related to \(A\). Hence, all these static matrix parameters are identified and pre-computed to avoid unnecessary accuracy loss when using mixed precision during the iterations.

Algorithm 1 acts as the dual variable update function of each iteration for ADMM lasso [10] with pre-computed parameters \(\hat{\alpha}, \hat{\beta} = 1/\rho, G = A^T y, H = A^T U^{-1} L^{-1} A/\rho^2\). This solves the compressive sensing problem addressed in Eq. 1. The LU decomposition of \(A\) is also pre-computed, \(\{L, U\} \in \mathbb{R}^{m \times m}\), and the inversion is transformed into separate inversions by the pre-computed \(L^{-1}\) and \(U^{-1}\). \(l_k\) is the function index pointing to the selected precision. As shown, the inputs are the pre-computed matrices \(G\) and \(H\) while the outputs are the dual iterative variables, \(z\) and \(u\).

Algorithm 1 \((\hat{z}, \hat{u}) = \text{mixiter}(G, H, z, u)\)

Input: \(G, z, u\)
Output: \(z, u\)

Initialization: const\(\{\alpha, \hat{\alpha}, \hat{\beta}, \rho, \kappa = \lambda/\rho\}\),
1: \(q = G + \rho \cdot (z - u)\)
2: \(x = \alpha \cdot (\hat{\beta} \cdot q - H \cdot q) + \hat{\alpha} \cdot z\)
3: \(z = \max\{0, (x + u) - \kappa\} - \max\{0, -(x + u) - \kappa\}\)
4: \(u = u + x - z\)
5: return \(z, u\)

In a real scenario, the LiDAR sensor derives the histogram of photon counts and transforms it into digital signals without any referenced feedback. The known characteristics of the LiDAR sensing parameters is utilized to tune the mixed precision schedule. As shown in Algorithm 2, assuming buffered measurements for each snapshot of a scene, \(G, H\) and other pre-defined parameters \(\alpha, \hat{\alpha}, \hat{\beta}, \rho, \hat{\beta}\) can be pre-computed. This establishes a schedule for arithmetic precision.

The quantising effect from single (32 bits) to reduced precision is considered as the criterion for precision scaling. Specifically, the normalized absolute difference is used for floating point (FP) arithmetic while the absolute difference is adopted for fixed point (F32) arithmetic due to the different functional relationships between the binary representations and decimal values. The most representative values, the maximum and minimum of those pre-computed parameters, are picked for the numerical analysis.

In Eq. 4, \(\epsilon\) is the normalized absolute difference used to judge the quantization effect of FP,

\[
\epsilon = \frac{\|v - \text{castc}(v)\|_1}{\|v\|_1} \leq \omega \tag{4}
\]

while in Eq. 5 \(\epsilon\) is the absolute difference used to judge the quantization effect of F32,

\[
\epsilon = \|v - \text{castc}(v)\|_1 \leq \omega \tag{5}
\]

where \(\omega\) is the accuracy criterion, \(\omega = 5 \times 10^{-3}\) in this work.

Compact ADMM is illustrated in Algorithm 2 using the iterative function \(\text{mixiter}\) of Algorithm 1 with mixed precision. Fixed iteration number, \(k_{\text{max}} = 5\), is adopted in our case as no further improvement on reconstruction depth information.
Therefore, increasing precision is selected by demanding finer accuracy of the iterative dual variable update. The least possible number of bits for arithmetic precision is[index_6]{\textit{l}_k}\text{ is increased by 1 to meet the demanding finer accuracy of the iterative dual variable update. Therefore, increasing precision is selected by }\text{*mixiter}{\textit{[l}_k]};\text{ this changes the elementary bit width gradually, pointing to its corresponding function as described in Algorithm 1. \textit{cast}_c\text{ represents the general function that transforms the input to the current arithmetic precision. Note that }\text{cast}_c\text{ only demonstrates the arithmetic precision transformation at the software level, and is not an overhead in hardware design at the appropriate precision. In this work, only mixed precision within the same arithmetic type is considered.}

**B. Mixed Precision Accelerator Generation**

![Fig. 2: Mixed Precision Design Flow](image)

Our framework, illustrated in Fig. 2, has been developed by using Matlab tools to prototype quickly an approximate accelerator with mixed-precision arithmetic on reconfigurable platforms. To support mixed precision design, an approximate library is created to support arbitrary arithmetic precision linear algebra based firstly on a customized floating-point library FloatX [11] and secondly the Xilinx fixed-point library [15]. The library is established with C++ generic programming that allows later specific arithmetic type allocation. The major features are listed as follows:

- General (non-symmetric) real value operations,
- Arbitrary bit width floating- and fixed-point arithmetic,
- Basic vector/matrix algebra arithmetic (addition, subtraction, multiplication, division, inversion),
- Triangular factorization (LU, and Cholesky decomposition),
- Orthogonal factorization (QR decomposition).

Based on the approximate library, a user-defined kernel, which indicates the specific portion of signal processing application to be the target for approximation, is considered as input in Fig. 2. By compiling the input source, a list of various iterative functions are created using different arithmetic precision, as expressed in Algorithm 1. The source compiler is developed based on the Matlab MEX compiler API where the Design Space Exploration (DSE) with precision adaptation is evaluated based on the input data. The kernel with chosen precision is adopted for the generating accelerator with a Hardware Description Language (HDL) by calling Xilinx High-Level Synthesis (HLS) tools within the Matlab Framework.

**IV. Evaluation**

The mixed precision ADMM algorithm using both floating- (FP) and fixed-point (FXP) arithmetic on two synthetic scenarios and a real underwater sensing data is illustrated and evaluated for compressive depth reconstruction. The reconstructed depth image using the dSparse [5] (a non-compressive approach with oversampling) with single floating point (FP 32 bits) is adopted as the ground truth for accuracy evaluation of reconstruction, while single floating point ADMM solution is considered as the baseline for comparisons of clock speed, resource usage and power consumed.

**A. Performance**

By using the design flow for different scenes at different stand-off distances and dynamic ranges, mixed FP from 18 to 22 bits, and FXP from 20 to 24 bits are employed for the short range synthetic scene with distances ranging from 1.8 to 3.4 meters. Mixed FP from 16 to 20 bits, and FXP from 30 to 34 bits are adopted for the long range, synthetic scene with distances ranging from 0 to 100 meters.

The Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [16] are both considered as quality metrics, showing the fidelity of reconstruction and the perceived quality based on the pixel-wise absolute difference for normalised similarity scaling.

Fig. 3 illustrates the comparison of depth reconstruction for a synthetic short range scenario. For floating point arithmetic, using constant, 18 bit FP for all the iterations in Fig. 3c is the worst, while using constant, 22 bit FP shows the best
18 bit FP, with 1 bit increment each iteration, there is a loss of only 0.02 dB and 1% in PSNR and SSIM respectively compared to the best. A similar phenomenon happens for fixed point arithmetic in Fig. 3f-3h.

Fig. 5 illustrates the detailed PSNR and SSIM metric values, comparing the mixed precision implementations to constant precision. This shows comparable performance when applying a mixed precision schedule (dashed blue line) to the ADMM algorithm to using a largest, constant bit width in the fixed schedule (solid red line).

Further illustrations in Figures 6, and 7 present the outcomes for both synthetic long range and real underwater scenarios, using either mixed or constant floating point precision. As shown, the recovered depth images are of similar perceptual quality. Having established that there is negligible degradation in signal quality, as measured by the common PSNR and SSIM metrics, the advantages in cost and power efficiency is demonstrated in the next section.

B. Resource use and power consumption

Fully pipelined architectures using single and mixed precision with FP and FXP arithmetic were created on a Xilinx Ultrascale+ ZCU106 architecture using Vivado 2019.1. The pipelined architecture is implemented for the mixed precision ADMM to solve the twin lasso optimization problem in Eq. 1, where each iteration corresponds to a pipeline stage with allocated precision.

The resource utilisation is derived from the Xilinx tools using the place and route implementation, while the power is estimated using the Xilinx Power Estimator (XPE). Tables I
and II present the resource and power comparisons when using constant and mixed precision for all scenarios.

**TABLE I: Constant Precision Resource Utilization**

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>LUT (×10^3)</th>
<th>FF (×10^3)</th>
<th>DSP48E1</th>
<th>Clock (MHz)</th>
<th>Dynamic Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13.03</td>
<td>21.02</td>
<td>105</td>
<td>427</td>
<td>0.591</td>
</tr>
<tr>
<td></td>
<td>11.32</td>
<td>16.87</td>
<td>30</td>
<td>467</td>
<td>0.291</td>
</tr>
<tr>
<td></td>
<td>13.47</td>
<td>17.85</td>
<td>30</td>
<td>479</td>
<td>0.332</td>
</tr>
<tr>
<td></td>
<td>7.78</td>
<td>5.07</td>
<td>30</td>
<td>421</td>
<td>0.142</td>
</tr>
<tr>
<td></td>
<td>8.43</td>
<td>6.27</td>
<td>60</td>
<td>448</td>
<td>0.217</td>
</tr>
<tr>
<td></td>
<td>9.41</td>
<td>6.88</td>
<td>60</td>
<td>467</td>
<td>0.243</td>
</tr>
</tbody>
</table>

**TABLE II: Mixed Precision Resource Utilization**

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>FP 16-20</th>
<th>FP 18-22</th>
<th>FP 20-24</th>
<th>FP 26-30</th>
<th>FP 30-34</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT (×10^3)</td>
<td>10.56</td>
<td>14.16</td>
<td>30</td>
<td>450</td>
<td>0.254</td>
</tr>
<tr>
<td>FF (×10^3)</td>
<td>11.71</td>
<td>16.64</td>
<td>30</td>
<td>464</td>
<td>0.292</td>
</tr>
<tr>
<td>DSP48E1</td>
<td>7.24</td>
<td>4.92</td>
<td>21</td>
<td>419</td>
<td>0.130</td>
</tr>
<tr>
<td>Clock (MHz)</td>
<td>7.86</td>
<td>5.87</td>
<td>42</td>
<td>403</td>
<td>0.166</td>
</tr>
<tr>
<td>Dynamic Power (W)</td>
<td>8.91</td>
<td>6.55</td>
<td>57</td>
<td>435</td>
<td>0.214</td>
</tr>
</tbody>
</table>

The mixed precision values correspond to the bit width variation chosen according to the depth scaling for each of the scenes while the constant precision cases are those of the highest bit width in the mixed precision combination. As shown, by adapting mixed precision with either FP or FXP, there are significant hardware resource and power saving when compared to single float precision.

The depth reconstruction performance is quantified against the consumed resource and power in Fig. 8a and 8b. Overall, with less than 1% degradation in the SSIM metrics for depth reconstruction shown in Fig. 8, the cost using mixed precision is reduced by up to 55% for Look-Up-Table (LUT) and Flip-Flop (FF) registers, up to 70% for DSP units and Block RAM (BRAM) usage when compared to single precision. FP mixed precision has similar resource utilization to constant precision, while the cost with FXP mixed precision is slightly lower than the comparative constant precision with reductions of up to 10% LUT in FF.

The estimated dynamic power consumption using mixed precision is also significantly reduced by over 78% compared to single precision and by approximately 10% compared to the corresponding constant precision. These savings are for single block processing; savings will be greater when parallel blocks are introduced.

V. CONCLUSIONS

A mixed precision framework with a pre-computed schedule has been illustrated for parallel compressive depth reconstruction using convex optimisation and linear algebra libraries. By adapting the mixed precision schedule to the known stand-off and dynamic range of the LiDAR-sensed scene, incremental precision scaling over iterations has applied to the ADMM solver of depth image reconstruction optimization problem. The results show that mixed precision iteration enables similar performance of depth reconstruction compared to single float precision with significant cost and power reductions, greater than 70% in the best cases.

VI. ACKNOWLEDGEMENT

This work is supported by the Engineering and Physical Sciences Research Council of the UK (EPSRC) Grant number...
EP/S000631/1 and the UK MOD University Defence Research Collaboration (UDRC) in Signal Processing

REFERENCES


