LANMC: LSTM-Assisted Non-Rigid Motion Correction on FPGA for Calcium Image Stabilization

Zhe Chen
University of California, Los Angeles
Los Angeles, California
zhechen@ucla.edu

Hugh T. Blair
University of California, Los Angeles
Los Angeles, California
tadblair@ucla.edu

Jason Cong
University of California, Los Angeles
Los Angeles, California
cong@cs.ucla.edu

ABSTRACT

Calcium imaging is an emerging technique for visualizing and recording neural population activity at large scale in *vivo*. Non-rigid motion correction is a critical step in the calcium image analysis pipeline due to non-uniform deformations of the brain tissue during the data collection. Existing non-rigid motion correction algorithms are costly in computation time and energy, and it is hard to implement such algorithm in real time on an embedded device. In this paper, we propose LANMC, an LSTM-assisted non-rigid motion correction method for real-time calcium image stabilization. This method reduces the computational cost by using the LSTM inference to predict the non-rigid motion. Based on this method, we demonstrate a non-rigid motion correction implementation for real-time calcium image stabilization on FPGA. Experimental results show that the non-rigid motion correction can be accomplished within 80 μs on the Ultra96 under 300 MHz frequency, and the latency outperforms that on a 12-thread CPU by 82x.

KEYWORDS

Calcium image, long short-term memory (LSTM), motion correction

ACM Reference Format:
Zhe Chen, Hugh T. Blair, and Jason Cong. 2019. LANMC: LSTM-Assisted Non-Rigid Motion Correction on FPGA for Calcium Image Stabilization. In *The 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '19), February 24–26, 2019, Seaside, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3289602.3293919

1 INTRODUCTION

Calcium imaging is a neural recording technique that can monitor neural population activity in *vivo* [15]. Recent progress in miniaturized fluorescent calcium imaging has enabled this technique to be realized in a light-weight head mounted miniscope sensor device for freely moving mice and rats [1, 5]. Such device can generate terabytes of data over days of recording, creating a huge computational burden for calcium imaging analysis [6].

Motion correction is a critical step in the calcium image analysis. It removes motion artifacts caused by inevitable displacement of the brain inside the skull during the image capturing. Due to the non-uniform deformations of the brain tissue, the motion artifacts can be non-rigid, which increases the difficulty for the motion correction. An effective non-rigid motion correction algorithm has been proposed in [13], but it is costly in computation and not efficient for the real-time implementation. Real-time rigid motion correction for calcium imaging has been realized on CPU [6], but it does not account for the brain tissue deformation.

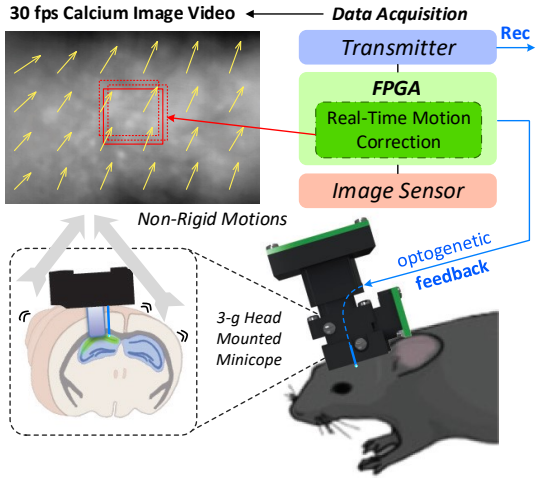

Figure 1: Real-time non-rigid motion correction for calcium image sensed by head-mounted miniscope [1].

A real-time non-rigid motion correction for calcium image stabilization is in demand. Fig. 1 illustrates a flow-diagram for a closed-loop neurofeedback application in which the real-time non-rigid motion correction plays an important role in supporting the optogenetic feedback stimulation. Compared with state-of-the-art embedded CPUs and GPUs, FPGA is more appropriate for this application, because its customizable processing architecture can provide higher energy efficiency, shorter processing latency, and more flexible interface in processing data from the sensor. As the temporal and spatial resolution of the miniscope image sensor keeps increasing, it remains challenging to implement highly efficient non-rigid motion correction on FPGA due to the algorithm complexity, especially considering the limited hardware resource and energy budget on a head-mounted device.

In this paper, we propose a long short-term memory (LSTM) assisted non-rigid motion correction (LANMC) algorithm and implement it on FPGA for real-time calcium image stabilization. First, we introduce the method in saving computation and evaluate its performance by making comparison with existing non-rigid motion correction algorithm. Then we introduce our FPGA design which is orthogonal to the proposed method for a highly efficient

FPGA '19, February 24–26, 2019, Seaside, CA, USA

ACM ISBN 978-1-4503-6137-8/19/02…$15.00
https://doi.org/10.1145/3289602.3293919

implementation. Finally, we report experiment results and compare processing speed and energy efficiency against the CPU which is commonly used in existing miniscope data acquisition platform.

The contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to propose using LSTM for the non-rigid motion correction and realize real-time non-rigid motion correction for calcium image stabilization in real time. Our method can reduce the computation cost by 95% while maintaining acceptable accuracy compared to a state-of-the-art offline algorithm.
- We propose a folding architecture for real-time contrast filtering by leveraging the central symmetry of a 17×17 filter kernel. The proposed architecture saves over 70% of operations, and achieves 25 cycle processing latency and 169 op/cycle computation density during the runtime.
- We implement non-rigid motion correction for calcium image stabilization on the Ultra96 with 4.5 W power consumption. Under 300 MHz, the processing latency achieves 80 μs, which outperforms the evaluation result on multi-core Xeon E5-2860 CPU by 82x. Combined with using the LSTM inference, our implementation has close to 4 orders of energy efficiency gain compared to the conventional offline non-rigid motion correction on CPU.

2 BACKGROUND

2.1 Conventional Non-Rigid Motion Correction

Motion correction is a critical processing step in a variety of calcium image analysis algorithms [6, 12, 13]. Recent work shows that piecewise rigid motion correction can effectively reduce non-rigid motion artifacts and outperform other methods for calcium image stabilization [13]. Fig. 2 illustrates the processing flow of this method[1]. For calcium image analyzed here, the first step is to enhance the image with a contrast filter that removes the bulk of the background [13]. The kernel size of the filter is determined by the diameter of the cell bodies in the source image, and 17×17 was the minimum kernel size for generating sufficiently accurate motion correction results. The second step is to divide the field of view of the image into overlapping patches $\boldsymbol{f}(i,j) \in \mathbb{R}^{N_P \times N_P}$, $i \in [1, \lceil W/N_P \rceil]$, $j \in [1, \lceil L/N_P \rceil]$, to perform the piecewise rigid motion correction. For each patch, the algorithm calculates its cross correlation $\boldsymbol{CC}(i,j) \in \mathbb{R}^{N_P \times N_P}$ against a template $\boldsymbol{g}(i,j) \in \mathbb{R}^{N_P \times N_P}$ from the frequency domain through 2D FFT/IFFT operations. Finally, the motion vector for each image patch is extracted by finding the position of the maximum amplitude of the cross correlation, and the subpixel resolution is achieved through the interpolation [6].

2.2 Algorithm Complexity

For the algorithm described in Section 2.1, suppose the kernel size of the filter is $N_K \times N_K$, and the image patch size is $N_P \times N_P$. The operation count per image frame can be estimated by

$$C_{NoR} = \left(2N_K^2 - 1\right) WL + \left(8N_P^2 \log_2 N_P + 2N_P^2\right) \left\lceil \frac{W}{N_P} \right\rceil \left\lceil \frac{L}{N_P} \right\rceil \quad (1)$$

[1]Histogram equalization is adopted on the filtered image for better visibility.

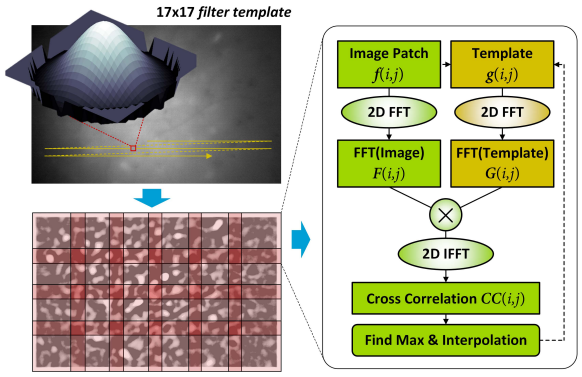

Figure 2: Conventional non-rigid motion detection flow.

in which W and L represent the width and length of the calcium image. For simplicity, we assume N_P is a power of two, and ignore the cost of the interpolation. The first addend is derived from the contrast filter with complexity $O(N_K^2)$ per pixel, while the second addend is contributed by the motion vector extraction with complexity $O(N_P^2 log_2(N_P))$ per patch. For a typical parameter set with N_K=17 and N_P=128, the contrast filter dominates the operation count. The goal of this paper is to reduce the complexity described in Eq. 1 and realize an efficient non-rigid motion correction with short latency for real-time calcium image analysis.

2.3 LSTM Inference

LSTM is a type of recurrent neural networks that has been successfully used in a variety of time-series prediction tasks [9]. For an LSTM model with one layer and N_H hidden nodes, the inference output for the current time step is based on the updates of four independent types of gates: input gate (I), forget gate (F), cell gate (G) and output gate (O) by taking advantage of the recurrent connections from the hidden nodes. Recent work shows that a compact LSTM model with a typical setting of N_H=5 can achieve sufficient accuracy in approximating IIR filter functions [3]. The inference complexity of N_H-node LSTM derived by

$$C_{LSTM}(N_H) = 16N_H^2 + 6N_H \quad (2)$$

shows its good potential for high efficiency on-line inferencing.

3 PROPOSED METHOD

3.1 LSTM-based Non-Rigid Motion Correction

We propose a non-rigid motion correction algorithm for miniscope captured calcium image based on LSTM inference to reduce the computation cost. As Fig. 3(a) shows, instead of performing heavy motion calculations for each divided image patch $\boldsymbol{f}(i,j)$, we evaluate the motion only at a central $N_C \times N_C$ pixel region, and then we use the calculation result to predict non-rigid motions of all image patches based on LSTM inference.

Fig. 3(b) shows the motion extraction from the central region and all image patches throughout the calcium image video session. The displacement of each patch $\boldsymbol{f}(i,j)$ is represented by a motion vector containing two values $\{x_{i,j}, y_{i,j}\}$. These values represent the rigid motion against the template along the horizontal and vertical axes with sub-pixel precision. For motion extraction at the central

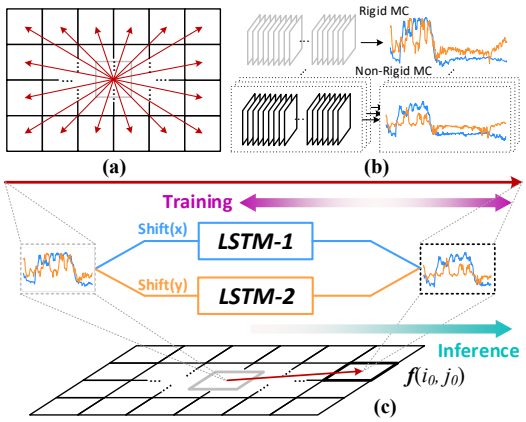

Figure 3: (a) Proposed LSTM-assisted non-rigid motion correction. (b) Motion vector extraction for the central region and all image patches. (c) LSTM inference on one patch.

region, a light-weight rigid motion correction described in Fig. 2 is performed to reduce the computation cost. For motion extraction at all image patches, a non-rigid motion correction algorithm NoRMCorre [13] is used to achieve high accuracy.

Fig. 3(c) illustrates the LSTM-assisted method in detail. Since all image patches are independent, we only show one projection from the central region to an image patch $\boldsymbol{f}(i_0, j_0)$ for simplicity. The operation of the LSTM can be divided into two stages: offline training and online inference. During the training, the motion vector time series $\{x_C, y_C\}$ extracted at the central region is used as input, and the motion vector time series extracted at the patch $\boldsymbol{f}(i_0, j_0)$ are used as the target. A pair of compact LSTM networks are trained to adapt the motion vector components of the central region to those of the patch $\boldsymbol{f}(i_0, j_0)$ along the horizontal and vertical axes, respectively. After the LSTM networks are well trained, they are deployed to inference the offline motion correction at much shorter latency and much lower computational cost.

For the rigid motion correction at the central region, the operation count per frame can be derived by:

$$C_{Rigid} = \left(2N_K^2 - 1\right) N_C^2 + \left(8N_P^2 \log_2 N_P + 2N_P^2\right) \tag{3}$$

and the computation cost saving can be estimated by:

$$G_{eff} = C_{NoR} / \left(C_{Rigid} + C_{LSTM}\left(N_H\right) \left\lceil \frac{W}{N_P} \right\rceil \left\lceil \frac{L}{N_P} \right\rceil\right) \tag{4}$$

Considering the miniscope [1] resolution 752×480 and parameters N_C = 128 and N_H = 5, the G_{eff} results in 22.2x by keeping the default settings in Section 2.2. It indicates that 95% of operations can be saved by taking advantage of the LSTM inference for the non-rigid motion correction.

3.2 Algorithm Evaluation

We carried out an evaluation based on 26 sessions of calcium image videos, lasting for 50 seconds each. We used 25 sessions as the training set, and the remaining session as the test set. During the offline training, we derived the input based on the rigid motion correction shown in Fig. 2, and extracted non-rigid motion vectors

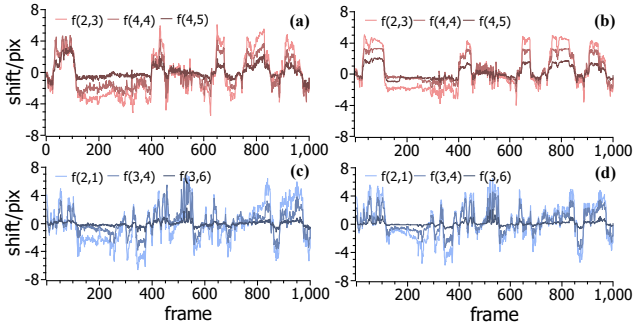

Figure 4: Extracted motion vectors in horizontal/vertical direction from three different image patches by using the NoRMCorre method (a)/(c) and the LSTM inference (b)/(d).

using the NoRMCorre [13] as the training targets. For each image patch $\boldsymbol{f}(i, j)$, we trained a pair of compact 5-node LSTM networks using Caffe [10] to predict motion vectors in horizontal and vertical directions. We adopted the step learning rule, and set both the base learning rate and the gamma to be 0.1. During the inference, we fed the rigid motion vector of the central region extracted from the test video to the well-trained LSTMs, and the outputs were used as an approximation of non-rigid motion vectors for each image patch $\boldsymbol{f}(i, j)$. Fig. 4 shows a comparison on non-rigid motion correction carried out by conventional method and the LSTM inference. Fig. 4(a) and (c) show vectors extracted by conventional methods from three selected image patches in horizontal and vertical directions, respectively, whereas Fig. 4(b) and (d) shows motion vectors inferenced by the LSTMs correspondingly.

We used the residual of optical flow (ROF) measurement [13] to evaluate the accuracy of the proposed method. Fig. 5 (a) and (b) show the evaluation results on No.451-453 and No.554-556 frames with obvious non-rigid motion artifacts. We first extracted the ROF of the registered frames based on optical flow [4], and then calculated the averaged ROF map for the vector plot and evaluated the ROF per pixel within the map. As the results show, the non-rigid motion correction algorithm NoRMCorre can get rid of most of the non-uniform motion artifacts, whereas the LANMC corrects significant amount of non-rigid motions by approximating the NoRMCorre. By using the LSTM inference, the ROF can be reduced by 69.2% and 50% for the selected frame periods, respectively.

Table 1 summarizes the evaluated ROF for non-rigid, rigid and LSTM-assisted methods on the test session. Evaluation results show that the LANMC on average reduces 48% ROF compared with the rigid method, and the accuracy is comparable with the NoRMCorre. We also compared the horizontal component mean absolute difference (MAD) $diff(x)$, vertical component MAD $diff(y)$ and amplitude MAD $diff(a)$ among these methods. Comparison results in Table 1 show that the LSTM-assisted method achieves higher accuracy than the rigid method in approximating the NoRMCorre method.

4 FPGA DESIGN

The FPGA design of the LANMC consists of a contrast filter accelerator, an FFT-based cross correlation module for motion vector extraction at the central region, and an LSTM inference kernel for non-rigid motion prediction at all distributed image patches.

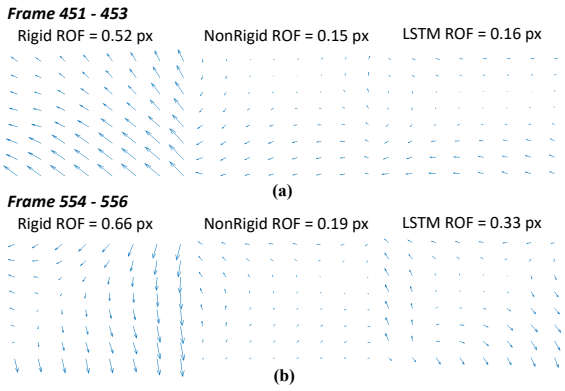

Figure 5: Mean ROF of the motion corrected (a) No.451-453 frames and (b) No.554-556 frames across time by the rigid, non-rigid and LSTM-assisted methods.

Table 1: Performance of the LSTM method compared with conventional motion correction algorithms

	ROF(pixel)	$diff(x)$	$diff(y)$	$diff(a)$
NoRMCorre [13]	0.19±0.11	0	0	0
Rigid Method	0.60±0.45	0.879	0.658	1.248
LANMC	0.31±0.22	0.621	0.362	0.804

4.1 Contrast Filter Design

A key observation at the contrast filter described in Section 2.1 is that the template of the filter is symmetric about the horizontal, vertical and diagonal axes. This provides the opportunity to reduce the computation efforts. Instead of performing multiplication for each coefficient in the filter template, we can first add up the input values corresponding to the coefficients in symmetry, and then multiply the sum with the coefficient to get the equivalent result. By leveraging this reordering, logic and memory resource costs in realizing the contrast filter can be reduced. Fig. 6 shows the proposed folding architecture for the contrast filter. The architecture features three stages of folding data flow. On the first stage, instead of feeding the N_K 8-bit pixels from the 1D column buffer directly to a $N_K \times N_K$ processing element (PE) array [14], we inserted an adder stage between the column buffer and a shift register array, as Fig. 6(a) shows. The adder stage reduced the amount of data and the corresponding array size by half with negligible latency overhead. In a similar manner, Fig. 6(b) shows the second-stage folding architecture leveraging the vertical symmetry. For each row of PE, we employed a row of 9-bit adders to fold the horizontal partial sums. Finally, in the quarter-size array, we used a last stage of 10-bit adders to sum the values having a diagonal line of symmetry as shown in Fig. 6(c). With three stages of folding, the requirement on the multiplication count is reduced to one eighth, and we in further skipped multiplication with zeros to save computation. Table 2 shows a comparison on hardware cost between the implementations with and without folding. By taking advantage of the folding, the proposed architecture saves >80% logic and registers and >60% DSPs. Evaluation on multicore CPU with OpenMP shows that the

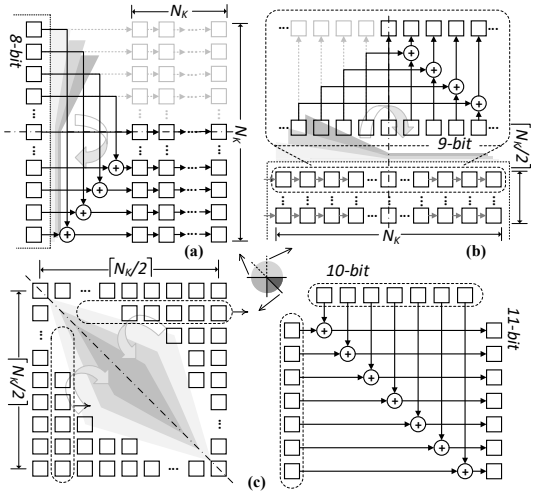

Figure 6: Proposed (a) 1st stage (b) 2nd stage and (c) 3rd stage folding architecture for the contrast filter featuring centrally symmetric coefficients.

Table 2: FPGA resource saving by the folding architecture

	W/ Folding	W/O Folding	Resource Saving
LUT	1211	6373	81.0%
FF	1972	11809	83.3%
DSP	22	56	60.7%
SRL	15	489	96.9%

Table 3: Comparison on runtime of the full frame filtering with multi-core CPU

	This work		4 thr	8 thr	12 thr	16 thr
Freq(MHz)	100	300	1200 - 1500			
Runtime(ms)	3.73	1.25	135	90	53	62

folding speeds up the contrast filtering by 1.8x. Table 3 shows the runtime comparison on contrast filtering with folding between the FPGA and the Xeon E5 2860 CPU. The FPGA design achieves 25 clock cycle latency, and the runtime under 300 MHz outperforms the evaluation on the CPU by over 40x speedup.

4.2 Reused Parallel FFT/IFFT

As Section 2.1 introduces, an efficient way to calculate the cross correlation is to perform dot product at the frequency domain. We implemented the 2D $N_P \times N_P$ FFT/IFFT transforms by carrying out consecutive NP horizontal and N_P vertical N_P-point FFT/IFFTs. Computation unit and BRAM buffer resource for each round of N_P-point transforms are reused. In order to achieve short processing latency, we unrolled the FFT/IFFT transformer by 4. In addition, we partitioned the BRAM into 4 blocks to cope with the computation throughput. Fig. 7 shows the proposed timing for the BRAM access and the transfer of the data to the FFT/IFFT units. We stored both the input and the result data in a row-major order. For horizontal

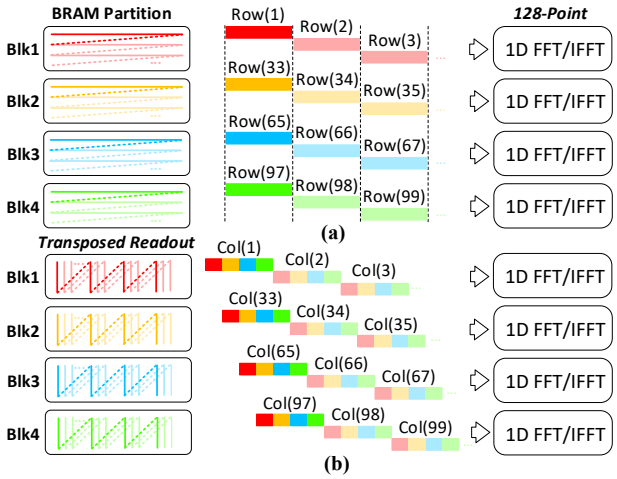

Figure 7: (a) BRAM partition and FFT/IFFT unrolling for row-wise parallel operation, and (b) transposed BRAM readout timing for column-wise parallel FFT/IFFT operations.

Table 4: Comparison on latency and FPGA resource usage

	Freq	Latency	LUT	FF	DSP
This work	300MHz	45.8 μs	8913	12624	60
	100MHz	137.4 μs	8886	12539	60
Ref. [2]	100MHz	328.2 μs	14036	14622	108

FFT/IFFT transforms, the data can be fetched independently from the BRAMs, and the FFT/IFFT units operate in parallel, as Fig. 7(a) shows. For vertical FFT/IFFT transforms, the data access from the BRAM is organized in pipeline avoiding confliction, as Fig. 7(b) shows. The initial latency between two rows is NP/4 clock cycles, and the FFT/IFFT units are in full operation. Table 4 shows the comparison on processing latency and hardware cost of 2D FFT/IFFT for motion detection. Our proposed reusable parallel FFT/IFFT implementation outperforms [2] by 2.38x in processing latency under the same 100 MHz clock frequency, and our design reduces 25% logic resource and >40% DSP.

4.3 LSTM Inference Kernel

Fig. 8 shows the proposed microarchitecture for the LSTM inference kernel [3]. The LSTM inference for all distributed image patches can be fully unrolled. However, it will cause a linear increase on hardware cost as the number of image patches increases. In order to save the computation resource and balance the processing time for each stage of the non-rigid motion correction algorithm, we shared the LSTM kernel in predicting motion vector components in each patch, and reused the same kernel for all patches. We stored weights and states for all LSTM instantiations in an affordable on-chip buffer. Inside each LSTM kernel, the arithmetic units are unrolled by a factor of 4 corresponding to the number of gate types in the LSTM model. Within each unrolled unit, the matrix vector multiplication between LSTM weights and states can be pipelined with initial interval (II) equal to 1.

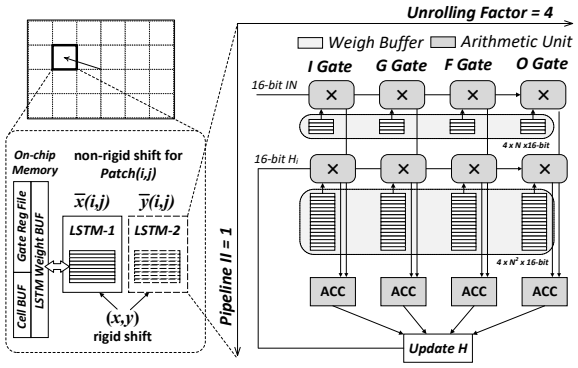

Figure 8: Microarchitecture for the LSTM inference [3].

Figure 9: Power measurement of the non-rigid motion correction implementation on the Ultra96 board.

Table 5: FPGA resource utilization on the Ultra96

	LUT	FF	BRAM	DSP
Contrast Filter	1211	1972	0	22
Cross Correlation	8901	12624	34	60
Interpolation	3212	5175	0	36
LSTM	2129	1896	2	28
Overall	28338	37446	139	146
Utilization	40.16%	26.53%	64.35%	40.56%

5 EXPERIMENT RESULT

We carried out the non-rigid motion correction implementation on both the ZC706 and the Ultra96 boards. We first verified the proposed LSTM-assisted non-rigid motion correction on the ZC706 by testing it with an external miniscope image sensor connected through the FMC interface. Then we ported the same design onto the Ultra96, and test it with emulated data stream generated inside the SoC device. The operating frequency on the Ultra96 achieves 300 MHz, and the processing latency for each frame is 79.65 μs, which leaves a large margin for consecutive calcium image analysis algorithm considering the 33-ms frame interval. The power consumption of this implementation is 4.5 W, as shown in Fig. 9, and the breakdown of the hardware usage is shown in Table 5. The interpolation is realized in single precision floating point, while the rest kernels are in 16-bit fixed-point. The adopted bit quantization does not degrade the accuracy achieved in simulation in Section 3.2. Besides the listed kernels, our design also included a frame buffer, DMA controller, AXI interface and peripherals, which contribute to the overall hardware cost.

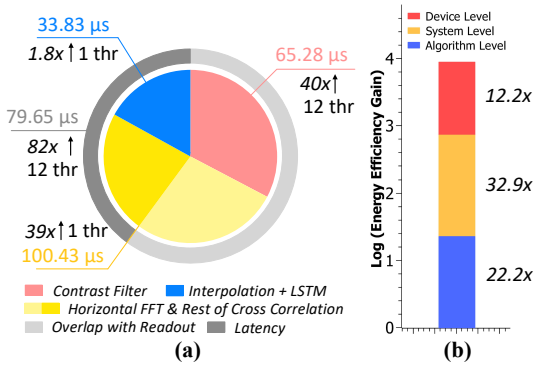

Figure 10: (a) Breakdown of runtime on FPGA and (b) energy efficiency gain over the CPU implementation.

We compared our implementation with the evaluation on the Xeon E5-2680 CPU with 12 threads. The runtime comparison results are shown in Fig. 10(a). The comparison result shows that the FPGA can achieve 82x speedup in processing latency over CPU. The main reason is that the FPGA design not only speeds up the most time consuming contrast filter by 40x, but also hides 60% runtime behind the read-out timing of the image sensor. For the cross correlation and the LSTM inference, we only evaluated the CPU performance with single thread because the data dependency limits the performance on multi threads, and the runtime is trivial compared to the filter stage. Compared with the single-thread CPU implementation, the FPGA achieves 39x speedup for the cross correlation with the 100.43 μs runtime, and 1.8x speedup for the LSTM inference stage with the 33.83 μs runtime. We can potentially achieve higher speedup for the LSTM inference by duplicating the LSTM kernels, but benefit will be very limited considering additional hardware resource cost given that the LSTM inference is not the bottleneck in performance. Fig. 10(b) summarizes the energy efficiency gain over the high performance multi-core CPU achieved by our work. First, our proposed LSTM-based method contributes 22.2x energy efficiency gain by reducing the computational complexity from the algorithm level. Secondly, our FPGA system design provides 32.9x gain by realizing consistent speedup for each kernel step. Finally, the adoption of the state-of-the-art FPGA device adds another 12.2x gain, in which we suppose the power consumption of the CPU using 12 threads is 51.4 W based on the thermal design power specification, and the power consumption of the FPGA is 4.5 W. In all, we get close to 4 orders of energy efficiency gain over a conventional non-rigid motion correction implementation on CPU.

6 RELATED WORK

Motion correction for calcium image has been sufficiently discussed in previous literatures [6, 12, 13]. A non-rigid motion correction method proposed in [13] outperforms other methods in accuracy, but it costs a long processing time due to the high computation complexity. [12] proposed another motion correction methods for calcium image, but it is only suitable for offline analysis. [6] realized a real time rigid motion correction on CPU for calcium image analysis, but it operates offline and the runtime is >60x longer than this work. [7] proposed FPGA acceleration for motion estimation based on block matching, and [2] realized motion blur removal on FPGA based on 2D FFT. [11] implemented a real-time video stabilization on FPGA based on feature point matching. Since these methods are not customized for calcium image non-rigid motion correction, the accuracy performance limit their use for calcium image analysis. Finally, beyond conventional image processing and customized computing techniques, neural network training has also been recognized as effective method to improve computing efficiency in recent research [8].

7 CONCLUSION

In this paper, we proposed a non-rigid motion correction algorithm for calcium image stabilization by taking advantage of the LSTM inference. It largely reduces the computation complexity and remains high accuracy. We introduced the FPGA design for the LSTM-assisted non-rigid motion correction method. Our design can achieve short latency for real-time calcium image non-rigid motion correction and high energy efficiency, and it has the potential to be built into existing miniaturized head mounted miniscope device.

ACKNOWLEDGMENTS

This work is partially supported by the NSF under Grant No.: CCF-1436827 and No.:DBI-1707408. The authors would like to thank Dr. Daniel Aharoni for his help on the miniscope device.

REFERENCES

[1] Denise J. Cai, Daniel Aharoni, Tristan Shuman, and et al. 2016. A shared neural ensemble links distinct contextual memories encoded close in time. *Nature* 534 (2016), 115–118.
[2] T.N. Chandrapala, L.M.A.P. Cabral, S. Ahangama, and et al. 2012. Hardware implementation of motion blur removal. In *Int. Conf. Field Program. Log. Appl.* 243–248.
[3] Zhe Chen, Andrew Howe, Hugh T. Blair, and et al. 2018. CLINK: Compact LSTM inference kernel for energy efficient neurofeedback devices. In *Proc. Int. Symp. Low Power Electron. Des.* 2:1–2:6.
[4] Gunnar Farnebäck. 2003. Two-frame motion estimation based on polynomial expansion. In *Image Anal.* Springer Berlin Heidelberg, 363–370.
[5] Kunal K Ghosh, Laurie D Burns, Eric D Cocker, and et al. 2011. Miniaturized integration of a fluorescence microscope. *Nat. Methods* 8 (2011), 871.
[6] Andrea Giovannucci, Johannes Friedrich, Matthew Kaufman, and et al. 2017. OnACID: Online analysis of calcium imaging data in real time. In *Adv. Neural Inf. Process. Syst.* 2378–2388.
[7] Diego Gonzalez, Guillermo Botella, Soumak Mokheerje, and et al. 2011. FPGA-Based acceleration of block matching motion estimation techniques. In *Int. Conf. Field Program. Log. Appl.* 389–392.
[8] Xin He, Liu Ke, Wenyan Lu, and et al. 2018. AxTrain: Hardware-oriented neural network training for approximate inference. In *Proc. Int. Symp. Low Power Electron. Des.* 20:1–20:6.
[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.
[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, and et al. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proc. 22nd ACM Int. Conf. Multimed.* New York, NY, USA, 675–678.
[11] Jianan Li, Tingfa Xu, and Kun Zhang. 2017. Real-time feature-based video stabilization on FPGA. *IEEE Trans. Circuits Syst. Video Technol.* 27, 4 (2017), 907–919.
[12] Jinghao Lu, Chunyuan Li, Jonnathan Singh-Alvarado, and et al. 2018. MIN1PIPE: A miniscope 1-photon-based calcium imaging signal extraction pipeline. *Cell Rep.* 23, 12 (2018), 3673–3684.
[13] Eftychios A. Pnevmatikakis and Andrea Giovannucci. 2017. NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data. *J. Neurosci. Methods* 291 (2017), 83–94.
[14] Cong Shi, Jie Yang, Ye Han, and et al. 2014. A 1000fps vision chip based on a dynamically reconfigurable hybrid architecture comprising a PE array and self-organizing map neural network. In *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Pap.* 128–129.
[15] Christoph Stosiek, Olga Garaschuk, Knut Holthoff, and et al. 2003. In vivo two-photon calcium imaging of neuronal networks. *Proc. Natl. Acad. Sci. U. S. A.* 100, 12 (2003), 7319–7324.