Fast Calcium Trace Extraction for Large-Field-of-View Miniscope

Zhe Chen
Computer Science Department
UCLA
Los Angeles, U.S.
zhechen@ucla.edu

Garrett J. Blair
Department of Psychology
UCLA
Los Angeles, U.S.
gblair92@gmail.com

Hugh T. Blair
Department of Psychology
UCLA
Los Angeles, U.S.
tadblair@ucla.edu

Jason Cong
Computer Science Department
UCLA
Los Angeles, U.S.
cong@cs.ucla.edu

2021 IEEE Biomedical Circuits and Systems Conference (BioCAS) | 978-1-7281-7204-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/BIOCAS49922.2021.9644936

***Abstract*—Recent advancement in miniaturized calcium imaging microscope enables monitoring the cell activity from hundreds of neurons simultaneously *in vivo*. However, extracting calcium traces from a large population of cells in real time under a strict resource and energy constraint remains a challenge. To overcome it, we design a customized accelerator on a low-power FPGA for fast calcium trace extraction. This enables us to extract calcium traces from hundreds of cells in real time with a short and deterministic latency. In addition, we propose a series of techniques, including the region segmentation, the double buffering and the fast forward mechanisms, to further reduce the latency with minimal overhead. Applying these techniques, our implementation can achieve real-time calcium trace extraction for a maximum of 1024 cells from a 512×512 calcium video with sub-ms latency, which is promising in support of closed loop neurofeedback applications.**

***Index Terms*—Calcium image, FPGA, latency, trace extraction**

I. Introduction

The miniaturized calcium imaging microscope is one of the most exciting advancements in the neural recording field during the past decade [1], [2]. It can be head-mounted on a freely behaving rodent and record spike-related calcium fluorescence from hundreds of cells at a certain brain region in real time. Fig. 1(a) shows the experimental setup of a rat wearing a head-mounted miniaturized microscope (also "Miniscope" in Fig. 1(b), http://www.miniscope.org). Fig. 1(c) presents a cropped 512×512 field-of-view (FOV) calcium image captured by the Miniscope with 760 detected and superimposed cell contours. Miniaturized microscopes with ever increasing spatial resolution and imaging quality are widely used in different neuroscience research fields, including memory, navigation, and social behavior, to name a few.

Several real-time calcium image processing pipelines have been proposed [3]–[5]. [3] used the constrained non-negative matrix factorization (CNMF) approach to perform accurate and simultaneous cell contour and trace extractions. [4] established a dataflow based system to realize efficient online neuron detection and signal extraction in real time. [5] combined the CNMF and the recurrent neural network (RNN) inference to carry out neural signal extraction from noisy calcium images. These pipelines mainly target offline calcium image analysis and are usually developed on general-purpose

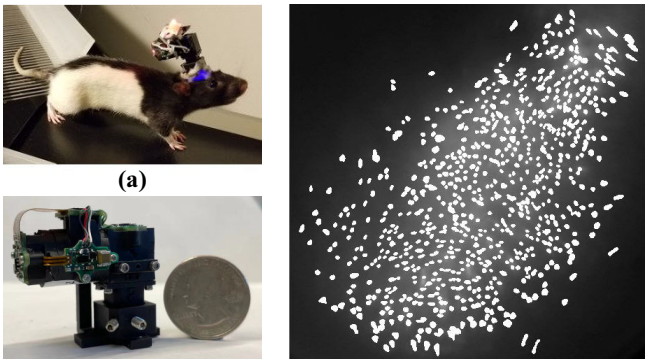

Fig. 1. Miniaturized microscope enables in vivo calcium imaging at a certain brain region. a) A freely behaving rat wearing a head-mounted Miniscope. b) The Miniscope device. c) Calcium image frame with 760 superimposed cell contours detected from a 6-min recording session.

CPUs or GPUs. Most of these implementations take a batch of images as input, which prevents them from achieving a short latency for each frame. Considering the millisecond timing precision of the cell firing and the optogenetic intervention [6], the challenge to implement an efficient calcium trace extraction for a large population of cells under a strict latency requirement for closed-loop applications remains.

In this paper, we design a customized accelerator on a field-programmable gate array (FPGA) for fast and efficient trace extraction from calcium images. We summarize our main contributions as:

- We introduce a tracing accelerator and a dedicated cell mapping algorithm for frame-based real-time calcium trace extraction on an FPGA.
- We propose the region segmentation, double buffering and fast forward mechanisms to reduce the the calcium trace extraction latency with minimal overhead.
- We implement the first FPGA-based system that can take input from the Miniscope and extract calcium traces from 1024 cells in a 512×512 FOV with sub-ms latency.

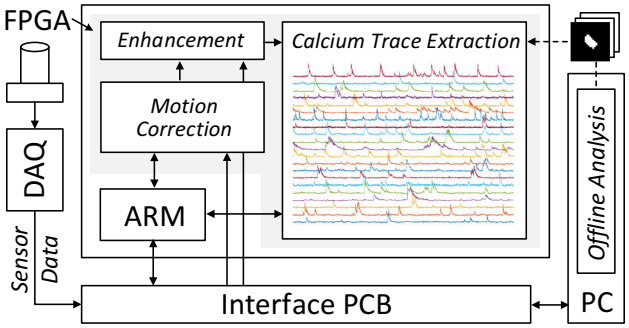

Fig. 2. FPGA-based real-time calcium image processing system.

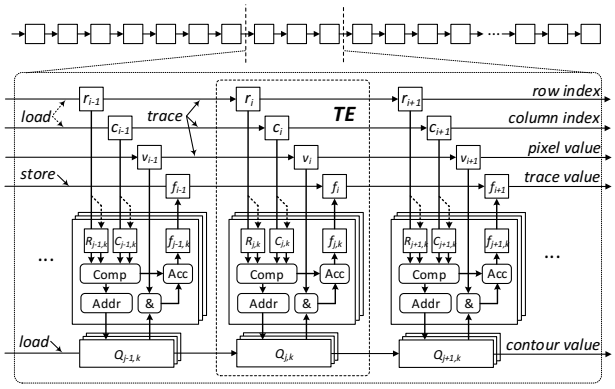

Fig. 3. Microarchitecture of the proposed tracing accelerator.

II. Real-Time Trace Extraction

A. Real-Time Calcium Image Processing

We develop a customized calcium image processing pipeline on a system on chip (SoC) device that integrates the FPGA and the ARM processor as shown in Fig. 2. The FPGA takes the input image from the data acquisition (DAQ) board and the ARM processor sends the processing results to a host computer over the Ethernet. Our customized pipeline is composed of 3 steps: 1) The rigid motion correction stabilizes the images based on template matching [8]. 2) The image enhancement improves the signal-to-noise ratio by eliminating the estimated background [5]. 3) The calcium trace extraction obtains traces based on N binary $N_C \times N_C$ cell contours with corresponding centers (R_i,C_i), i$\in$[1,N]. It calculates the trace value by accumulating pixel values under a binary mask for each cell at every frame. The cell contours are detected by the offline CNMF analysis [3]. We focus on 3) as the runtime of the 1) and 2) can largely overlap with the image sensor readout.

B. Tracing Accelerator

The tracing accelerator is composed of a chain of J tracing elements (TEs) as illustrated in Fig. 3. Each TE contains registers for 9-bit row and column indices (r_i,c_i), an 8-bit pixel value v_i and a 16-bit trace value f_i, a local memory for storing K cell contours, and the computation logic for calculating the trace values from the image stream and cell contours:

$$f_{j,k} = \sum_{dr_{ijk},dc_{ijk}=0}^{N_C} v_i \cdot Q_{j,k}(dr_{ijk}, dc_{ijk}), \qquad (1)$$

where $f_{j,k}$ is the trace value for the kth cell mapped to the jth TE, and $Q_{j,k}$ represents the cell contour corresponding to this specific cell. dr_{ijk} and dc_{ijk} are derived indices calculated by:

$$\begin{cases} dr_{ijk} = r_i & R_{j,k} + N_C/2 \\ dc_{ijk} = c_i & C_{j,k} + N_C/2 \end{cases}. \qquad (2)$$

The row and column indices, and the pixel and trace values stream down the TE chain at 1 load and store per clock cycle throughput. The tracing accelerator operates in 3 steps. 1) *Load*: Cell centers and contours are preloaded through the tracing chain to local indices registers and contour memory. 2) *Compute*: The image data flows through the tracing chain row by row at 1 pixel/cycle throughput. The distributed TEs perform the trace value calculation by accessing the local indices and pixel registers and the local contour memory in a massive parallel fashion. 3) *Store*: The calculated trace value at each TE shifts back through the tracing chain in a streaming fashion. The tracing accelerator reuses the index registers for the *Load* and *Compute* steps.

C. Cell Mapping

As we implement the local contour memory as a simple dual-port BRAM on the FPGA, each TE can fetch only one contour value from the local memory at a single clock cycle. This means that cells with overlapping contours cannot be mapped onto the same TE because of the memory access conflict. We propose an algorithm to guarantee that the cells mapped to the same TE do not have overlapping contours:

First, we allocate each cell contour with a specific location (j,k) on the $J \times K$ map and perform a conflict screening which identifies those cells that have overlapping contours with other cells within their allocated TEs, as seen in Fig. 4(a). Second, we loop through all the conflict cells on the map in the order of the TEs. For each conflict cell p mapped to a TE$_m$, we randomly pick up another cell q currently mapped to another TE$_n$ $(m \neq n)$ to form a pair (p,q). Third, we swap cells p and q and check for any conflicts between the cell p and the rest of mapped cells in TE$_n$ as well as between the cell q and the rest of mapped cells in TE$_m$. If there is no conflict, then the algorithm executes this swap, updates the map accordingly and goes on to address the next conflict cell. Otherwise, the algorithm gives up this swap trial and returns to the second step to randomly pick up another cell.

We tested the mapping algorithm with a 1024-cell dataset in which each cell had 8 duplicands. It finished remapping with 702 swap steps in 0.55 s on a single thread CPU.

D. Performance Analysis

If we suppose all cell contours can be stored in the tracing accelerator, then the trace extraction can finish within a single

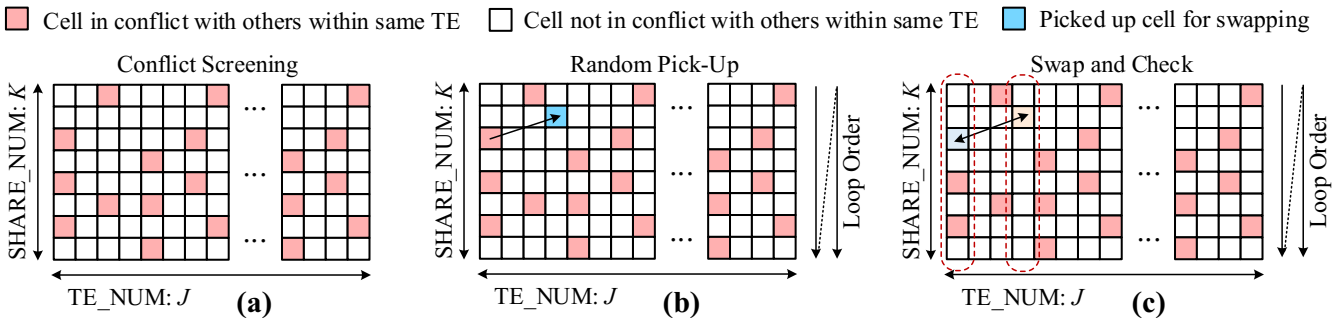

Fig. 4. The cell mapping operations ensure that there is no conflict in memory access of contours among cells allocated to the same TE.

TABLE I
FPGA RESOURCE UTILIZATION FOR THE TRACING ACCELERATOR

	J	*K*	LUT	FF	BRAM
No Reuse	128	8	83743	56516	86
4x Reuse	32	8	21180	14280	38

pixel-scan round. However, this oftentimes requires too much of the hardware resources. In order to fit the design onto an FPGA with limited resources, we reduce the number of TEs in the tracing accelerator and reuse the accelerator for multiple rounds of pixel scans. Table I shows the FPGA resource utilizations for a 128-TE full-mapping design and a 32-TE quarter-mapping design of the tracing accelerators. We can reuse the 32-TE design 4 times to complete the same 1024-cell trace extraction task as the 128-TE design.

Considering a reuse time of R, we estimate the clock cycle count for the trace extraction with the following formula:

$$Cycle = (N_C^2/8 + 2) \cdot N + L^2 \cdot R, \tag{3}$$

where L denotes the image size. The load of the cell centers and the store of the calcium traces cost N cycles each. The load of the cell contours costs $N \cdot N_C^2/8$ cycles as 8 binary contour values are coalesced into one 8-bit value during the loading. The tracing computation takes $L^2 \cdot R$ cycles. In a typical case, if L=512, R=4 and N_C=25, then the tracing computation dominates the cycle count, and increasing the number of cells N does not quite affect the overall runtime. According to a cycle-accurate simulation, the FPGA accelerator finishes the trace extraction and all pre-processing steps with 4.66 ms latency at 300 MHz clock frequency.

III. Latency Optimization

In this section, we introduce three mechanisms that can further reduce the latency for the proposed tracing accelerator.

A. Region Segmentation

The first mechanism is segmenting the FOV of the image into R regions, as shown in Fig. 5. Without the region segmentation (RS), each round of trace extraction requires the entire FOV image to be scanned, because the cell contours distribute evenly across the FOV. With the RS, we constrain the cell locations within $1/R$ of the full FOV region under each segment. So each round of scan concentrates on a single subregion, reducing the runtime of the tracing by (R-1)/R. Note that it requires a few rows of overlap between subregions to cover the cells at the boundaries.

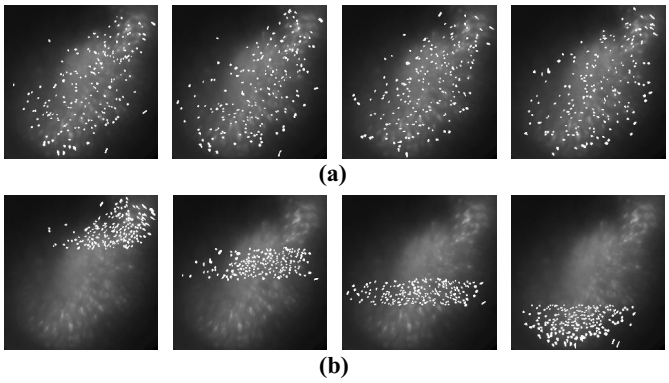

Fig. 5. Contour distributions before (a) and (b) after the region segmentation.

We implement the RS by sorting and segmenting the contours before mapping. First, we sort all of the contours by their center locations and divide them equally into R segments. Then, we apply the algorithm introduced in Section 2.3 to map the contours to the tracing accelerator for each segment. The sorting and segmentation are performed offline, and they cause no additional overhead on the hardware implementation.

B. Double Buffering

As the *Compute* step is optimized by the RS, we can no longer ignore the *Store* and *Load* time. Double buffering (DB) is a common technique for overlapping the computation with the communication. For our tracing accelerator, as the *Store* and *Load* time is shorter than the *Compute* time, we apply the DB to completely hide the communication time.

In order to realize the DB, we divide the single chain of TEs into two separate chains. As one chain enters the *Store* and *Load* process, the other chain starts the *Compute* process. The DB design removes the first term from the latency estimation in Eq. 3 whereas it does not significantly increase overhead as the number of TEs remains unchanged.

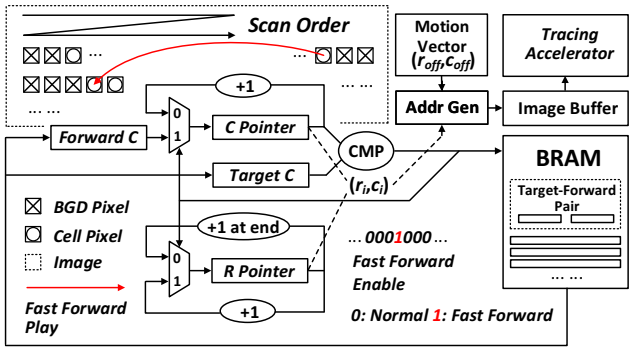

Fig. 6. The fast forward mechanism skips over pixels in the background.

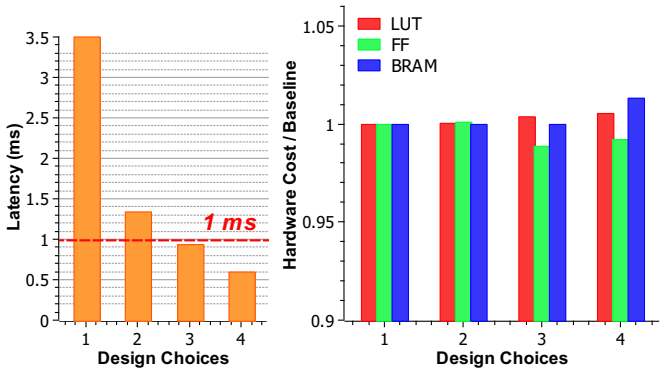

Fig. 7. Latency and hardware cost comparison among design choices: (1) the baseline, (2) the RS, (3) the RS+DB and (4) the RS+DB+FF optimizations.

C. Fast Forward

We observe that the contour distribution is very sparse at the boundary of the calcium image captured by the Miniscope. Scanning over background pixels with no cell contour coverage makes no effect and wastes computation time and resources. As a result, we come up with a fast forward (FF) mechanism that skips over background pixels at the beginning and the end of each row. We define background pixels as those without the cell contour coverage.

Fig. 6 shows the hardware implementation of the FF mechanism. R and C pointer registers keep track of the row and column indices of pixels during the scan process. Multiplexers ahead of the R and C pointer registers can switch between the normal and FF modes. Under the normal mode, the C pointer increments by 1 at every clock cycle and the R pointer increments by 1 only at the end of a row. Under the FF mode, the C pointer updates its value with the forward C index, and the R pointer increments by 1 jumping to the next row. A comparator takes both the current C pointer value and the target C index as inputs. When the C pointer value equals the target C index, it triggers a fast forward event. It enables an update of the forward and target C indices and triggers a new pair of forward and target indices to be fetched from a local BRAM. An address generator calculates the pixel indices from the R and C pointer values (r_i, c_i) and the motion vector (r_{off}, c_{off}) [8]). Pixel values are streamed to the tracing accelerator based on the calculated pixel indices at 1 pixel/cycle throughput.

In the L=512 case, we only need to store 512 entries in the local BRAM. Each entry is an 18-bit value, combining the 9-bit target and forward indices. It requires 1.15 kB memory, which can be implemented by a single BRAM block on the FPGA. It is worth mentioning that the proposed FF implementation can also benefit other stencil computation problems where the input data has a similar sparse feature.

D. Evaluation

We evaluated our proposed latency optimization mechanisms and implementations with a 760-cell calcium image dataset recorded from a real rat by the Miniscope. Fig. 7 shows the comparison results on the overall latency and the hardware cost for the tracing accelerator across different combinations of the optimization mechanisms. As the results show, we can reduce the overall latency to $<$1 ms by taking full advantage of the proposed latency optimizations while keeping the hardware cost overhead minimal.

IV. System and Implementation

A. Real-Time Processing System

We implemented the proposed calcium image processing pipeline on the Ultra96 FPGA at 300 MHz. Based on that, we developed a real-time processing system as presented in Fig. 8. We built a customized interface PCB for connecting the Miniscope DAQ board and a host computer to the Ultra96 FPGA. The image sensor data is transferred over the PCB to the FPGA through the general I/O pins at 66.67 MHz. 512×512 images are processed at a 22.8-fps frame rate. The system can stream both the image and extracted traces to the host computer over the Ethernet on the PCB in real time. We also developed a graphical user interface on the host computer for sending commands to the FPGA, receiving motion corrected calcium images and displaying calcium images and extracted traces in real time.

Table II summarizes the FPGA resource utilization. This includes the tracing accelerator, the rest of the processing modules in the pipeline and a virtual image sensor for the debugging purpose. The measured power consumption of our real-time system is 5.3 W, with a standby power of 2.2 W.

B. Performance

The implemented accelerator achieves 589 μs latency on extracting calcium traces from 760 cells from a 512×512

TABLE II
FPGA Resource Utilization for the Real-Time Calcium Trace Extraction on Ultra96

	LUT	FF	BRAM	DSP
Available	70560	141120	216	360
Utilization	47199	52985	201	111
Utilization %	66.9	37.6	93.1	30.8

calcium video when applying all latency optimizations. It can maintain the sub-ms latency when the number of traced cells reaches 1024. We demonstrated the real-time calcium image trace extraction for 1024 cells by using the virtual sensor, which replayed 1000 frames of recorded calcium images in real time. The relative error of our trace extraction among all cells on the FPGA is less than 0.04% compared to the offline simulation. Our trace extraction implementation on the FPGA achieves 2.9x and 132x speedup against a high-end multi-core CPU using 4 threads and the embedded ARM processor on the Ultra96, respectively.

We can extend our proposed FF mechanism to skip over all background pixels instead of just those at both ends of the rows. In order to achieve this goal, the BRAM needs to store more indices, which increase the hardware cost. We evaluated the benefit and cost of using such an aggressive FF compared to the FF introduced in Section 3.3 across datasets collected from 6 different rats. Fig. 9 shows the comparison results. The aggressive FF contributes to 21.3% latency reduction on average, but it requires 7.2x the memory for storing the indices. We took the reduced cycle count per index (RCPI) as a metric for the trace extraction. The FF outperforms the aggressive FF consistently by a mean RCPI of 105 over 21. Although the aggressive FF causes more memory overhead, it can be useful for applications where the latency requirement is high and the extra hardware cost is affordable.

C. Discussion

Unlike prior works that mainly focused on getting real-time throughput on general purpose platforms [3]–[5], we leveraged customized hardware acceleration to achieve deterministic and short latency for the calcium trace extraction, which offers unique advantage to neurofeedback closed-loop applications. Considering the voltage imaging can reach 1k fps frame rate [7], our proposed customized calcium image processing can be a promising solution for closed loop neurofeedback applications that require millisecond response time.

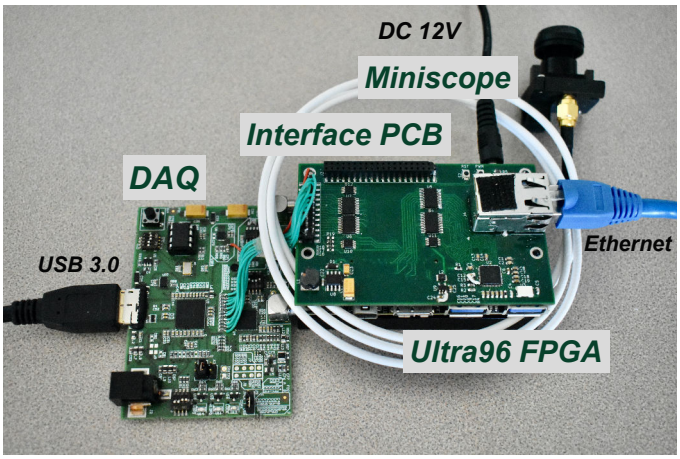

Fig. 8. Real-Time calcium image processing hardware setup (flexible instead of rigid coaxial cable is used for real experiments with rats).

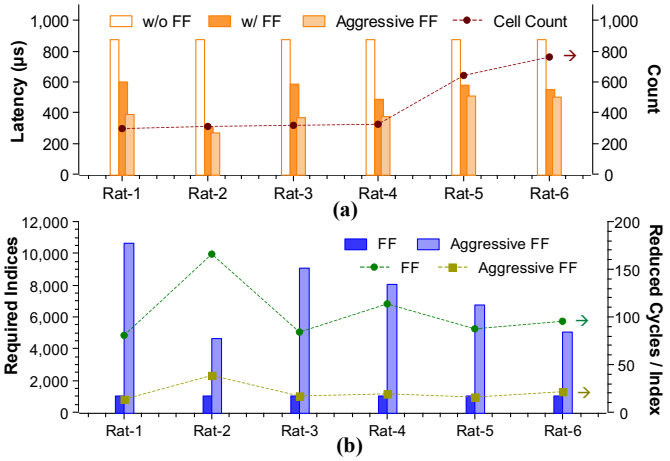

Fig. 9. Experimental analysis on the tradeoff between the latency and the memory cost for the FF.

V. Conclusion

In this paper, we propose a tracing accelerator design, a corresponding cell mapping algorithm, and dedicated latency optimizations that can extract calcium traces from a large population of cells in calcium image video with sub-ms latency. Our implementation has the potential to enable a variety of closed-loop feedback experiments based on the Miniscopes and other high temporal resolution in vivo neuron imaging techniques for brain research.

Acknowledgment

This work is supported by the NSF under Grant No.: CCF-1436827 and No.: DBI-1707408. The authors would like to thank Prof. Peyman Golshani, Prof. Daniel Aharoni and Dr. Changliang Guo for their support on the Miniscope device.

References

[1] K. K. Ghosh, L. D. Burns, E. D. Cocker, A. Nimmerjahn, Y. Ziv, A. El Gamal, and et al., "Miniaturized integration of a fluorescence microscope," Nature Methods, vol. 8, p. 871, 2011.

[2] D. Aharoni, B. S. Khakh, A. J. Silva, and P. Golshani, "All the light that we can see: a new era in miniaturized microscopy," Nature Methods, vol. 16(1), pp. 11–13, 2019.

[3] J. Friedrich, A. Giovannucci, and E. A. Pnevmatikakis, "Online analysis of microendoscopic 1-photon calcium imaging data streams," PLoS Computational Biology, vol. 17(1), no. e1008565, 2021.

[4] Y. Lee, J. Xie, E. Lee, S. Sudarsanan, D.-T. Lin, R. Chen, and et al., "Real-time neuron detection and neural signal extraction platform for miniature calcium imaging," Frontiers in Computational Neuroscience, vol. 14, p. 43, 2020.

[5] J. Lu, C. Li, J. Singh-Alvarado, Z. C. Zhou, F. Fröhlich, R. Mooney, and et al., "MIN1PIPE: a miniscope 1-photon-based calcium imaging signal extraction pipeline," Cell Reports, vol. 23(12), pp. 3673-–3684, 2018.

[6] M. Häusser, "Optogenetics: the age of light," Nature Methods, vol. 11(10), pp. 1012-–1014, 2014.

[7] A. Kazemipour, O. Novak, D. Flickinger, J. S. Marvin, A. S. Abdelfattah, J. King, and et al., "Kilohertz frame-rate two-photon tomography," Nature Methods, vol. 16(8), pp. 778-–786, 2019.

[8] Z. Chen, H. T. Blair, and J. Cong, "LANMC: LSTM-assisted non-rigid motion correction on FPGA for calcium image stabilization," Proc. of the ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA), pp. 104-–109, 2019.