# PulseSync: Deterministic MCU–NPU Streaming Co-Design for SRAM-Only rPPG Inference

Pin-Hao Chen, and Ching-Han Chen

***Abstract*—We present *PulseSync*, a communication-aware MCU–NPU co-design framework for streaming rPPG inference on an SRAM-only platform. Its *Rolling Temporal Feature Queue (RTFQ)* buffers compact per-frame features instead of ROI frames, reducing temporal-buffer SRAM by 98.4% (91.1 KiB → 1.5 KiB). Combined with asynchronous double-buffered zero-copy execution, software-managed cache-coherent coordination, and Helium-optimized FP32 temporal fusion, PulseSync achieves 3.61 ms per-frame processing latency at 30 FPS within a 1.53 MB full-system on-chip SRAM budget on STM32N6570-DK. On-device evaluation on eight subject-disjoint UBFC-rPPG test subjects yields a mean absolute error in heart rate (HR-MAE) of 6.75 BPM against synchronized contact-PPG ground truth. A 30-minute continuous run logged all 1,800 expected HR records at 1 Hz without gaps.**

***Index Terms*—Embedded systems, MCU–NPU co-design, on-chip SRAM, streaming inference, rPPG, low-power.**

## I. Introduction

RECENT edge SoCs integrate neural processing units (NPUs) alongside microcontrollers (MCUs), enabling on-device remote photoplethysmography (rPPG) [1], [2]. SRAM-only streaming inference faces two coupled challenges: ROI-frame caching consumes scarce on-chip SRAM [3], [4], while concurrent camera, MCU, and NPU operation requires explicit buffer ownership and cache coherence to avoid races and stalls [5], [6].

Reported model-level memory for existing rPPG designs does not establish full-system SRAM feasibility, which also requires accounting for capture buffers, activations, and temporal state. Complementing existing MCU–NPU execution frameworks [7], PulseSync addresses these storage and coordination requirements for continuous rPPG inference.

Its contributions are: (i) a *Rolling Temporal Feature Queue (RTFQ)* that replaces ROI-frame caching with compact features, reducing temporal-buffer SRAM by 98.4% and eliminating historical spatial recomputation; (ii) an *asynchronous double-buffered zero-copy pipeline* that decouples capture from inference through barrier-ordered buffer handoff; and (iii) a *communication-aware feature bridge* with software-managed cache coherence and epoch-based non-blocking NPU dispatch.

## II. Background and Positioning

Signal-processing rPPG methods are lightweight but sensitive to motion and illumination changes [8], [9]. PhysNet [10] and PhysFormer [11] learn spatiotemporal representations, while ME-rPPG targets memory-efficient inference [4]. Continual video models reduce temporal recomputation [12], and FPGA/ASIC accelerators improve computational efficiency through specialized dataflows and fixed-point arithmetic [13]–[15]. PulseSync complements these approaches by addressing temporal storage and cache-coherent MCU–NPU coordination.

TABLE I
Deployment context and on-device baseline.

| Method | Platform | Memory (MB) | | Latency (ms) |
|---|---|---|---|---|
| | | Model-side | Full-system | |
| PhysNet [10] | GPU/Server | ∼230 | — | 62.10 |
| PhysFormer [11] | GPU/Server | ∼576 | — | 2330.0 |
| ME-rPPG [4] | Edge AI | ∼3.6 | — | 9.46 |
| POS [9] | STM32N6 | — | 1.54 | 9.70 |
| **PulseSync** | STM32N6 | — | **1.53** | **3.61** |

Literature resource figures follow [4]; STM32N6 results are measured in this work. Full-system SRAM includes capture buffers, inference memory, temporal state, and runtime queues. — denotes an unreported or inapplicable value.

Table I separates model-side memory from full-system SRAM to distinguish accounting scopes. Cross-platform latencies are not directly comparable because hardware and protocols differ. We implement plane-orthogonal-to-skin (POS) [9] on the same STM32N6 platform as a direct on-device baseline.

## III. PulseSync System Design

### A. System Overview and Hardware Partitioning

PulseSync is implemented on the STM32N6570-DK, integrating an Arm Cortex-M55 core at 800 MHz and a Neural-ART NPU at 1 GHz, with on-chip AXI-SRAM accessible to both compute units. The pipeline (Fig. 1) comprises: (1) camera capture via the Digital Camera Interface Pixel Pipeline (DCMIPP), (2) ROI preprocessing on the MCU, (3) INT8 spatial 2D-CNN inference on the NPU, and (4) FP32 temporal 1D-CNN fusion on the MCU over the RTFQ.

We deploy a fixed-shape spatial graph for per-ROI NPU inference. Temporal fusion remains in FP32 because INT8 temporal quantization increased mean subject-level HR-MAE from 6.75 to 9.66 BPM on the same held-out UBFC subjects. To reduce MCU latency while retaining FP32 arithmetic, the temporal stage uses Cortex-M55 Helium vector instructions. This partition combines NPU spatial acceleration with an accuracy-preserving choice of temporal precision.

### B. Asynchronous Double-Buffering Pipeline

Two frame buffers (FB0 and FB1) are statically allocated in AXISRAM3 as capture and processing buffers. The

P.-H. Chen and C.-H. Chen are with the Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan (e-mail: pierre@g.ncu.edu.tw).

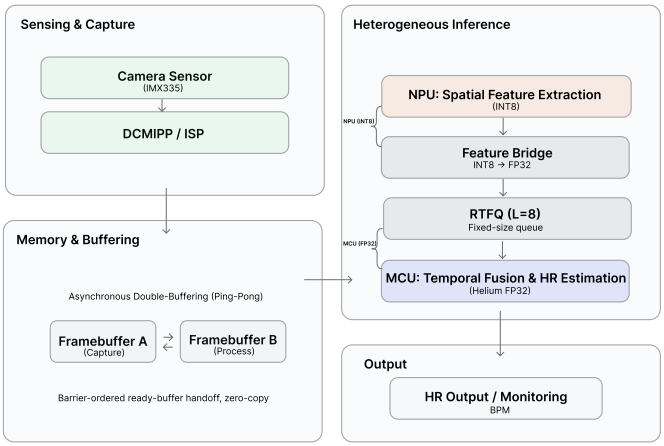



Fig. 1. PulseSync architecture with INT8 spatial extraction on the NPU and Helium-optimized FP32 temporal fusion on the MCU.

memory protection unit (MPU) maps this region as inner-shareable, non-cacheable Normal memory, preventing stale D-Cache reads without per-frame invalidation. After each DCMIPP frame-complete event, the interrupt service routine (ISR) records the completed-buffer index and frame number, executes `__DMB()`, and then publishes the ready flag. After observing the flag, the main-loop consumer executes a corresponding barrier and processes a local snapshot of the ready-buffer index. This ordering makes buffer metadata visible before consumption, supporting producer–consumer separation and zero-copy frame handoff [16], [17].

As shown in Fig. 2, the measured 3.61 ms per-frame processing latency fits within the 33.3 ms frame period at 30 FPS, leaving approximately 29.7 ms of slack during normal operation.

### C. Communication-Aware Feature Bridge

NPU inputs use model-defined INT8 quantization [18]:

$$q = \mathrm{clip}_{[-128,127]}\left(\mathrm{round}(x/s) + z\right), \tag{1}$$

where $x$ is the floating-point value, $s$ the scale, and $z$ the zero-point. Outputs are dequantized as $x = s(q - z)$ using the corresponding output parameters before entering the FP32 RTFQ. All quantization parameters are determined offline.

MCU–NPU exchange buffers are cacheable, unlike capture buffers. The bridge cleans inputs before dispatch and invalidates outputs after completion for coherence [6]. Epoch matching rejects completions unrelated to the active request. A 200 ms watchdog initiates recovery on stalls; this exceeds one frame period, so uninterrupted outputs during recovery are not guaranteed.

## IV. Rolling Temporal Feature Queue (RTFQ)

For each frame $t$, PulseSync extracts features $f_{\mathrm{ROI}_i,t} \in \mathbb{R}^D$ from three ROIs (forehead and two cheeks), forming

$$F_t = \mathrm{concat}(f_{\mathrm{ROI}_0,t},\ f_{\mathrm{ROI}_1,t},\ f_{\mathrm{ROI}_2,t}) \in \mathbb{R}^{3D}. \tag{2}$$

The RTFQ stores $L$ vectors as an $L \times 3D$ temporal input. With $L = 8, D = 16$, each frame contributes 48 FP32 values to an $8 \times 48$ queue.

**Frame-level commit.** $F_t$ is enqueued only if all three ROI inferences succeed; otherwise, the queue remains unchanged and continued HR outputs reflect the previous window. This prevents partial-frame commits but does not detect ROI misalignment. The 30-minute run rejected 0 of 54,000 frames.

**Sliding window and cold start.** For $L$ consecutive successful frames,

$$Q_t = [F_{t-L+1}, F_{t-L+2}, \ldots, F_t]. \tag{3}$$

Each commit overwrites the oldest slot (Fig. 3), giving constant-time updates for fixed $D$ and memory usage independent of stream duration. Following rejections, the latest $L$ committed frames need not be consecutive camera frames.

No HR estimate is produced until $L$ valid entries are available. Without drops, collecting eight frames at 30 FPS spans seven intervals (233 ms); the first HR additionally requires processing the eighth frame and completing temporal inference.

**Memory efficiency.** Uint8 ROI-frame caching requires $M_{\mathrm{frame}} = 3LHWC$ bytes, versus $M_{\mathrm{RTFQ}} = 12LD$ bytes for FP32 features. With $L = 8$, $D = 16$, $H = W = 36$, and $C = 3$, storage decreases from 93,312 bytes (91.1 KiB) to 1,536 bytes (1.5 KiB), a 98.4% temporal-buffer reduction. Spatial extraction uses per-channel weight and per-tensor activation INT8 quantization; Helium-optimized FP32 temporal kernels use static allocation without runtime dynamic allocation.

## V. Experimental Evaluation

### A. Setup

Experiments use the STM32N6570-DK platform and partitioning described in Section III. The post-optimization linker map confirms 1.53 MB of full-system SRAM, dominated by approximately 1.5 MB of frame buffers in AXISRAM3. NPU activations reside in AXISRAM5 and the 1,536-byte RTFQ in AXISRAM1. Deployment uses ST Edge AI Core v2.2.0-20266 and X-CUBE-AI v10.2.0. The Cortex-M55 Data Watchpoint and Trace (DWT) cycle counter measures processing latency over 200 frames, from ROI preprocessing through HR estimation, covering all three ROIs and one temporal update per frame.

Following public-dataset evaluation practices [19], training uses 34 UBFC-rPPG videos [20] and 19 rPPG-10 videos [21]. Eight subject-disjoint UBFC-rPPG videos, excluded from training and INT8 calibration, are replayed frame-by-frame through the deployed pipeline against synchronized contact-PPG ground truth. HR-MAE is the unweighted mean of eight subject-wise MAEs, each computed over that subject's evaluated windows. With identical inputs and preprocessing, the Helium-optimized STM32 and PC reference produce identical HR sequences on these eight videos (maximum absolute difference: zero).

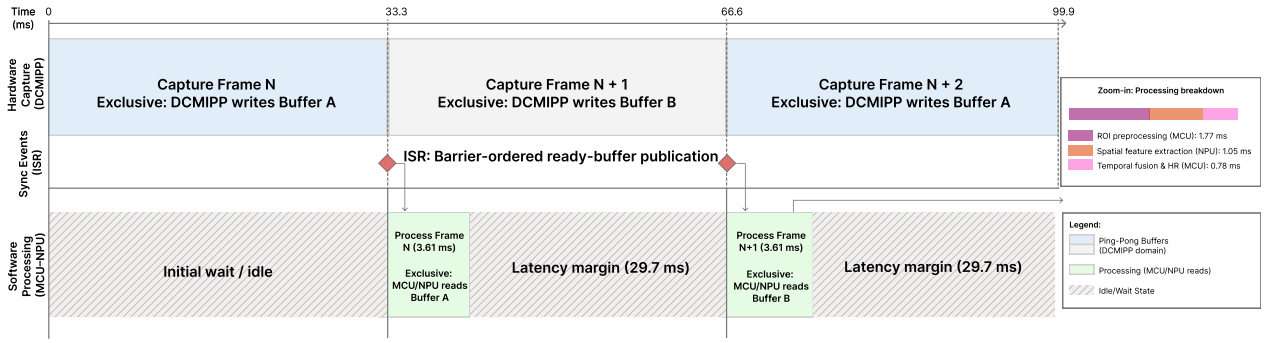



Fig. 2. Asynchronous double buffering with barrier-ordered handoff and a 3.61 ms processing interval within each 33.3 ms frame period.

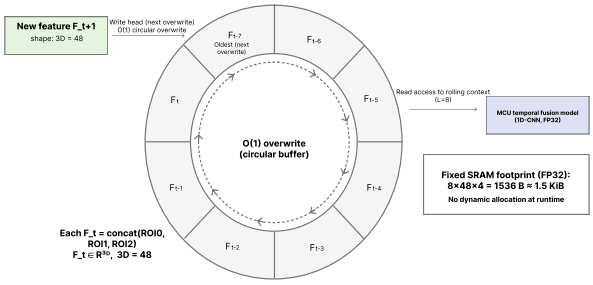



Fig. 3. RTFQ with $L$ circular slots, each storing a $3D$-dimensional FP32 feature vector.

TABLE II
PER-FRAME LATENCY OF THE HELIUM-OPTIMIZED PIPELINE.

| Stage | Mean (ms) | Std (ms) | Max (ms) |
|---|---|---|---|
| ROI preprocessing (MCU) | 1.77 | 0.00 | 1.78 |
| Spatial extraction (NPU) | 1.05 | 0.01 | 1.07 |
| Temporal fusion & HR (MCU) | 0.78 | 0.00 | 0.78 |
| Total processing pipeline | 3.61 | — | — |

ROI preprocessing and spatial extraction each include all three ROIs. Statistics are rounded to two decimal places; 0.00 does not imply zero variation. — denotes an unreported statistic.

### B. Latency, Throughput, and Stability

Table II reports per-frame latency. Helium vectorization, four-way time-position unrolling, and pre-padded boundaries remove inner-loop boundary checks and reduce loop and indexing overhead. Temporal fusion and HR estimation decrease from 5.30 to 0.78 ms (6.79× versus scalar C), reducing total processing latency from 8.12 to 3.61 ms (2.25×). The temporal share decreases from approximately 65% to 22%; ROI preprocessing now accounts for the largest share, approximately 49%.

A 30-minute run at 30 FPS processes 54,000 frames and logs all 1,800 expected HR records at 1 Hz without gaps or watchdog timeouts.

### C. RTFQ Ablation and Parameter Sensitivity

The implemented no-RTFQ baseline caches $L = 8$ ROI frames. It first evaluates the current frame's three ROIs, then recomputes the full cached window ($8 \times 3$ ROIs), including the current frame again. Thus, it performs 27 ROI evaluations per update versus three with RTFQ. Both use the same Helium temporal kernel (Table III). RTFQ reduces temporal-buffer SRAM by 98.4% and processing latency from 18.24 to 3.61 ms (5.05×). This speedup is relative to the implemented baseline, including its duplicated current-frame computation.

TABLE III
IMPLEMENTED BUFFERING VARIANTS WITH THE SAME HELIUM TEMPORAL KERNEL.

| Method | Temporal buffer (KiB) | ROI evaluations per update | Latency (ms) |
|---|---|---|---|
| ROI-frame caching | 91.1 | 27 | 18.24 |
| RTFQ | 1.5 | 3 | 3.61 |

TABLE IV
TEMPORAL-LENGTH AND FEATURE-WIDTH TRADE-OFFS.

| $L$ | $D$ | RTFQ (bytes) | NPU spatial (ms/ROI) | MCU temporal (ms/frame) | HR-MAE (BPM) |
|---|---|---|---|---|---|
| 4 | 16 | 768 | — | 0.50 | 8.12 |
| 8 | 16 | 1536 | 0.35 | 0.78 | 6.75 |
| 16 | 16 | 3072 | — | 1.44 | 6.98 |
| 8 | 8 | 768 | 0.33 | — | 7.77 |

At fixed $D$, $L$ does not change per-frame spatial work. — denotes latency not separately reported; SRAM covers only the RTFQ.

Table IV compares $L$ and $D$ on the same held-out subjects, with hardware-measured latency. At $D = 16$, $L = 4$ increases HR-MAE to 8.12 BPM, whereas $L = 16$ offers no observed accuracy gain over $L = 8$ despite greater storage and temporal latency. At $L = 8$, $D = 8$ increases HR-MAE to 7.77 BPM while saving 768 bytes of queue storage and 0.02 ms of NPU latency per ROI. These trade-offs support retaining $L = 8, D = 16$ among the evaluated configurations.

### D. Public-Dataset Validation

Table V reports results on eight held-out UBFC-rPPG subjects. Mixed precision achieves a mean subject-level HR-MAE of 6.75 BPM, compared with 9.66 BPM for full INT8 and 7.95 BPM for our on-device POS implementation using the same test videos and ROI preprocessing. The prediction head directly regresses window-level HR without reconstructing a pulse waveform or beat-to-beat intervals, so waveform SNR

TABLE V
HR ESTIMATION ON EIGHT HELD-OUT UBFC-RPPG SUBJECTS.

| Deployment | MAE (BPM) | RMSE (BPM) | HR Pearson $r$ |
|---|---|---|---|
| INT8 spatial + FP32 temporal | $6.75 \pm 3.10$ | 8.93 | 0.40 |
| Full INT8 | 9.66 | 11.38 | 0.38 |
| POS | 7.95 | — | — |

Ground truth is synchronized contact PPG. MAE, RMSE, and Pearson $r$ are computed within each subject and averaged without weighting; $\pm$ denotes the SD of subject-wise MAEs. Pooled over all windows, $r$ is 0.78 for mixed precision and 0.68 for full INT8. Correlations depend on aggregation and are not directly comparable across protocols. — denotes an unreported metric.

and HRV are not evaluated. Pearson correlation therefore describes HR estimates, not waveform agreement.

Subject 20 has the highest mixed-precision MAE (14.05 BPM) and a mean reference HR of 126.02 BPM. Only 0.58% of raw training windows have HR labels at or above this level. Sparse upper-tail coverage is a plausible contributor to this error, although training uses an HR-balanced sampler and the raw proportion does not represent sampling frequency.

A PC-based sensitivity test independently perturbs each ROI's horizontal and vertical coordinates per frame, uniformly within $\pm 4$ or $\pm 8$ pixels at native resolution before cropping and resizing. Mean HR-MAE increases from 6.75 to 11.52 and 13.36 BPM, respectively, with subject-dependent effects. This temporally uncorrelated perturbation tests sensitivity to ROI misalignment; deployed ROI coordinates remain fixed between detection updates. The eight-subject, single-dataset test cohort limits generalization, and absent demographic and skin-tone metadata preclude subgroup conclusions. These results establish deployment feasibility rather than state-of-the-art accuracy.

### *E. Power Consumption*

A HiDANCE USB inline meter with 0.01 W resolution measures steady-state board-input power of 2.40 W during camera streaming without inference and 2.41 W with the full pipeline operating at 30 FPS. The observed incremental board-level average power is approximately 0.01 W, equal to one meter increment. This measurement includes platform overhead and does not isolate MCU or NPU compute power; the difference is reported at the instrument's resolution limit.

## VI. CONCLUSION

PulseSync's RTFQ replaces ROI-frame caching with compact features, reducing temporal-buffer SRAM by 98.4%. The MCU–NPU system achieves 3.61 ms per-frame processing latency at 30 FPS within 1.53 MB of full-system SRAM on STM32N6570-DK, with 6.75 BPM mean HR-MAE on eight held-out UBFC-rPPG subjects and gap-free HR logging over 30 minutes. The small cohort and ROI-misalignment sensitivity limit accuracy generalization; broader subject validation and cross-platform evaluation remain future work.

## REFERENCES


[1] Z. Lin, Y. Xue, Z. Wang, and J. Cai, "Remote physiological measurement with multiscale feature extraction and global attention module," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2468–2479, 2024.

[2] Y. Li, J. Huang, J. Zhao, D. Wu, and M. Zheng, "TS-CAN+: A improved TS-CAN architecture for non-contact heart rate measurement," *IEEE Transactions on Consumer Electronics*, vol. 71, no. 1, pp. 1393–1401, 2025.

[3] S. Somvanshi, M. M. Islam, G. Chhetri, R. Chakraborty, M. S. Mimi, S. A. Shuvo, K. S. Islam, S. Javed, S. A. Rafat, A. Dutta, and S. Das, "From tiny machine learning to tiny deep learning: A survey," *ACM Computing Surveys*, vol. 58, no. 7, 2025.

[4] K. Wang, J. Tang, Y. Fan, J. Ji, Y. Shi, and Y. Wang, "Memory-efficient low-latency remote photoplethysmography through temporal-spatial state space duality," *arXiv preprint*, 2025, arXiv:2504.01774.

[5] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[6] ARM Limited, "Implementing dma on arm smp systems," ARM Ltd., Tech. Rep. Application Note 228 (ARM DAI 0228A), Aug. 2009, open Access Application Note. [Online]. Available: https://documentation-service.arm.com/static/5ed10086ca06a95ce53f8632

[7] J. Kwon, J. Lee, and H. Kim, "Pipelining of a mobile SoC and an external NPU for accelerating CNN inference," *IEEE Embedded Systems Letters*, vol. 16, no. 2, pp. 150–153, 2024.

[8] W. Verkruysse, L. O. Svaasand, and J. S. Nelson, "Remote plethysmographic imaging using ambient light," *Optics Express*, vol. 16, no. 26, pp. 21 434–21 445, Dec. 2008. [Online]. Available: https://doi.org/10.1364/OE.16.021434

[9] W. Wang, A. C. den Brinker, S. Stuijk, and G. de Haan, "Algorithmic principles of remote ppg," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 7, pp. 1479–1491, 2017.

[10] Z. Yu, X. Li, and G. Zhao, "Remote photoplethysmograph signal measurement from facial videos using spatio-temporal networks," in *British Machine Vision Conference*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:199001018

[11] Z. Yu, Y. Shen, J. Shi, H. Zhao, P. Torr, and G. Zhao, "PhysFormer: Facial video-based physiological measurement with temporal difference transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 4176–4186.

[12] L. Hedegaard and A. Iosifidis, "Continual 3D convolutional neural networks for real-time processing of videos," in *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 369–385.

[13] B. Chen, M. T. Khan, G. Goussetis, M. Sellathurai, Y. Ding, J. F. C. Mota, and J. Lee, "COMET: Co-optimization of CNN models using efficient-hardware OBC techniques," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2026.

[14] M. T. Khan and M. A. Alhartomi, "Digit-serial DA-based fixed-point RNNs: A unified approach for enhancing architectural efficiency," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 5, pp. 8240–8254, 2025.

[15] M. A. Alhartomi, M. T. Khan, S. Alzahrani, A. Alzahmi, R. A. Shaik, J. Hazarika, R. Alsulami, A. Alotaibi, and M. Al-Harthi, "Low-area and low-power VLSI architectures for long short-term memory networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 4, pp. 1000–1014, 2023.

[16] C. Zinner and W. Kubinger, "ROS-DMA: A DMA double buffering method for embedded image processing with resource optimized slicing," in *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, 2006, pp. 361–372.

[17] S. W. Kim, S. Park, J. Jun, and Y. Han, "Design and implementation of display stream compression decoder with line buffer optimization," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 3, pp. 322–328, 2019.

[18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[19] X. Liu *et al.*, "rPPG-toolbox: Deep remote PPG toolbox," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 68 485–68 510.

[20] S. Bobbia, R. Macwan, Y. Benezeth, A. Mansouri, and J. Dubois, "Unsupervised skin tissue segmentation for remote photoplethysmography," *Pattern Recognition Letters*, vol. 124, pp. 82–90, 2019.

[21] G. Rodrigues and N. Garcia, "rPPG-10," Mendeley Data, Version 1, 2025.