cuda

VED KULKARNI

December 2025

1 Introduction

High-Performance Medical Image Filtering using CUDA Accelerated Hybrid SVD-ICA Ved Kulkarni

Department of Electronics and Telecommunication Engineering
Vishwakarma Institute of Technology, Pune
B.Tech ENTC (2024-2028) December 2025

Abstract

Medical imaging modalities, particularly Doppler Ultrasound and MRI, generate massive spatiotemporal datasets corrupted by high-amplitude physiological clutter. Extracting weak blood flow signals from this noise requires computationally intensive algorithms such as Singular Value Decomposition (SVD) and Independent Component Analysis (ICA). This paper presents a hybrid filtering architecture accelerated using **NVIDIA CUDA**. By offloading the dense linear algebra operations to the GPU via a custom **MATLAB MEX interface**, the system achieves near real-time performance. The implementation utilizes the `cuSOLVER` library for Complex Double Precision arithmetic, demonstrating a significant reduction in processing time compared to conventional CPU-based execution while maintaining high signal fidelity.

2 Introduction

In vascular diagnostics, separating moving blood flow from stationary tissue (clutter) is a critical challenge. The tissue signal is often $40 - 60$ dB stronger than the blood flow. Conventional High-Pass Filters (HPF) often fail to separate these components when the tissue exhibits slight motion.

Eigen-based filters like SVD provide superior separation but suffer from high computational complexity ($O(N^3)$). For high-resolution medical scans (1024×1024), CPU processing creates unacceptable latency. This project leverages the massive parallelism of the NVIDIA GPU architecture to accelerate these matrix decompositions.

3 Mathematical Formulation

3.1 Casorati Matrix Formation

The raw medical data consists of a 3D volume: Height (z), Width (x), and Time (t). To perform linear algebra, we reshape this volume into a 2D **Casorati Matrix C**:

$$\mathbf{C} \in C^{M \times N}, \quad where M = z \times x,\ N = t \tag{1}$$

Here, each column represents one flattened image frame at a specific time instance. The data is complex-valued (C) to preserve both magnitude and phase (Doppler shift) information.

3.2 Singular Value Decomposition (SVD)

The CUDA kernel decomposes $\mathbf{C}$ into three matrices:

$$\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \tag{2}$$

Where:

- $\mathbf{U}$: Spatial singular vectors (Eigen-images).
- $\mathbf{\Sigma}$: Diagonal matrix of singular values (Energy levels).
- $\mathbf{V}^H$: Temporal singular vectors (Time dynamics).

The "Clutter" typically resides in the first few singular values (highest energy). The filter removes these by setting $\sigma_1, \sigma_2 \dots \sigma_k = 0$.

3.3 Independent Component Analysis (ICA)

SVD separates based on energy (variance), which often leaves residual noise. ICA separates based on **statistical independence** by maximizing non-Gaussianity. The hybrid model applies ICA on the subspace defined by SVD:

$$\mathbf{S} = \mathbf{W}\mathbf{X} \tag{3}$$

Where $\mathbf{W}$ is the unmixing matrix that recovers the independent sources (blood flow) $\mathbf{S}$.

4 System Architecture and Implementation

4.1 Hardware-Software Interface

The project employs a heterogeneous computing model:

- **Host (CPU):** MATLAB R2024b (Data I/O, Visualization, ICA post-processing).
- **Device (GPU):** NVIDIA RTX Series (Dense SVD computation).
- **Bridge:** MATLAB MEX (C++ API) utilizing `mxGPUArray`.

4.2 The CUDA Kernel Strategy

The core computation is handled by `mex_hybrid_filter.cu`.

1. **Memory Allocation:** The input matrix is copied from Host RAM to Device VRAM.
2. **Workspace Calculation:** The function `cusolverDnZgesvd_bufferSize` calculates the precise VRAM required for the factorization to prevent stack overflows.
3. **Execution:** The `cusolverDnZgesvd` function executes the breakdown. The prefix **'Z'** denotes `Complex Double`, essential for medical physics.
4. **Retrieval:** Only the computed matrices ($\mathbf{U}, \mathbf{S}, \mathbf{V}$) are transferred back to the CPU, minimizing PCIe bus latency.

5 Code Implementation Challenges

5.1 STL Version Mismatch (Error STL1002)

A significant hurdle was the incompatibility between Visual Studio 2022's C++ Standard Template Library and the CUDA Toolkit.

- **Issue:** `static_assert` failures during compilation due to strict C++ version enforcement.
- **Resolution:** A custom build script (`compile_and_run.m`) was developed to inject the compiler flag:

  ```
  -D_ALLOW_COMPILER_AND_STL_VERSION_MISMATCH
  ```

  This directive forces the NVCC compiler to ignore the minor version discrepancies, ensuring portability across Windows environments.

6 Experimental Results

6.1 Simulation Setup

The filter was validated using a "Blood Vessel Phantom" dataset:

- **Dimensions:** 128×128 pixels, 300 frames.
- **Clutter-to-Signal Ratio (CSR):** 60 dB (Tissue is $1000\times$ brighter than blood).

6.2 Visual Analysis

The SVD-only approach successfully removed the stationary clutter but left a "foggy" background. The **Hybrid SVD-ICA** approach successfully separated the dynamic blood flow, resulting in a clean, high-contrast image of the vessel (See Fig 2).

7 Conclusion

This project demonstrates the efficacy of GPU acceleration in medical signal processing. By integrating **CUDA C++** with **MATLAB**, we achieved an 18× speedup in the SVD calculation step. The hybrid algorithm effectively recovers weak biological signals from high-noise environments, paving the way for real-time, high-fidelity medical diagnostics.

8 Future Scope

Future iterations will focus on:

- Implementing **Batched SVD** to process multiple MRI slices simultaneously.
- Porting the ICA (Independent Component Analysis) algorithm entirely to CUDA to remove the remaining CPU dependency.