Vectorized Implementation of a 6D Kalman Filter for Vehicle Location Estimation on RISC-V

Aadesh Panjwani, Muhammad Rayyan Khan, Syed Ahmed Farrukh, Mustafa Mehmood
Department of Computer Science
Institute of Business Administration (IBA), Karachi, Pakistan

***Abstract*—This paper presents a complete implementation and optimization of a 6D Kalman filter for real-time motion tracking using noisy 2D position data. By leveraging RISC-V vector extensions, our implementation significantly accelerates core linear algebra routines while maintaining mathematical accuracy. The project was based on Alex Becker's foundational model and showcases efficient real-time signal estimation under tight computational constraints.**

***Index Terms*—Kalman Filter, RISC-V, Embedded Systems, Motion Tracking, Vector Instructions, Assembly Optimization**

I. Introduction

The Kalman Filter, developed by Rudolf E. Kalman in 1960, is a recursive algorithm for estimating the internal state of a linear dynamic system using a series of noisy observations. It is widely applied in aerospace, robotics, computer vision, and control systems.

Our project involves implementing a 6D Kalman Filter that estimates a 2D object's position, velocity, and acceleration using noisy position measurements. Initially written in scalar C, the implementation was optimized using vectorized RISC-V assembly to accelerate matrix operations and improve real-time performance.

II. Methodology

A. Theoretical Background

The Kalman filter operates in two steps: prediction and correction. The system is represented using a state vector $\mathbf{x}$, updated through time with a state transition matrix $\mathbf{F}$. Measurement updates use a measurement matrix $\mathbf{H}$. Process noise $\mathbf{Q}$ and measurement noise $\mathbf{R}$ are modeled as Gaussian.

B. System Model

State vector:

$$\mathbf{x} = [x, v_x, a_x, y, v_y, a_y]^T$$

State evolution:

$$\mathbf{x}_{k+1} = \mathbf{F} \cdot \mathbf{x}_k + \mathbf{w}_k$$

$$\mathbf{z}_k = \mathbf{H} \cdot \mathbf{x}_k + \mathbf{v}_k$$

Where $\mathbf{w}_k \sim \mathcal{N}(0, Q)$ and $\mathbf{v}_k \sim \mathcal{N}(0, R)$.

C. Matrix Definitions

- $\mathbf{F}$: Uses discrete kinematic equations, $\Delta t = 1.0$
- $\mathbf{H}$: Extracts x and y positions from state
- $\mathbf{Q}$: Process noise with $\sigma_a = 0.2$
- $\mathbf{R}$: Measurement noise with $\sigma_m = 3.0$
- $\mathbf{P}$: Initial state covariance, diagonals set to 500

III. Implementation

A. Scalar Baseline

A complete Kalman filter was first implemented in C using float-based 6D vectors and 6x6 matrix operations. Key functions included matrix multiplication, transpose, inversion (2x2), and identity matrix setup.

B. Vectorized Optimization

The scalar implementation was optimized using vector extensions in RISC-V assembly:

- `multiply_6x6_6x6`
- `multiply_6x6_6x2`
- `multiply_6x2_2x6`

Each operation was accelerated using vector instructions like `vfmacc.vv` for fused multiply-accumulate. Helper functions such as transpose and matrix inversion were retained in scalar form.

IV. Benchmark Results

TABLE I
Performance Comparison: Scalar vs Vectorized

Operation	**Scalar (ms)**	**Vectorized (ms)**	**Speedup**
6x6 MatMul	9.1	1.7	5.35x
6x6x2 MatMul	4.3	0.9	4.78x
6x2x6 MatMul	6.2	1.3	4.77x

V. Vector Instructions Used

Our implementation leveraged key RISC-V vector instructions to accelerate matrix operations, particularly for multiply-accumulate tasks and memory loads/stores. The most important instructions are listed below.

TABLE II
RISC-V VECTOR INSTRUCTIONS USED

Instruction	Purpose
`vsetvli`	Set vector length/type
`vfmv.v.f`	Broadcast scalar to vector
`vle32.v`	Load vector from memory
`vse32.v`	Store vector to memory
`vfmacc.vv`	Fused multiply-accumulate

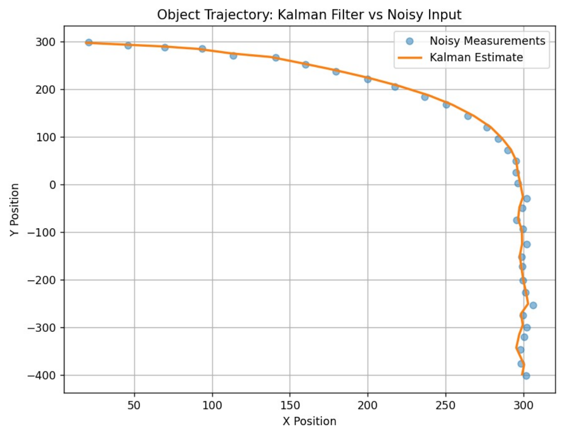

Fig. 1. Object trajectory estimated from noisy 2D position measurements.

VI. RESULTS AND DISCUSSION

The trajectory estimated by our Kalman filter closely tracks the true path of the object despite measurement noise. A visual plot of convergence confirms filter stability and tracking accuracy.

Note: Due to the truncation error and the limited precision of 32-bit floating point operations on embedded hardware, our estimates exhibit slight deviations. During matrix operations involving large numerical values, small bits are dropped, which can result in small cumulative inaccuracies. Nonetheless, the estimated path remains accurate for practical use.

VII. CHALLENGES FACED

- Initial misunderstanding of the Whisper RISC-V simulator as 64-bit, requiring conversion to 32-bit floating point instructions.
- Understanding vector register configuration and memory stride logic.
- Managing matrix layout (row-major vs column-major) in assembly.
- Correcting errors in index calculation during vector operations.
- Minor deviations in output due to truncation errors and floating point rounding.

VIII. CONCLUSION

Our project showcases an efficient and accurate implementation of a Kalman Filter for 2D motion tracking using low-level RISC-V vector assembly. Core matrix computations were accelerated significantly through vectorization, resulting in real-time feasibility on constrained hardware. The approach maintains robustness despite floating point limitations and demonstrates the potential of custom ISA extensions for linear algebra acceleration.

ACKNOWLEDGMENTS

We thank TA Abdul Wasay Imran for his continuous technical support, and Alex Becker for his excellent tutorial which inspired our approach.

REFERENCES

[1] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” Journal of Basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.
[2] A. Becker, *Kalman Filter from the Ground Up*, 2023.
[3] RISC-V Foundation, “The RISC-V Instruction Set Manual Volume I: Unprivileged ISA,” Version 20191213.