Real-Time mmWave Radar Point-Cloud Preprocessing on Resource-Constrained Microcontrollers

Andreas Nicolaou

TU Delft

1. Background

Why Human Pose Estimation?

Human Pose Estimation systems track the position and movement of human limbs, often in real-time. These systems have exceptional applications in areas like **Interactive Gaming** (e.g. the XBox Kinect), and patient monitoring in **Healthcare**.

Why mmWave Radars?

Millimeter wave radars use **high-frequency electromagnetic waves** to detect points on objects. They have many **advantages over camera sensors** when applied to Human Pose Estimation:

1. privacy-preserving,
2. robust to lighting conditions,
3. lower cost.

Figure 1. The Microsoft Kinect, a camera-based sensor used for HPE

These properties make mmWave radars **attractive for monitoring systems,** such as **human-pose estimation**.

The Challenge of mmWave Point Clouds

Radar detections are transformed into point-cloud (**PC**) arrays. A point-cloud has 5 dimensions and describes a point in 3-D space on the surface of an object [1]. Each point contains:

1. X, Y, Z Cartesian Coordinates
2. Doppler Effect $\propto$ Velocity
3. Signal Intensity

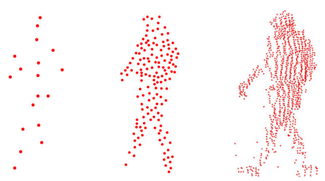

Figure 2. An example of point-clouds

Unlike technologies like LIDAR, mmWave point clouds are **highly irregular** and **non-uniform**, requiring **extensive pre-processing** before they can be used in AI Models.

2. Problem Statement

The goal of HPE systems is to be deployed on **Edge Devices**, which involves implementing the end-to-end system on **resource-constrained devices** such as **Micro-controller units (MCU)**

MCUs introduce major challenges:

1. Limited Memory
2. Low processing power
3. Low-level languages

It is currently **unclear** whether python-based pre-processing pipelines developed for testing and development can run efficiently on embedded hardware in real time. This study answers the following research question:

How efficiently can mmWave radar point-cloud pre-processing pipelines be executed on memory-constrained micro-controllers, and can they meet real-time performance requirements?

3. System Overview

Target System

1. STM32 ARM Cortex-M7 @ 200 MHz
2. 320 KB SRAM
3. Bare-metal C implementation
4. 32-bit Floating Point Unit (FPU)
5. DMA Controller with memory-to-memory transfer support

Figure 3. Arm Cortex-M7 Illustration

End-to-End Pipeline

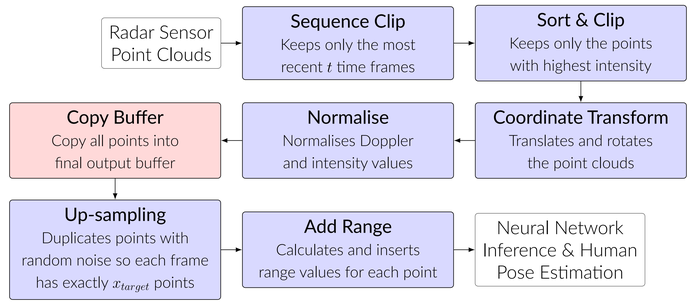

Figure 4. End-to-end mmWave radar human pose estimation pipeline. Highlighted stages represent the pre-processing pipeline evaluated on the micro-controller. Red stages were proposed to improve efficiency.

4. Optimization Methods Explored

Minimizing Latency

The **Up-Sampling** operation is the **most intensive** stage in the pipeline. It requires existing points to be **copied** to a new buffer, and new points to be **generated** using **random sampling**.

Memory Transfer Parallelization was explored using the **DMA Controller**. The DMA Controller may be used to offload the copy operation, while the CPU generates new points. This method reduces total latency by an average of **0.5 ms**.

Random Number Generation using the lightweight **XORShift** algorithm yielded the best up-sampling performance.

Redundant Operations were eliminated by structuring the pipeline using **Clip First, Process Later** and **Process Before Duplicating** policies.

Minimizing Memory Footprint

Temporary memory buffers dominate the peak memory usage of the pipeline. **Consolidating memory operations** in one stage, where a single terminal output buffer is instantiated allowing all processing operations to run **in-place**, reduces both **latency** and **memory usage**.

Improving Cache Utilization

Processing **stage-by-stage** under-utilizes the cache by iterating over data multiple times. We propose the **single-pass** approach, which executes all operations on each piece of data, and completes a **single iteration** over all data, minimizing **expensive cache misses**. This method achieved the **best latency**.

6. Results

Performance results of 7 pipeline variants were gathered using 12 data samples.

The **Hardware FPU** improves average latency by a **factor of 6**, allowing for real-time pre-processing at **15.75 ms**.

The **XORShift** RNG algorithm enabled **faster up-sampling** than both the standard C `rand()` function and the board's **hardware RNG peripheral**.

Accelerated memory transfers using the **DMA Controller** yielded **slightly better performance** on all tested samples. The pipeline variant using **XORShift** and **DMA paralellization** ran in **14 ms** on average.

Eliminating redundant operations using a restructured flow dropped latency to **9.62 ms**, while using the **single-pass** implementation aimed at reducing cache misses dropped latency to an average of **8.13 ms**.

Variant	Peak Stack Usage (KB)	Average Latency (ms)
Without FPU	90.63	93.64
With FPU (Baseline)	90.71	15.75
With XORShift & DMA	90.74	13.98
Restructured Flow	50.08	9.62
Single-Pass	50.15	8.13

Table 1. Performance evaluation for the main pipeline variants, gathered over 12 samples

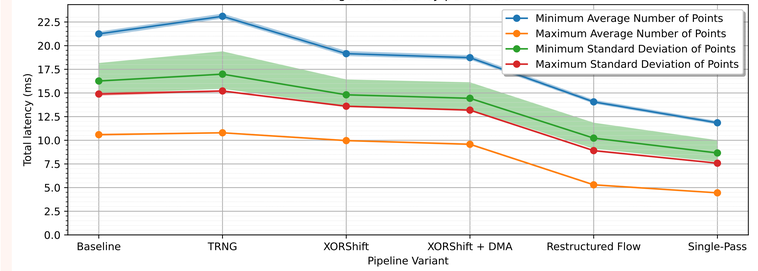

Figure 5. Average latency for each pipeline variant, for 4 groups of 3 dataset samples each. The error areas are bound by the minimum and maximum recorded latency. The statistical significance of each sample group is denoted in the legend.

7. Conclusions and Future Work

This work **validates and profiles** the efficiency and performance of mmWave point-cloud pre-processing pipelines on embedded devices. It was found that using optimization techniques such as **lightweight RNG** algorithms, **memory transfer parallelization**, **eliminating redundant operations** and **memory buffers**, and **single-pass iteration** approaches, mmWave pre-processing pipelines can run in **under 10 ms**, allowing for **responsive, real-time end-to-end HPE** systems on embedded devices.

Future work may include porting the **complete end-to-end HPE system** with a mmWave sensor for **real-time inference**.

References

[1] Harry D. Mafukidze, Amit K. Mishra, Jan Pidanic, and Schonken W. P. Francois. Scattering centers to point clouds: A review of mmwave radars for non-radar-engineers. *IEEE Access*, 10:110992–111021, 2022.

Responsible Professor: Marco Zuniga Zamallo

a.nicolaou@student.tudelft.nl

Supervisor: Nicole Rosi