Edge Computing for Mobile Robots: Multi-Robot Feature-Based Lidar Odometry with FPGAs

L. Qingqing[1], F. Yuhong[1], J. Peña Queralta[2], T. N. Gia[2], H. Tenhunen[3], Z. Zou[1] and T. Westerlund[2]

[1] School of Information Science and Technology, Fudan Universtiy, China
[2] Department of Future Technologies, University of Turku, Finland
[3] Department of Electronics, KTH Royal Institute of Technology, Sweden
Emails: [1]{qingqingli16, zhuo}@fudan.edu.cn, [2]{yuhofu, jopequ, tunggi, tovewe}@utu.fi, [3]hannu@kth.se

***Abstract*—Offloading computationally intensive tasks such as lidar or visual odometry from mobile robots has multiple benefits. Resource constrained robots can make use of their network capabilities to reduce the data processing load and be able to perform a larger number tasks in a more efficient manner. However, previous works have mostly focused on cloud offloading, which increases latency and reduces reliability, or high-end edge devices. Instead, we explore the utilization of FPGAs at the edge for computational offloading with minimal latency and high parallelism. We present the potential for modelling feature-based odometry in VHDL and utilizing FPGA implementations.**

***Index Terms*—Odometry; Autonomous Robots; FPGA; Lidar; Lidar Odometry; Laser rangefinder; SLAM; Mobile Robots;**

I. Introduction

The penetration of lidars in the robotic industries has considerably grown over the last two decades [1]. Efficient lidar odometry was proposed by Zhang *et al.* and has been extensively used in the development of autonomous robots and vehicles [2]. However, processing lidar data for odometry and mapping purposes requires on-board computers. A recent approach is to exploit the robot's connectivity to offload computationally intensive tasks to cloud servers. However, this increases latency and reduces reliability. Therefore, we study the offloading at the local network level, in order to tightly control the communication latency and provide a robust solution. With this approach, a single gateway can perform odometry calculations for multiple robots in real-time [3]. However, as the number of robots connected to the same gateway increases, a much more powerful processor is needed, and the control over the system latency can be partially lost. We propose the utilization of FPGAs as their parallelism enables rapid scaling without an impact on performance.

The edge computing paradigm is a distributed network paradigm that aims at moving computational processing power and data analytics closer to where the data originates, in the so-called edge of the network. In a hybrid Edge-Cloud architecture, part of the data processing is done at the local network level and only the analysis results are transmitted to cloud servers. This allows for reduced bandwidth and increased reliability. In robotics, this paradigm can increase the capabilities of resource-constrained robots by streaming sensor data and performing the analysis at edge gateways with higher parallelist and power efficiency [4].

II. Related Work

Offloading computationally intensive sensor data processing has been extensively studied in mobile robot navigation. Through the simplification of on-board hardware, significant saving in energy consumption can be achieved, incrementing the robot's battery life [3]. Integrating edge computing with mobile robot navigation is a relatively recent field and the related works are few. From a more generic point of view, Dey *et al.* studied the potential benefits of offloading computationally intensive tasks within simultaneous localization and mapping (SLAM) algorithms with edge computing [5]. The authors concluded that edge computing can bring significant enhancements to localization and positioning algorithms with low latency and high reliability services. Most other works have been focusing on offloading to cloud servers. Nonetheless, network connectivity to cloud servers can be unreliable and latency uncontrollable in some situations. Therefore, it is not suitable for time-critical applications such as localization and movement estimation in autonomous robots.

III. Feature-based lidar odometry on FPGAs

Lidar odometry algorithms use the lidar's data to compute the motion of a robot between two consecutive sweeps. In general, lidar odometry algorithms can be divided into three steps. The first step is to extract features from lidar data, the features can be the geometric distributions, or some stable points which can be observed in the two consecutive sweeps. The next step is to find the feature correspondence through the position difference between sweeps. The last step is estimating the lidar movement through the time between two sweeps [2]. FPGA's have the ability to process lidar data in real time with a limited resource utilization, and naturally parallelize the processing of data from multiple lidars [6].

Our goal in this work is to design a pure VHDL implementation. Instead of comparing complete lidar sweeps, as most recent implementations do, we aim at a implementation that analyzes lidar data as it is available and compares features in real-time. With this, we expect to be able to increase odometry accuracy and the positioning update frequency.

IV. Experiment and Results

In order to test the efficiency and usability of the feature-based odometry algorithm, we have first implemented in

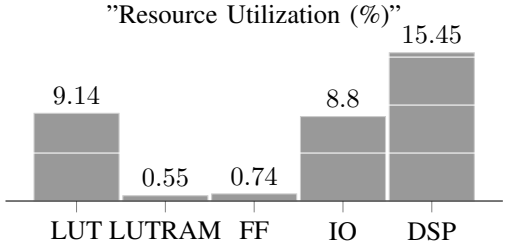

Fig. 1. FPGA Resource Utilization Summary

C++ within the Robot Operating System (ROS). ROS is the most popular open-source meta-operating system for robots to support code reuse in the development of robots. We have utilized the inexpensive RPLidar A1 for our experiment, a 360° two-dimensional lidar with 1° resolution at 10Hz.

A. FPGA Implementation

We have utilized a Zybo Z7-20 board for implementing our algorithm. This relatively small board is built around the Z-7010, the smallest chip in the Xilinx Zynq-7000 family. Even though the board integrates a dual-core ARM Cortex-A9 processor, we only use the FPGA logic in order to provide a generic design that can be easily ported to other platforms.

Implementing the proposed algorithm using VHDL hardware modelling presents some challenges. On one side, the need for calculation of trigonometric functions. In order to solve this, we utilize coordinate rotation digital computer (CORDIC) algorithms implemented in VHDL. In particular, we generate sine samples and from those calculate the values for cosine. On the other side, conversion types and integrating the CORDIC calculations into a state machine, requiring complimentary intermediate signals.

B. FPGA Resource Utilization

In order to study the potential of the FPGA-based implementation to be parallelized, we have synthesized an initial and unoptimized version of the design. The preliminary results show that the main resource utilization occurs with the IO banks (8.8%), Logic LUTs (9.14%) and DSP modules (15.45%). Nonetheless, the IO banks can be easily multiplexed, and additional lidars can be connected with a single input, so they do not represent a significant limitation. Moreover, a single nRF or Wi-Fi receiver can be utilized to receive information from multiple units. In terms of DSP utilization, the modules are used in the CORDIC implementations. A single CORDIC module can be shared among multiple parallel processes with a relatively simple state machine. Therefore, the proposed algorithm is not limited by the number of available DSPs. Similarly, we have estimated that around 90% of the LUTs can be shared. This is possible because the timing constraints that the frequency of the lidar scanner impose are very relaxed compared to the maximum performance that the VHDL implementation can deliver. Therefore, we expect that a single FPGA board will be able to run in parallel over 50 lidar odometry calculations. This can be combined with wireless communication solutions that provide enough available channels, such as Wi-Fi or nRF. In contrast, on an Intel Atom x5-Z8350 CPU @ 1.44GHz × 4, the proposed algorithm can be executed approximately 5 to 15 times in parallel, depending on whether other processes are being run.

TABLE I
PERFORMANCE COMPARISON BETWEEN FPGA AND CPU IMPLEMENTATIONS

	Xilinx Zynq XC7Z010 (VHDL Impl.)	**Intel Atom x5-Z8350** (C++ Impl.)
Aprox. Resource	10% (fixed) + 1%	4% (fixed) + 7% CPU
Max. concurrency	>20-50*	< 5-15

*Expected range with optimized code.

TABLE II
FPGA RESOURCE UTILIZATION BREAKDOWN

Resource Type	**Used**	**Available**
Slice LUTs	4961	53200
LUT as Logic	4865	53200
LUT as Memory	96	17400
Slice Registers	791	106400
F7 Muxes	32	26600
F8 Muxes	0	13300
Bonded IOB	11	125
DSPs	34	220

V. CONCLUSION AND FUTURE WORK

We have presented preliminary work on an odometry offloading solution for multi-robot systems. We have designed and implemented a feature-based lidar odometry algorithm that is flexible and can accommodate to a variety of indoors or outdoors environments, and implemented in an FPGA with pure VHDL modeling. The results presented in this paper show potential for high parallelism and low-latency odometry.

In future work, we will integrate this solution within a real team of mobile robots and test their navigability with minimum hardware on-board and edge offloading.

ACKNOWLEDGMENT

This work has been supported by NFSC grant No. 61876039, and the Shanghai Platform for Neuromorphic and AI Chip (NeuHeilium).

REFERENCES

[1] S. A. Hiremath *et al.* Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter. *Computers and Electronics in Agriculture*, 100:41 – 50, 2014.
[2] J. Zhang *et al.* Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.
[3] V. K. Sarker *et al.* Offloading slam for indoor mobile robots with edge-fog-cloud computing. In *ICASERT*, 2019.
[4] S. Biookaghazadeh *et al.* Are fpgas suitable for edge computing? In *USENIX HotEdge Workshop*, 2018.
[5] S. Dey *et al.* Robotic slam: a review from fog computing and mobile edge computing perspective. In *MobiQuitous*, pages 153–158. ACM, 2016.
[6] J. Peña Queralta *et al.* FPGA-based Architecture for a Low-Cost 3D Lidar Design and Implementation from Multiple Rotating 2D Lidars with ROS. In *IEEE SENSORS 2019 (SENSORS)*, 2019.