Architectures for SLAM and Augmented Reality Computing

Nikolaos Bellas
Christos D. Antonopoulos
Spyros Lalis
Maria-Rafaela Gkeka
Alexandros Patras
University of Thessaly
Volos, Greece

Georgios Keramidas
Iakovos Stamoulis
Think Silicon S.A.
Patras, Greece

Nikolaos Tavoularis
Stylianos Piperakis
Emmanouil Hourdakis
Panos Trahanias
FORTH
Heraklion, Greece

Paul Zikas
George Papagiannakis
Ioanna Kartsonaki
OramaVR
Heraklion, Greece

***Abstract*—In the next few years, new demanding applications will be supported on mobile platforms by reconciling two conflicting requirements: high performance (often with real-time limitations) and low power consumption. The objective of the *vipGPU* project is to develop hardware and software technology to provide efficient support for two such application scenarios, namely (a) simultaneous localization and mapping (SLAM) in mobile robotics systems, and (b) virtual reality (VR) in portable devices to simulate serious games with emphasis on simulating surgical interventions and medical training in general. In this project, we aim at developing a new heterogeneous platform consisting of hardware accelerators for low power embedded systems optimized (at the hardware and software level) for the implementation of the two applications mentioned above.**

***Index Terms*—Simultaneous Localization and Mapping, FPGA, Low Power, Augmented Reality, Approximate Computing**

I. Introduction

The proliferation of autonomous robots has created the need to construct highly accurate 3D maps of their observed environment and to track the position and the trajectory of these agents within these maps. However, the process of localizing a walking agent and mapping its environment, which is referred to as Simultaneous Localization and Mapping (SLAM), is hard to solve. This process typically merges data from various sensors such as stereo/mono and RGB-D cameras, laser scanners (lidars) and Inertial Measurement Units (IMUs) and involves a non-trivial amount of data processing.

The scenarios for the virtual reality system focuses on (i) evaluation of shaders to be used to simulate virtual characters, and (ii) evaluation of the robustness of the learning game in different medical scenarios. In virtual reality applications, the speed at which the image is refreshed is very important for the smooth presentation of virtual content with the minimum image refresh rate in virtual reality scenarios reaching 40 frames/sec.

In order for embedded systems to be useful they have to deliver high performance in a power constrained environment, typically in the order of few Watts for autonomous robots. Specialized architectures are required to execute parts of the SLAM algorithm and graphics rendering that are computationally demanding. In section II, we briefly describe the NEOX^{TM} architecture designed specifically for such workloads. In sections III and IV, we focus specifically on the visual SLAM algorithm and we present results from our experimental evaluation on an FPGA acceleration prototype. A variation of the SLAM algorithm presented in section IV will be ported and evaluated in the NEOX^{TM} platform.

II. The NEOX^{TM} Multicore Accelerator

The *vipGPU* platform is equipped with NEOX^{TM}, the commercial, programmable hardware accelerators offered by Think Silicon S.A. [1]. NEOX is a parallel multicore and multithreaded GPU architecture based on the RISC-V RV64C ISA instruction set. In the final version of the product, the number of cores will vary from 4 to 64 cores organized in 1-16 cluster elements with configurable cache sizes and thread counts. Depending on cluster / core configuration, NEOX compute power is expected to range from 12.8 to 409.6 GFLOPS at 800MHz with support for FP16, FP32 and optionally FP64 and SIMD instructions. Figure depicts a high-level design of NEOX^{TM}.

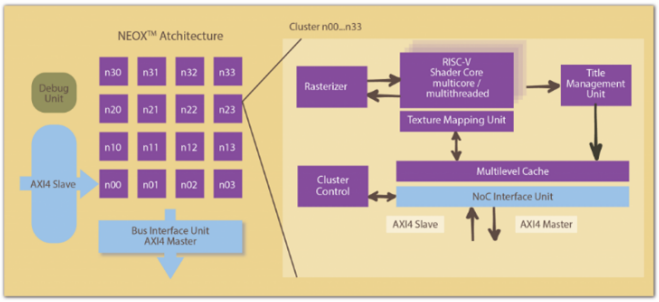

Fig. 1. The NEOX^{TM} Architecture

III. SLAM Algorithm

The application of visual SLAM in the domain of humanoid robots is a challenging task and remains an open-research problem.The SLAM algorithm used in this project is based on KinectFusion algorithm of the SLAMBench suite [2]. Our work in this project introduces a feature extraction based

loop closure extension on the KinectFusion pipeline, to help recover the agent's pose when the sensor input or aggressive approximations cause failures of the tracking system.

In this section, we start from the baseline C++/OpenMP implementation, and we develop precise as well as approximate hardware accelerators for the most important components of the algorithm as shown in Figure 2. The input to the algorithm are depth frames.

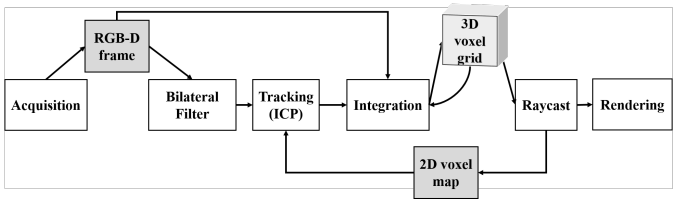

Fig. 2. KinectFusion data processing pipeline [2].

The **bilateral filter** is a low-pass edge-preserving filter that blurs the depth image in order to reduce the effects of noise and invalid depth values. **Tracking** estimates the 3D pose of the agent by registering the input depth frame with the 2D projection of the currently reconstructed model from the most recent camera position. **Integration** merges the corresponding depth map into the current 3D reconstructed model. KinectFusion utilizes a 3D voxel grid as the data structure to represent the global map, employing a truncated signed distance function (TSDF) to represent 3D surfaces [3]. **Raycasting** is a computer graphics algorithm used to render 3D scenes to 2D images.

Figure 3 shows the input and output of the SLAM algorithm for a scene. Profiling the SLAM code in x86-64 and ARM processors shows that the integration and raycast kernels contribute more than 70% of total execution time.

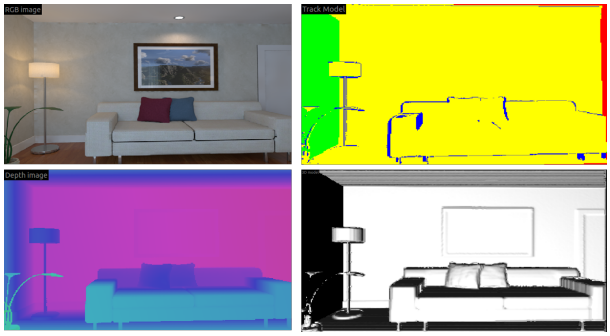

Fig. 3. The scene (top left), the input depth (RGB-D) frame (bot. left), the tracking output (top right), and the reconstructed 3D map (bot. right).

IV. Experimental Evaluation

A number of well known HLS optimizations such as loop unrolling, software pipelining and array partitioning are used to improve the performance of KinectFusion. Moreover, approximate source-level optimizations, which may affect the accuracy of the baseline code are also used to further improve performance. Approximate optimizations include loop perforation [4], reduced floating point precision, and, in general, skipping (or replacing) part of the computation not critical for the accuracy of agent tracking.

We implemented our designs using the Vitis™ Platform targeting the Xilinx UltraScale+ MPSoC ZCU102 Evaluation Kit. As input, we use three camera trajectories lr.kt[0-2] from ICL-NUIM, a synthetic dataset providing RGB-D sequences from a living room model [5].

To place an upper bound on errors, we discard all approximate configurations that increase the number of untracked frames wrt. the baseline.

Table I shows the performance improvements achieved by the fastest precise and fastest approximate accelerators. The last column shows the average RMSE per frame when only the corresponding kernel is approximate and all other kernels run precisely. For example, for a single bilateral filter HW accelerator, precise optimizations yield $705x$ speedup compared with the unoptimized HW implementation, whereas approximate optimizations further increase the speedup to $1044x$. For more information on the experimental evaluation of SLAM, refer to [6]. Our best FPGA design achieves 27.5 fps at an 320x240 input depth frame resolution without exceeding the tight error bounds necessary for tracking convergence.

TABLE I
Performance of HW kernel implementations (1 accelerator).

	Unopt.	**Fastest and Precise**	**Fastest and Approximate**	
	Hz	Hz (Speedup)	Hz (Speedup)	RMSE
Bilateral	0.54	380.5 ($705x$)	564 ($1044x$)	2.28
Tracking	0.49	17.7 ($36x$)	323 ($659x$)	2.02
Integration	0.72	10.1 ($14x$)	42.4 ($59x$)	2.54
Raycast (SW)	6.28	-	110 ($17.5x$)	2.07
Raycast (HW)	0.22	0.28 ($1.27x$)	5.7 ($25.9x$)	2.22

Acknowledgement

This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code: T1EDK-01149).

References

[1] https://think-silicon.com/tsi-products/hardware/neox-v/.

[2] B. Bodin *et al.*, "SLAMBench2: Multi-Objective Head-to-Head Benchmarking for Visual SLAM." in *ICRA*, 2018.

[3] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images." in *SIGGRAPH*, 1996.

[4] S. Sidiroglou-Douskos *et al.*, "Managing Performance vs. Accuracy Trade-Offs with Loop Perforation," in *ESEC/FSE, Szeged, Hungary, Sept.*, 2011.

[5] A. Handa *et al.*, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *ICRA Hong Kong, China, May*, 2014.

[6] MR Gkeka, A. Patras, C.D. Antonopoulos, S. Lalis, and N. Bellas, "FPGA Architectures for Approximate Dense SLAM Computing," in *Design and Test in Europe (DATE)*, 2021.