applied sciences

MDPI

Article

An FPGA Based Energy Efficient DS-SLAM Accelerator for Mobile Robots in Dynamic Environment

Yakun Wu [1], Li Luo [1], Shujuan Yin [2], Mengqi Yu [3], Fei Qiao [3,*], Hongzhi Huang [1], Xuesong Shi [4], Qi Wei [5] and Xinjun Liu [6,*]

1 School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China; 18120025@bjtu.edu.cn (Y.W.); lluo@bjtu.edu.cn (L.L.); 13213037@bjtu.edu.cn (H.H.)
2 School of Information Science and Technology, Baotou Teachers' College, Baotou 014030, China; 60466@bttc.edu.cn
3 Department of Electronic Engineering and BNRist, Tsinghua University, Beijing 100084, China; yumq17@mails.tsinghua.edu.cn
4 Intel Labs China, Beijing 100090, China; xuesong.shi@intel.com
5 Department of Precision Instrument, Tsinghua University, Beijing 100084, China; weiqi@tsinghua.edu.cn
6 Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China
* Correspondence: qiaofei@tsinghua.edu.cn (F.Q.); xinjunliu@mail.tsinghua.edu.cn (X.L.); Tel.: +86-138-1035-5024 (F.Q.); +86-135-5279-5847 (X.L.)

Abstract: The Simultaneous Localization and Mapping (SLAM) algorithm is a hotspot in robot application research with the ability to help mobile robots solve the most fundamental problems of "localization" and "mapping". The visual semantic SLAM algorithm fused with semantic information enables robots to understand the surrounding environment better, thus dealing with complexity and variability of real application scenarios. DS-SLAM (Semantic SLAM towards Dynamic Environment), one of the representative works in visual semantic SLAM, enhances the robustness in the dynamic scene through semantic information. However, the introduction of deep learning increases the complexity of the system, which makes it a considerable challenge to achieve the real-time semantic SLAM system on the low-power embedded platform. In this paper, we realized the high energy-efficiency DS-SLAM algorithm on the Field Programmable Gate Array (FPGA) based heterogeneous platform through the optimization co-design of software and hardware with the help of OpenCL (Open Computing Language) development flow. Compared with Intel i7 CPU on the TUM dataset, our accelerator achieves up to 13× frame rate improvement, and up to 18× energy efficiency improvement, without significant loss in accuracy.

Keywords: visual semantic SLAM; semantic segmentation; FPGA; hardware acceleration; mobile robots

Citation: Wu, Y.; Luo, L.; Yin, S.; Yu, M.; Qiao, F.; Huang, H.; Shi, X.; Wei, Q.; Liu, X. An FPGA Based Energy Efficient DS-SLAM Accelerator for Mobile Robots in Dynamic Environment. *Appl. Sci.* **2021**, *11*, 1828. https://doi.org/10.3390/app11041828

Academic Editor: Pavol Božek
Received: 10 December 2020
Accepted: 15 February 2021
Published: 18 February 2021

1. Introduction

Mobile robots are not only widely used in fields such as industry, agriculture, medical and services [1,2], but are also well used in dangerous situations such as urban security [3], national defense [4] and space detection [5,6]. With the development of artificial intelligence technology, mobile robots perform complex algorithms to complete more advanced tasks. The insufficient computing capability and limited energy of traditional computing platforms has become the bottleneck of intelligent robot development [7,8].

Mobile robots not only need to perceive the information of the surrounding environment but to also clearly determine their position, which leads to the SLAM (Simultaneous Localization and Mapping) algorithm. Nowadays, the visual SLAM framework is quite mature, mainly composed of critical parts such as visual odometry, optimization, loop closing and mapping. However, visual SLAM has some problems with the dynamic environment and the demand for advanced tasks, such as the lack of understanding about environmental information and the stability of the system in dynamic environments [9].

Semantic information introduced through deep learning is more and more tightly integrated with SLAM to solve these problems. On the one hand, semantic SLAM can improve human-computer interaction ability and help robots understand the surrounding environment as human beings do. On the other hand, the result of semantic segmentation can help to remove additional feature points and improve the accuracy of mapping. As a new development direction, there is a series of semantic SLAM represented by DS-SLAM (Semantic SLAM towards Dynamic Environment) [9]. DS-SLAM introduces SegNet [10] network for semantic segmentation in the front-end visual odometry to solve the problems of location caused by moving objects in the environment. Compared with ORB (Oriented Fast and Rotated Brief)-SLAM2 [11], the accuracy of the localization in dynamic scenes has been improved by an order of magnitude.

However, the addition of a semantic segmentation module increases the computational complexity of the system and cannot be processed on the CPU (Central Processing Unit) in real-time. When paired with the high-performance GPU (Graphics Processing Unit) for computing acceleration, the system consumes a lot of energy and increases the hardware costs, which limits the actual deployment on the terminal devices and mobile robots with limited battery capacity. The FPGA (Field Programmable Gate Array) platform has been widely used in Convolutional Neural Network (CNN) acceleration because it is reconfigurable, energy-efficient, and parallelizable.

For the design of FPGA, traditional development is mainly based on the RTL (Register Transfer Level) [12,13], which has a long development cycle and complicated debugging. This article uses the high-level synthesis tool OpenCL [14] to realize algorithm acceleration. OpenCL works with the host-device model and its code is divided into two parts—the host program and device kernel. The host program is used to obtain device information, set memory objects and execute kernels. In order to control the device, users need to build the host programming code, which meets the OpenCL development specification. The device kernel mainly writes functions that need to be accelerated and these functions are compiled into the accelerator. In the OpenCL development environment, FPGA compiler and synthesis and kernel simulation and debugging are used to get the image for FPGA execution. It can automatically compile from the high-level programming language (C/C++) to the low-level RTL design, provide a faster hardware development cycle and be easily integrated with the applications.

As an industry standard, OpenCL can program heterogeneous computing platforms supported by many suppliers and is not limited by the hardware device. The processing platform used in this article is shown in Figure 1, which is composed of the Turtlebot2 robot, the Kincet camera and the HERO (Heterogeneous Extensible Robot Open) platform. Turtlebot2 is a small, low-cost, fully programmable, ROS (Robot Operating System) based mobile robot with simple operation and strong scalability. HERO is the heterogeneous extensible robot open platform provided by Intel [15]. It has high-quality features such as low power consumption, high performance and small size (18 cm $\times$ 14 cm $\times$ 7.5 cm). The whole of the HERO platform is composed of the host system and the external FPGA acceleration card. In the whole hardware system, CPU has rich IO (Input/Output) functions, provides various high-speed and low-speed interfaces, and its performance is enough to meet that of the simple algorithms. FPGA integrates abundant logic and computing units, storage and routing resources, which ensures the performance acceleration of FPGA in neural network inference. Besides, there is an 8-channel PCIe Gen 3 interface that provides a bandwidth of 7.88 GB/s to perform transmission and communication [15].

The main contributions of this work are summarized below:

1. The quantization strategy of combining convolution and batch normalization into one operation and grouping convolution filters with different data distribution is proposed to solve the problem of the large model and long execution time of a semantic segmentation network.

2. A function configurable pipeline design method and a multi-level memory access optimization scheme for the convolutional Encoder-Decoder architecture of semantic segmentation networks is put forward.
3. The high energy-efficient hardware acceleration solution of the DS-SLAM algorithm on a heterogeneous computing platform is proposed, which can be utilized for the functional module to provide mobile robots with autonomous localization.

Figure 1. The processing platform.

The remainder of this paper is organized as follows—Section 2 provides an overview of the current implementation of different SLAM algorithms on the embedded platform. In Section 3, we give a brief description of DS-SLAM algorithms and the SegNet semantic segmentation network, as well as our proposed quantization strategy. Section 4 presents the framework of our FPGA accelerator and some optimization methods. Subsequently, Section 5 provides the experimental results of the proposed energy-efficient DS-SLAM implementation. Finally, a brief conclusion is provided in Section 6.

2. Related Work

As the SLAM algorithm of mobile robots becomes more and more complex, the general processors on traditional mobile platforms cannot meet the requirements of real-time and energy consumption. The hardware acceleration for the SLAM algorithm has become vital research in the field of robotics.

In [16], an accelerator implemented in FPGA can expand the actual EKF-SLAM (Extended Kalman Filter SLAM) system and observe and correct up to 45 landmarks under the real-time constraints. However, because the complexity of the EKF-SLAM algorithm increases with conversion landmarks, it is not suitable for large-scale environments. Most of the operations are executed sequentially; thus, the parallel architecture's advantages cannot be reflected.

For the LSD-SLAM (Large Scale Direct monocular SLAM) algorithm, Konstantinos Boikos et al. propose a hardware implementation based on an FPGA SoC (System-on-a-Chip) [17]. The design shows that an FPGA SoC can introduce a more intensive SLAM algorithm into embedded low-power devices, and has a higher performance than other solutions. However, the SLAM algorithm implemented in this work is relatively simple, which limits the actual deployment of the system in complex scenarios.

M. Abouzahir et al. use OpenCL to implement the large-scale parallel processing on the FPGA to achieve the Fast-SLAM system [18]. Compared with the GPU-based implementation, the processing time is accelerated by 7.5 times. The research shows that real-time SLAM systems can be implemented on heterogeneous embedded platforms using software and hardware co-design methods.

The eSLAM is proposed by Runze Liu et al. [19]. On a heterogeneous platform based on the ARM (Advanced RISC Machine) processor and FPGA board, a highly energy-efficient ORB-SLAM (Oriented Fast and Rotated Brief SLAM) system is implemented. This work puts the most time-consuming feature extraction and matching process on the FPGA for acceleration, while the remaining threads are executed on the ARM processor. The eSLAM proposes an improved rotationally symmetric BRIEF (Binary Robust Independent Elementary Features) descriptor. By reducing the operation of rotating test locations, the calculation and storage consumption are significantly reduced. Simultaneously, through the pipeline design, the throughput is further improved and the memory is reduced.

It can be seen that the SLAM algorithm implemented on embedded platforms is relatively simple and the related work is still lacking on semantic SLAM hardware implementation. So, this paper proposes a method to achieve the semantic SLAM algorithm based on the HERO platform.

3. Algorithm Implementation

3.1. DS-SLAM Overview

The DS-SLAM algorithm introduces semantic information through deep learning, which endows the robot with learning ability to understand the surrounding environment and reduces the impact of dynamic objects. As can be seen from the algorithm in Figure 2, the DS-SLAM algorithm is a five-thread parallel architecture, including tracking, semantic segmentation, local mapping, loop closure and dense mapping creation. The core idea is to extract ORB feature points from the tracking thread and then check these feature points to get potential dynamic points through a moving mobile consistency check. At the same time, the segmentation results are obtained in parallel with a semantic segmentation thread. Finally, dynamic objects are removed by combining the results of two threads. Table 1 shows the processing time of some major modules in the corresponding thread. The results show that the semantic segmentation thread, whether running on CPU or GPU, is the bottleneck that affects the system's real-time performance. Figure 2 shows the overall framework of the DS-SLAM algorithm based on the HERO platform. Firstly, the images and model parameters are read from CPU. The computationally complex semantic segmentation thread can be accelerated on FPGA with the help of Intel OpenCL SDK (Software Development Kit) and BSP (Board Support Package). Finally, the output is transmitted back to CPU to complete the other threads, enough for real-time processing.

Table 1. Execution time of different modules in the Semantic Simultaneous Localization and Mapping towards Dynamic Environment (DS-SLAM) algorithm [9].

Module	ORB Feature Extraction	Moving Consistency Check	Semantic Segmentation	
Thread	Tracking	Tracking	Semantic segmentation	
Platform	Intel Core i7-6800K CPU	Intel Core i7-6800K CPU	Intel Core i7-8750H CPU	NVIDIA Quadro P4000 GPU
Time (ms)	9.37	29.5	2582	37.6

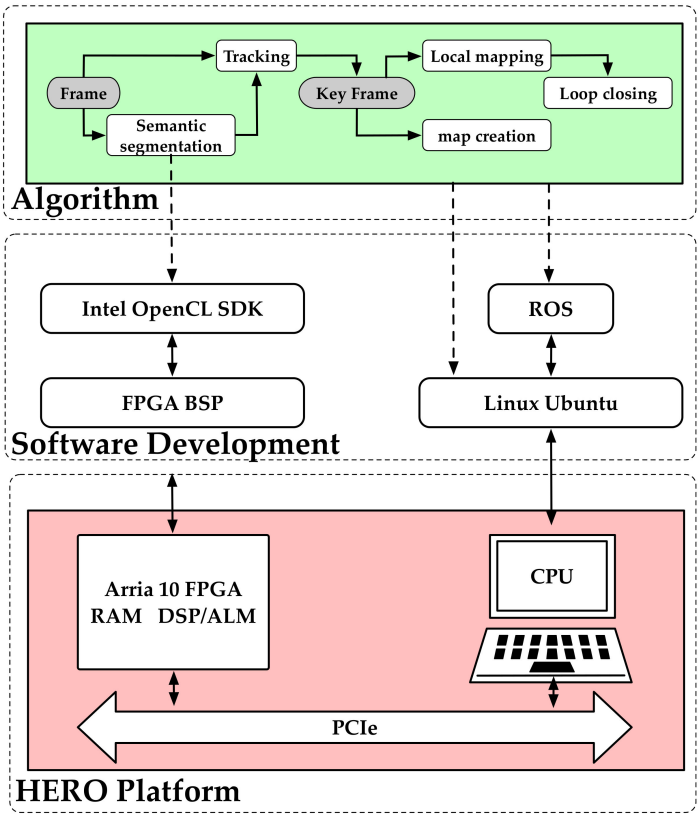

Figure 2. The DS-SLAM framework on Heterogeneous Extensible Robot Open (HERO).

The extraction of semantic information is realized by the semantic segmentation network. In order to reduce the complexity of the whole system and ensure the accuracy and real-time, the SegNet algorithm is selected to achieve semantic segmentation. It is a typical convolutional encoder-decoder architecture.

3.2. Quantization Strategy

The model size of SegNet is 112.3 MB, which should be compressed by pruning, quantization, coding and other means [20] before running on the embedded platform to reduce the computational cost and storage consumption. The method used in this paper is linear quantization, and the overall process is shown in Figure 3.

First, convolution and Batch Normalization (BN) are combined into one operation, because the model parameters are fixed during inference. For SegNet, each convolution layer is followed by a BN layer. Merging these two layers can reduce the use of computing resources and precision loss and improve the inference speed. Next, the data distribution of filters in a convolution layer is calculated. Filters with similar data distribution are merged into one group, corresponding to the same scale factor f. The f determines the decimal point position of the quantized dynamic fixed-point number. Equations (1) and (2) take the weight as an example to introduce the calculation of the f. The w and w' represent the weight before and after quantization, and N represents the bit width of the fixed-point number. f is designed as the integer power of 2, so that the shift can be used instead of the multiplication, which is well implemented on FPGA. Load the preprocessed input image for convolution, and the generated output activations are arranged in the channel dimension. Use the same method described above to calculate the decimal position of the output activation f_o, and the f_o of the current layer is the f_i (decimal position of the input data) of the next layer. To ensure the following correct calculation, a set of 1×1 filter is used to complete the reordering of feature maps. Subsequent pooling and unpooling operations are unchanged. After the final convolution layer, the quantized model and the corresponding scale factor are exported. The whole quantization operation is processed

in the host CPU in advance, and will not increase the inference latency on FPGA. The 8-bit quantized SegNet algorithm is evaluated on the Pascal VOC dataset [21]. The global accuracy is 87.3%, the class accuracy is 69.4%, and the mean IoU is 51.5%, which is 0.8%, 1.1%, and 1.6% lower than the original model. The quantized model dramatically reduces the storage requirements and computational complexity without significant precision loss.

$$f_w = ceil(log_2 \frac{2^N - 1}{max(|w|)}) \tag{1}$$

$$w' = round(\frac{w}{2^{-f_w}}). \tag{2}$$

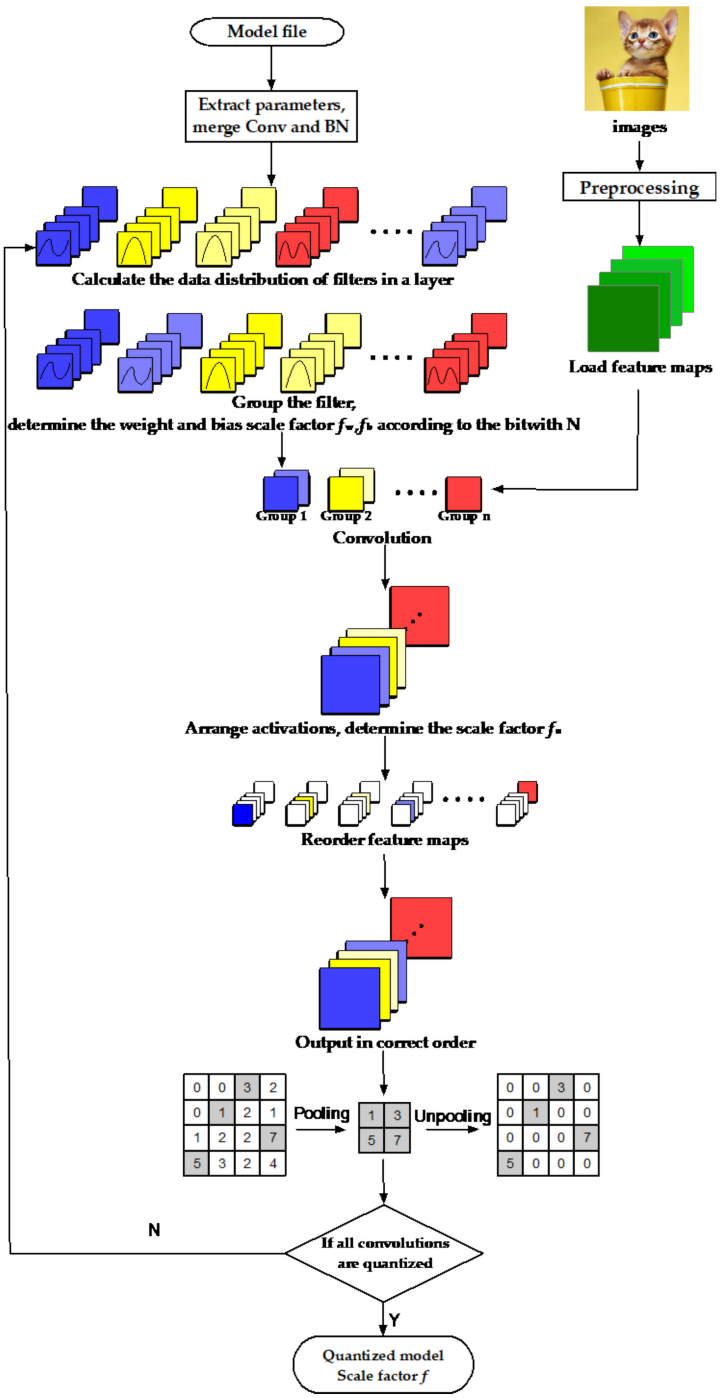

Figure 3. The process of quantization.

4. Proposed Method

The semantic segmentation thread implemented by SegNet is the most time-consuming part of the DS-SLAM algorithm and needs to be accelerated. In [9], SegNet runs on GPU but its high power consumption is not suitable for actual deployment on battery-powered mobile robots. Therefore, this paper will accelerate the inference of semantic segmentation network on FPGA to improve the real-time performance and energy efficiency of the DS-SLAM system.

4.1. Arria 10 FPGA Description

The Arria 10 GX 1150 FPGA board used in this paper adopts TSMC 20 nm technology and has rich design resources. The adaptive logic module (ALM) is the basic component of the FPGA logic structure and 427,200 ALMs can be equivalent to 1,150,000 logic elements (LE) [22], so this FPGA board is named GX 1150. One thousand five hundred and eighteen precision adjustable DSPs (Digital Signal Processings) have encapsulated as hardcore IP (intellectual property) for developers to call, supporting high-performance multiplication and addition operations. Memory can be divided into two types; one is the M20k module, the size of a single module is 20 KB; the other is memory logic array block (MLAB), the size of a single module is 0.64 kb. The M20k module is the best choice for larger memory arrays and provides a large number of independent ports. MLAB is the best choice for the wide and shallow memory array, which can realize shift register and a small FIFO (first in first out) buffer.

4.2. Accelerator Architecture

Through the OpenCL development process, the overall architecture [23,24] of the SegNet accelerator implemented on Arria 10 GX 1150 FPGA is shown in Figure 4. The storage module is mainly divided into two parts. One is off-chip DRAM as the global memory, which is used to store model parameters (weights, bias, and decimal position generated by quantization), pooling indices caused by maximum value pooling and operation data of input and output. Another part is the on-chip local buffer as the local memory, which is used to store the input feature map and weights of the current calculation layer, as well as the output activation value. On-chip memory can be further subdivided into channels and registers, which, together with off-chip DRAM, form a multi-level memory access structure. The channel extension defined by OpenCL is widely used between local buffer and kernel, as well as between kernel and kernel. It acts like FIFO and allows the transfers on chip to reduce latency. The dataflow determines what data is read into which level of memory structure and when it is processed. Compared with on-chip memory, DRAM consumes more energy per access. Therefore, we should increase the data reuse of on-chip memory to reduce access to high-cost memory. In this paper, each weight is read from DRAM to local buffer and stays stationary for data reused and the input feature maps are broadcast to different CUs, then the partial sums are accumulated. Weight reuse minimizes the energy consumption of reading weights. Currently, the DRAM access bandwidth is 1234 MB/s.

OpenCL uses a host-device model, and the design is divided into two parts—the host program and the device kernel. The host program includes an API (Application Programming Interface) to define and control the platform, which is used to obtain device information. Besides, the host program operates memory objects and creates a command queue and executes kernel queues. The design of the device kernel is based on the standard C language with some restrictions and extensions. The kernel program mainly includes the function that needs to be accelerated and these functions are compiled into the accelerator. For the SegNet network, this paper mainly designs convolution kernel, pooling kernel and unpooling kernel for SegNet, and the ReLU activation function is integrated into the convolution kernel. Through a 2-bit control flag, the different combinations of kernels realize a function configurable pipeline. There are four modes in total, shown in Figure 5.

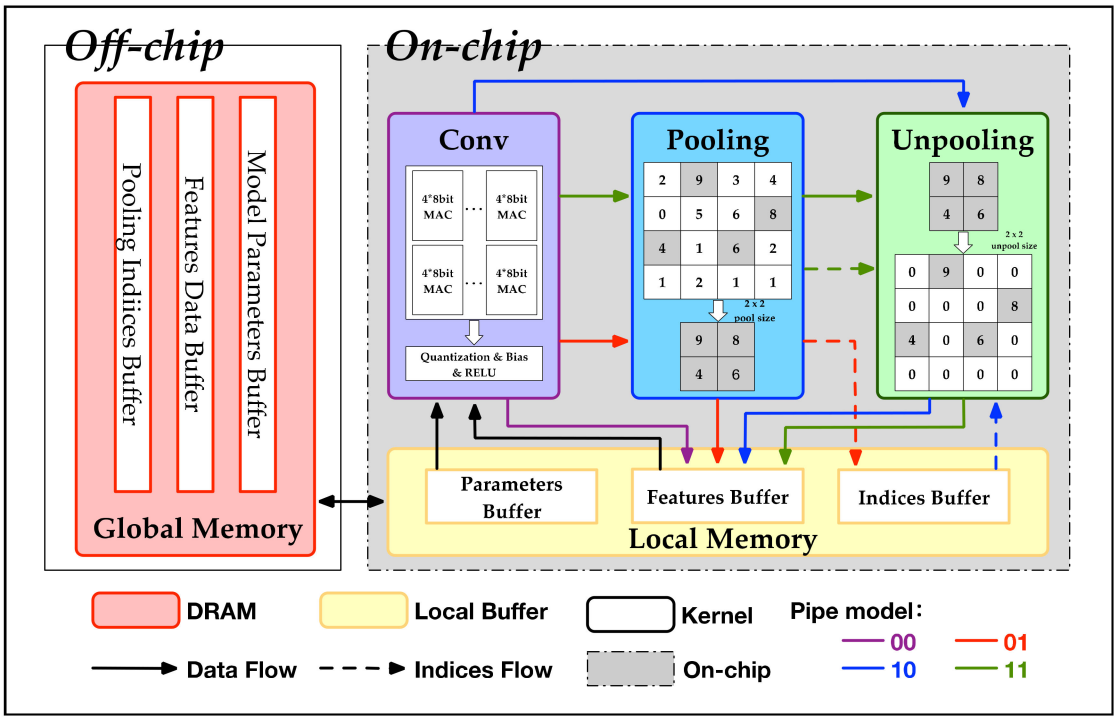

Figure 4. The architecture of the SegNet accelerator.

The 00 mode pipeline only executes convolution operation, shown in Figure 5a. The weight and decimal point of one layer are read from DRAM to the local buffer at one time and then they are pressed into the channel continuously and transferred to the convolution kernel for operation. However, the input feature map of the convolution layer is read in batches. The on-chip read buffer improves data transmission efficiency by Ping-Pong operation. Two buffers of Ping-Pong operation are designed on the chip. When one buffer transmits characteristics, the other buffer reads the input characteristic group of the next stage from the off-chip DRAM at the same time, so as to improve the throughput. Convolution operations have different computing units to execute in parallel. The result is transferred back to off-chip DRAM, through the quantification and ReLU (Rectified Linear Unit). Since there is no dependency between the result of off-chip DRAM, multiple work items are executed in parallel to write the result back.

In the 01 mode pipeline, convolution and pooling operations are executed in parallel. Part of the convolution results are passed to the pooling kernel through the channel, avoiding the data transmission delay caused by multiple access to DRAM to maintain a high-speed processing efficiency. The pooling results and pooling indices are passed to the local buffer through the channel and finally written back to the off-chip DRAM. The input and output memory are used alternately and the output memory of the current operation will become the input memory of the next operation. The data flow is shown in Figure 5b.

The 10 mode pipeline is a combination of convolution and unpooling, shown in Figure 5c. For the unpooling operation, the input not only has the calculation results of the previous kernel but also has the pooling indices with location information. Pooling indices need to be read from the off-chip DRAM to on-chip local buffer, which will cause unnecessary delay. Therefore, we prepare the required pooling indices in advance through data pre-reading. When the convolution results are transmitted to the unpooling kernel with the help of the channel, the operation can be executed directly without waiting time. This design method is beneficial for the encoder-decoder network, with a connection between the front and back layers. As shown in Figure 5d, the 11 mode pipeline performs all of the kernels, including convolution, pooling, and unpooling operations. The pooling indices required by unpooling are directly obtained from the last kernel through the channel without accessing off-chip DRAM.

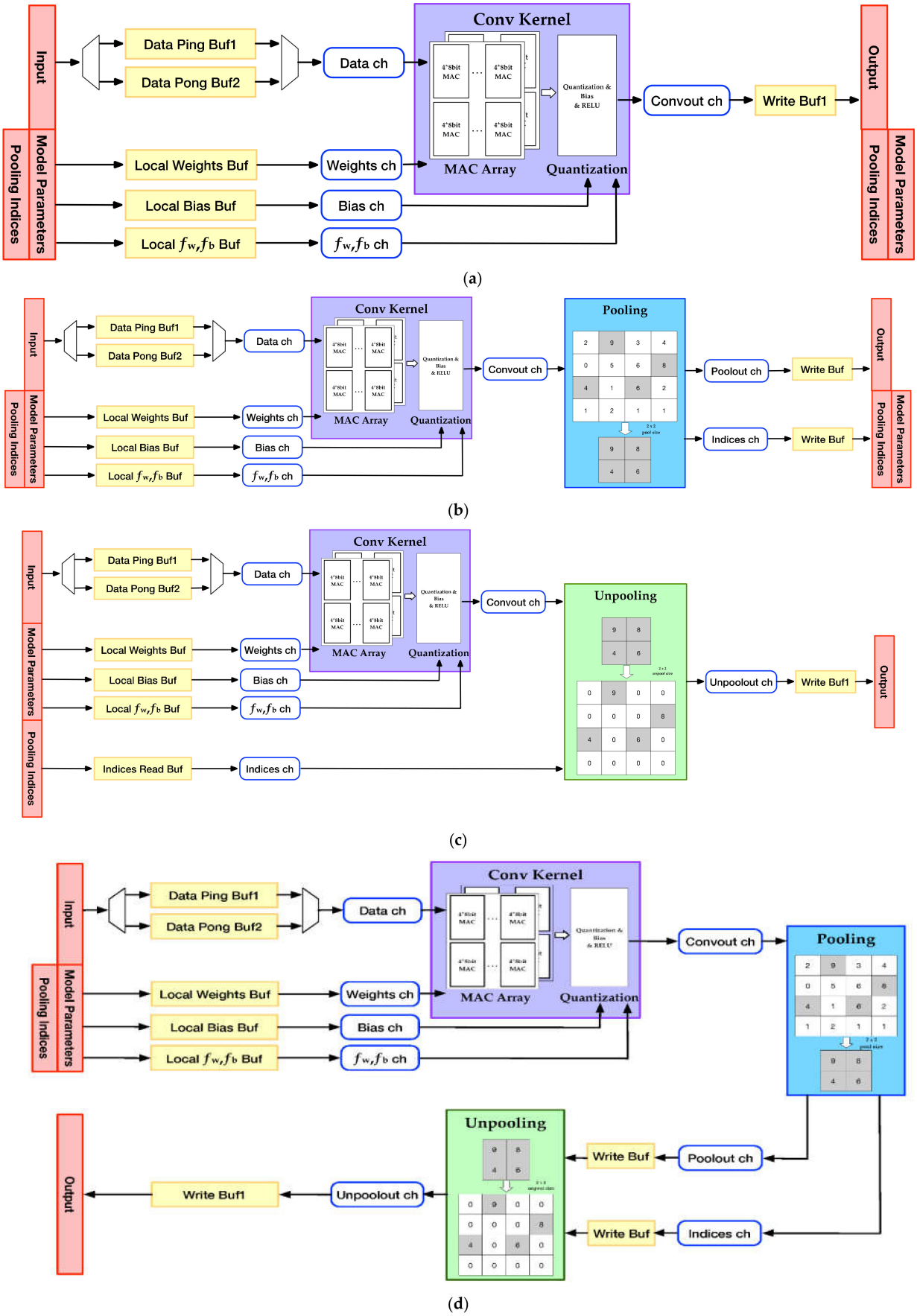

Figure 5. Four pipeline modes with configurable functions. (**a**) Mode 00: convolution; (**b**) Mode 01: convolution + pooling; (**c**) Mode 10: convolution + unpooling; (**d**) Mode 11: convolution + pooling + unpooling.

Each pipeline mode completes an iteration of "data reading in, computing, and writing back." Then multiple iterations are combined in an orderly way to realize the whole network. For example, to implement SegNet, the combination is in sequence of "00-01-00-01-00-00-01-00-00-01-00-00-11-00-00-10-00-00-10-00-00-10-00-10-00-00". The benefit of this design approach is that using multiple pipeline loops in different patterns allows us to implement different network frameworks flexibly. This kind of function configurable pipeline architecture has a certain expansibility. By designing and replacing different kernels, we can flexibly implement other convolutional neural networks. In the process of implementing other neural networks, we should evaluate the boards' resources in advance and focus on the use of local memory. We must pay attention to the kernel design and follow the coding rules of OpenCL, so as to improve the efficiency of the hardware implementation.

4.3. Softmax Optimization

In addition to the inference part of FPGA, the result of segmentation also needs to be accelerated on CPU in the implementation of the semantic segmentation thread. Softmax operation, as the last layer of the SegNet network, is used to realize multi classification. It maps the output of multiple neurons to the (0~1) range. The sum of the mapped values is 1, which can be regarded as the probability value. In the calculation of softmax, the maximum probability is taken as the final prediction result, as shown in Figure 6. This operation only normalizes the result and does not change the numerical relationship, so the operation can be omitted. Softmax is relatively complex because of its exponential operation, which affects the real-time performance of the system. After removing softmax, the speed of semantic segmentation is improved by nearly 77 ms.

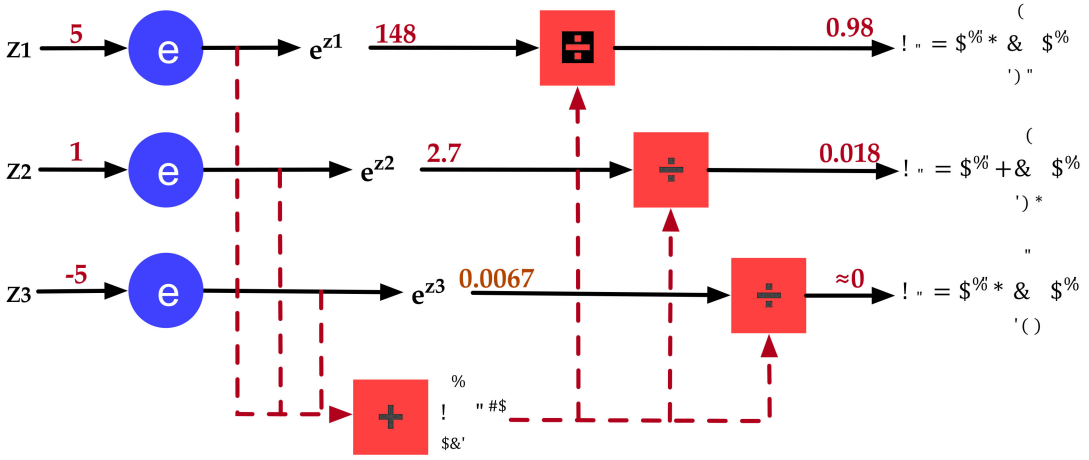

Figure 6. Softmax operation diagram.

5. Experimental Results

5.1. HERO Platform Description

The proposed DS-SLAM accelerator is implemented on the HERO platform, which contains an Intel i5-7260U CPU as a host side and an Arria 10 GX 1150 FPGA as an acceleration board. The semantic segmentation thread that needs to be accelerated is deployed on FPGA, and the others run on CPU. We use the OpenCL development process to implement the SegNet network through the Intel FPGA SDK. The clock frequency of the FPGA is 202 MHz and the hardware resource utilization is shown in Table 2. It can be seen that the utilization rate of DSP reaches 100%, almost all of which is used to realize convolution operation.

Table 2. Hardware resource utilization of Arria 10 Field Programmable Gate Array (FPGA).

	ALMS	Registers	M20Ks	DSPs
Utilization	24%	12%	63%	100%

The energy-efficient DS-SLAM system implemented on a heterogeneous computing platform is evaluated on the TUM RGB-D dataset [25]. The TUM dataset is mainly for office scenes. Its "walking" sequence contains dynamic objects, which are two moving people.

5.2. Performance Evaluation

In Table 3, the execution time of semantic segmentation thread and the whole DS-SLAM algorithm, frame rate, the power consumption and the energy efficiency of the proposed method in this paper are compared with the software implementations on the Intel i7-8750H CPU and the NVIDIA P4000 GPU. In contrast with CPU, our acceleration platform achieves up to 13× frame rate improvement and up to 18× energy efficiency improvement. The GPU server has more computing resources, thus achieves a higher real-time performance than ours. An important application of mobile robots is to provide indoor services. In order to ensure human safety, the moving speed is relatively slow, so the real-time performance of 5.3 fps can meet the application requirements. In [9], the experimental platform is Intel i7-6800K CPU with NVIDIA P4000 GPU and the overall power consumption is 245 W, which is unacceptable for energy-limited applications. Mobile robots are usually powered by batteries, so it is essential to reduce the power consumption of the processor. The HERO is a low power platform, and the proposed accelerator implemented on it is 2.2 times more energy-efficient than GPU.

Table 3. Execution time, frame rate, power consumption, and energy efficiency comparison results.

	Software Implementations		**Our Work**	**Improvements (vs. CPU)**
Platform	i7-8750H CPU	i7-6800K CPU + P4000 GPU	HERO	-
Power	45W	245 W	35 W	22.2%
Execution time of semantic segmentation	2582 ms	37.6 ms	155 ms	94.0%
Execution time of the whole DS-SLAM	2600 ms	59.4 ms	190 ms	92.7%
Frame rate	0.38 fps	16.8 fps	5.3 fps	13×
Energy efficiency	0.008 fps/W	0.068 fps/W	0.151 fps/W	18×

The proposed high energy efficiency accelerator provides the feasibility of deploying the DS-SLAM algorithm on mobile robots. The SegNet network, which is used by the semantic segmentation thread, can be used for 21 kinds of objects including indoor and outdoor scenes, and well supports the semantic segmentation of the datasets used in the SLAM algorithm. The evaluation results on the TUM dataset are shown in Figure 7 and the objects are well segmented. As the dynamic object, people will be eliminated in the final mapping to improve the localization accuracy.

Figure 7. The result of semantic segmentation on the TUM dataset.

5.3. Accuracy Analysis

The TUM dataset provides the ground truth of the real trajectory for the convenience of comparison with the experimental results. The indicators to evaluate the accuracy of DS-SLAM implemented in this paper are ATE (Absolute Trajectory Error) and RPE (Relative Pose Error). ATE is the direct difference between estimated pose and real pose, which can directly reflect the accuracy of the algorithm and the global consistency of the trajectory. RPE is used to evaluate the drift of the system. In this paper, we present the values of Root Mean Squared Error (RMSE), Mean Error, Median Error, and Standard Deviation (S.D.). The experimental results are shown in Tables 4 and 5, comparing the accuracy of ORB-SLAM2, DS-SLAM software algorithm and hardware implementation DS-SLAM proposed in this paper under different data sequences. Figure 8 shows ATE and RPE plots from ORB-SLAM2 and hardware implementation DS-SLAM in high dynamic sequence (fr3_walking_xyz) respectively. The ATE results show that black is the true value of the trajectory, blue is the estimated trajectory, and red is the error between the two.

Table 4. The results of metric absolute trajectory error (ATE).

Sequences	ORB-SLAM2				DS-SLAM				Our Work			
	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.
Fr3_walking_xyz	0.7521	0.6492	0.5857	0.3759	0.0247	0.0186	0.0151	0.0161	0.0316	0.0176	0.0129	0.0262
Fr3_walking_static	0.3900	0.3554	0.3087	0.1602	0.0081	0.0073	0.0067	0.0036	0.0078	0.0069	0.0063	0.0035
Fr3_walking_rpy	0.8705	0.7425	0.7059	0.4520	0.4442	0.3768	0.2835	0.2350	0.5031	0.4276	0.3066	0.2646
Fr3_walking_half	0.4863	0.4272	0.3964	0.2290	0.0303	0.0258	0.0222	0.0159	0.0292	0.0250	0.0214	0.0151
Fr3_sitting_static	0.0087	0.0076	0.0066	0.0043	0.0065	0.0055	0.0049	0.0033	0.0060	0.0052	0.0046	0.0030

Table 5. The results of metric translational drift (RPE).

Sequences	ORB-SLAM2				DS-SLAM				Our Work			
	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.
Fr3_walking_xyz	0.4124	0.3110	0.2465	0.2684	0.0333	0.0238	0.0181	0.0229	0.0459	0.0275	0.0198	0.0367
Fr3_walking_static	0.2162	0.0905	0.0155	0.1962	0.0102	0.0091	0.0082	0.0048	0.0115	0.0104	0.0099	0.0048
Fr3_walking_rpy	0.4249	0.2825	0.1487	0.3166	0.1503	0.0942	0.0457	0.1168	0.1903	0.1076	0.0436	0.1570
Fr3_walking_half	0.3550	0.2161	0.0774	0.2810	0.0297	0.0256	0.0226	0.0152	0.0414	0.0363	0.0335	0.0198
Fr3_sitting_static	0.0095	0.0083	0.0073	0.0046	0.0078	0.0068	0.0061	0.0038	0.0089	0.0079	0.0071	0.0042

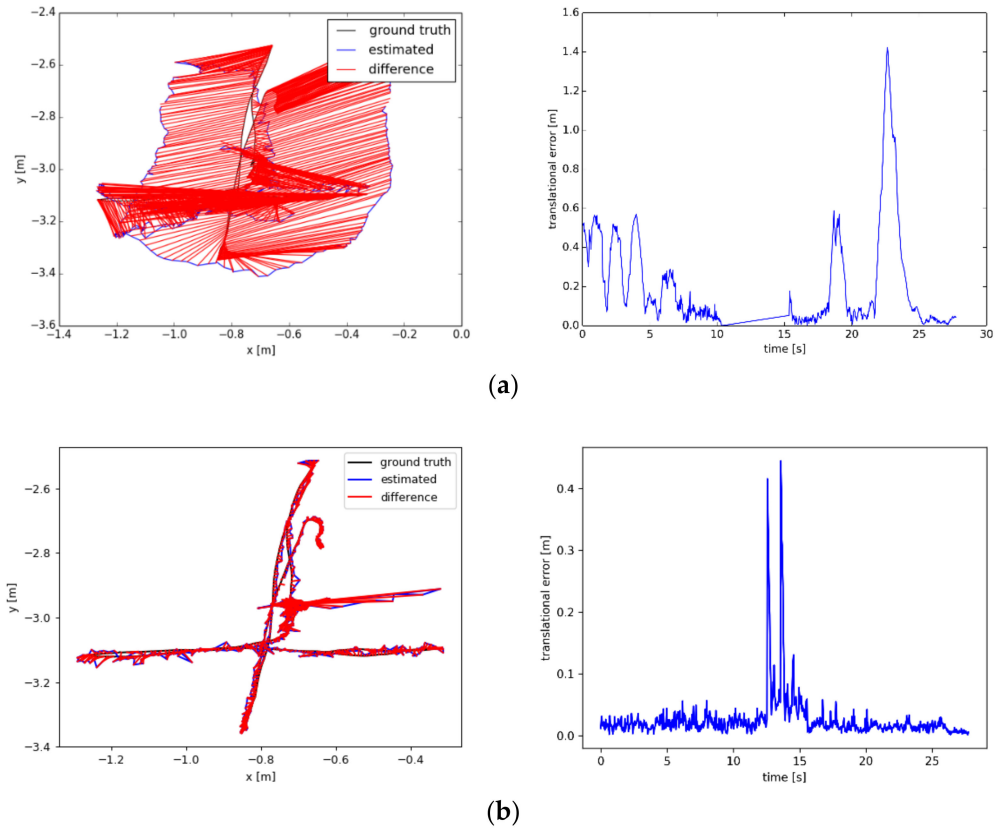

Figure 8. Visual experimental results: (**a**) ATE and RPE from ORB-SLAM2, (**b**) ATE and RPE from our work.

Compared with ORB-SLAM2, the introduction of semantic information in DS-SLAM can eliminate the interference of moving objects and significantly improve the accuracy in thr dynamic environment. The accuracy of our work is in the same order of magnitude as DS-SLAM, but it is slightly reduced because of model quantization. We have made a trade-off between accuracy and energy consumption, sacrificing a little precision to improve the energy efficiency of the system, providing a solution for the actual deployment of DS-SLAM on mobile robots.

6. Conclusions

In this paper, an energy-efficient DS-SLAM based semantic SLAM system is proposed on the HERO heterogeneous platform. The model of SegNet for extracting semantic information is first quantized to the 8-bits dynamic fixed-point number. The proposed quantization strategy reduces the storage requirements and computational complexity, and only loses 0.8% of the global accuracy on the Pascal VOC dataset. Meanwhile, as the most time-consuming part, the semantic segmentation thread is accelerated on the FPGA to reduce the latency. A function configurable pipeline architecture with expansibility is proposed. By designing and replacing kernels, other convolutional neural networks can be implemented flexibly. Compared with Intel i7 CPU, our DS-SLAM accelerator on HERO platform achieves up to 13× frame rate improvement and 18× energy efficiency improvement. Moreover, there is a minute localization accuracy loss on the TUM dataset. The proposed energy- efficient DS-SLAM accelerator fundamentally solves the dependence of the algorithm on high energy consumption GPU. It increases the working time of battery-powered robots, which can be utilized as a functional module to provide mobile robots with autonomous localization.

Author Contributions: Conceptualization, Y.W., M.Y., H.H. and F.Q.; data curation, Y.W., H.H. and M.Y.; formal analysis, Y.W.; funding acquisition, F.Q., Q.W. and X.L.; investigation, Y.W. and M.Y.; methodology, Y.W., M.Y., H.H. and X.S.; project administration, F.Q., S.Y. and L.L.; resources, X.S., F.Q., Q.W. and X.L.; software, M.Y. and X.S.; supervision, F.Q., S.Y. and L.L.; validation, Y.W. and X.S.; visualization, Y.W.; writing—original draft, Y.W.; writing—review & editing: M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key R&D Program of China under grant No. 2018YFB1702500.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors would like to acknowledge supports from National Key R&D Program of China under grant No. 2018YFB1702500. The authors would also acknowledge support from Beijing Innovation Center for Future Chips, Tsinghua University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Song, S.; Choi, D.; Hur, J.; Lee, M.; Park, Y.J.; Kim, J. Development and application of Mobile Robot system for Marking Process in LNGC cargo tanks. In Proceedings of the 2009 ICCAS-SICE, Fukuoka, Japan, 18–21 August 2009; pp. 2636–2638.
2. Saitoh, M.; Takahashi, Y.; Sankaranarayanan, A.; Ohmachi, H.; Marukawa, K. A mobile robot testbed with manipulator for security guard application. In Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, 21–27 May 2002; Volume 3, pp. 2518–2523.
3. Murphy, R. Human–Robot Interaction in Rescue Robotics. *IEEE Trans. Syst. Man. Cybern. Part C Appl. Rev.* **2004**, *34*, 138–153. [CrossRef]
4. Dunbabin, M.; Marques, L. Robots for Environmental Monitoring: Significant Advancements and Applications. *IEEE Robot. Autom. Mag.* **2012**, *19*, 24–39. [CrossRef]
5. Bapna, D.; Rollins, E.; Murphy, J.; Maimone, E.; Whittaker, W.; Wettergreen, D. The Atacama Desert Trek: Outcomes. In Proceedings of the 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Leuven, Belgium, 20 May 2002; Volume 1, pp. 597–604.
6. Rollins, E.; Luntz, J.; Foessel, A.; Shamah, B.; Whittaker, W. Nomad: A Demonstration of the Transforming Chassis. In Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, 20 May 1998; pp. 611–617.
7. Kimon, P.V. Classification of UAVs. In *Handbook of Unmanned Aerial Vehicles*; Kimon, P.V., George, J.V., Eds.; Springer: Dordrecht, The Netherlands, 2015.
8. Mei, Y.; Lu, Y.H.; Hu, Y.C. A case study of mobile robot's energy consumption and conservation techniques. In Proceedings of the International Conference on Advanced Robotics, Seattle, WA, USA, 18–20 July 2005.
9. Yu, C.; Liu, Z.; Liu, X. Ds-slam: A semantic visual slam towards dynamic environments. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1168–1174.
10. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
11. Mur-Artal, R.; Tardos, J.D. Orb-slam2: An open-source slam system for monocular, stereo and rgb-d cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
12. Zhang, C.; Li, P.; Sun, G. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2015; pp. 161–170.
13. Qiu, J.; Wang, J.; Yao, S. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 21–23 February 2016; pp. 26–35.
14. Khronos Group. OpenCL-The Open Standard for Parallel Programming of Heterogeneous Systems. Available online: http://www.khronos.org/opencl (accessed on 10 December 2020).
15. Shi, X.; Cao, L.; Wang, D. HERO: Accelerating Autonomous Robotic Tasks with FPGA. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
16. Tertei, D.T.; Piat, J.; Devy, M. FPGA design of EKF block accelerator for 3D visual SLAM. *Comput. Electr. Eng.* **2016**, *55*, 123–137. [CrossRef]
17. Boikos, K.; Bouganis, C.-S. Semi-dense SLAM on an FPGA SoC. In Proceedings of the 2016 26th International Conference on Field Programmable Logic and Applications (FPL), Lausanne, Switzerland, 29 August–2 September 2016; pp. 1–4.

18. Abouzahir, M.; Elouardi, A.; Latif, R.; Bouaziz, S.; Tajer, A. Embedding SLAM algorithms: Has it come of age? *Robot. Auton. Syst.* **2018**, *100*, 14–26. [CrossRef]
19. Liu, R.; Yang, J. Eslam: An energy-efficient accelerator for realtime orb-slam on fpga platform. In *DAC*; ACM: New York, NY, USA, 2019; p. 193.
20. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
21. Everingham, M.; Eslami SM, A.; Van Gool, L. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
22. Intel Corporation. Intel Arria 10 Device Overview. Available online: https://www.intel.com/content/www/us/en/programmable/documentation/sam1403480274650.html (accessed on 10 December 2020).
23. Yu, M.; Huang, H.; Liu, H.; He, S.; Qiao, F.; Luo, L.; Xie, F.; Liu, X.-J.; Yang, H. Optimizing FPGA-based Convolutional Encoder-Decoder Architecture for Semantic Segmentation. In Proceedings of the 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Guangzhou, China, 29 July–2 August 2019; pp. 1436–1440.
24. Huang, H. EDSSA: An Encoder-Decoder Semantic Segmentation Networks Accelerator on OpenCL-Based FPGA Platform. *Sensors* **2020**, *20*, 3969. [CrossRef] [PubMed]
25. Sturm, J.; Engelhard, N.; Endres, F. A benchmark for the evaluation of rgb-d slam systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 573–580.