INCA: INterruptible CNN Accelerator for Multi-tasking in Robots

1st Anonymous for Review

***Abstract*—In recent years, Convolutional Neural Network (CNN) has been widely used in robotics, which has dramatically improved the perception and decision-making ability of robots. In order to implement energy-efficient CNN on embedded systems, a series of CNN accelerators have been designed. However, despite the high energy efficiency on CNN accelerators, it is difficult for robotics developers to use it. Since the various functions on the robot are usually implemented independently by different developers, simultaneous access to the CNN accelerator by these multiple independent processes will result in hardware resource conflicts.**

To handle the above problem, we propose an INterruptible CNN Accelerator (INCA) to enable multi-tasking on CNN accelerators. In INCA, we propose a Virtual-Instruction-based interrupt method (VI method) to support multi-task on CNN accelerators. Based on INCA, we deploy the Distributed Simultaneously Localization and Mapping (DSLAM) on an embedded FPGA platform. We use CNN to implement two key components in DSLAM, Feature-point Extraction (FE) and Place Recognition (PR), so that they can both be accelerated on the same CNN accelerator. Experimental results show that INCA enables multi-task scheduling on the CNN accelerator with negligible performance degradation (within 0.3%). Compared to the layer-by-layer interrupt method, our VI method reduces the interrupt responding latency to 2%.

I. Introduction

With the development of algorithms and hardware platforms, Convolutional Neural Network (CNN) has greatly improved the perception and decision-making ability of unmanned platform, such as object detection [1] and scene segmentation [2] in perception, and path planning [3] and dynamic obstacle avoidance [4] in decision-making.

Distributed Simultaneously Localization and Mapping (DSLAM) is a basic task for many multi-robot applications, and is a hot topic in robotics. There are two key modules which consume most of the computation: Feature-point Extraction (FE) and Place Recognition (PR). FE provides the feature-points for the Visual Odometry (VO) to calculate the relative pose between two adjacent frames. PR generates the compact image representation, which produces the candidate place recognition matches between different robots. Recent works use CNN to extract feature-points [5]–[7] and generate the place representation code [8], [9]. The CNN-based feature-point extraction method, SuperPoint [5], achieves 10%-30% higher matching accuracy compared with the popular handcrafted extraction method, ORB [10]. The accuracy of the place recognition code from another CNN-based method, GeM [9], is also about 20% better than the handcrafted method, rootSIFT [11].

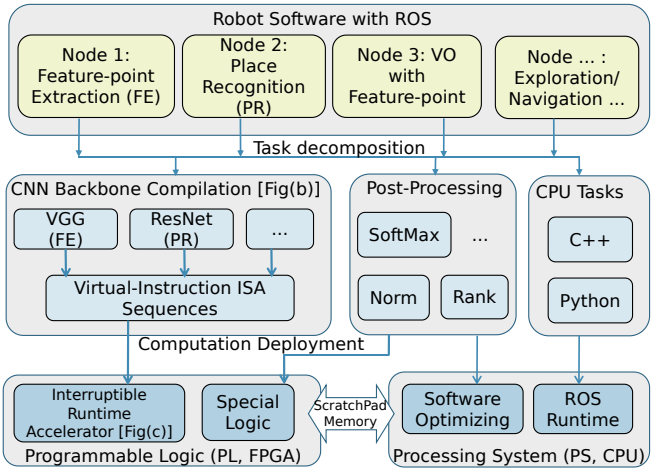


Fig. 1. INCA framework. At task decomposition step, the operations in different ROS nodes are separated to CNN backbones, CNN Post-Processing, and Other CPU Tasks. At computation deployment, the CNN backbone and Post-Processing are deployed with hardware modules and software optimizing.

However, CNN is computation consuming. A single inference forward of the CNN-based SuperPoint feature-point extraction consumes 39G operations [5], and a single inference forward of the CNN-based GeM place recognition consumes 192G operations [9]. Thus, specific hardware architectures on FPGA [12]–[16] are designed to deploy CNN on the embedded system. With the help of network quantization and on-chip data reuse, the speed of CNN accelerators on embedded FPGA achieves 3TOP/s [16], which can support the real-time execution of CNN-based feature-point extraction [5]. However, these CNN accelerators are designed and optimized to accelerate a single CNN. They can not automatically schedule two or more tasks simultaneously.

In order to facilitate robotic researchers to run different CNN tasks simultaneously on the FPGA accelerator, the accelerator should support the following features:

Multi-thread: Because different components in a robot are from different developers, thus, Robot Operating System (ROS) [17] is proposed as a middleware to fuse these independent components, and is widely used by robotic researchers. Each component is considered as an independent thread in ROS. Different threads should have independent access to the accelerator without knowing the status of others.

Finishing before deadline: In a robot, some tasks must be completed within the specified hard deadlines, such as feature-point extraction. The moving robot's perception, including

estimation of itself's location and the obstacles' position, is based on the feature-points. If the feature point extraction is not completed before the deadline, the robot can not estimate the surrounding environment, causing collisions or even damage. Those critical tasks with a more stringent headline need to be performed prior to some non-critical tasks [18]. In DSLAM, the priority of feature-point extraction (FE) is higher than that of place recognition (PR). Because PR is only related to efficiency, yet FE ensures system safety.

To address above challenges, we propose an INterruptible CNN Accelerator (INCA) for rapid deployment of robot application on FPGA. The work flow of INCA is illustrated in Figure 1. INCA is a two-step framework for mapping software to embedded FPGA. The first step is the task decomposition, which decomposes the computation in ROS nodes into different computation types, including CNN backbones, CNN post-processing, and other CPU tasks. The second step is to deploy the computation onto the FPGA. The CNN backbones of different tasks, such as the VGG model [19] in SupoerPoint feature-point extraction [5] and the ResNet101 model [20] in GeM place recognition [9], are compiled to the interruptible Virtual-Instruction Instruction Set Architecture (VI-ISA), which runs on the CNN accelerator. The VI-ISA is a simple extension of the original ISA, in which the extension method is not limited to a specific original ISA. Thus, the virtual-instruction-based interrupt can be easily applied to various instruction-based CNN accelerators [13], [15], such as Angel-Eye [12] and DPU [21].

In conclusion, INCA facilitate robotic researchers to run different CNN tasks simultaneously on the FPGA with the following contributions:

- We propose a **virtual-instruction-based** interrupt method to make the CNN accelerator support dynamic multi-task scheduling by priority. The method solves the hardware resources conflicts when accelerating different CNN tasks on ROS [17].
- We propose a CNN-based DSLAM system to evaluate INCA. CNN-based methods for feature-point extraction (FE) and place recognition (PR) are accelerated with FPGA on ROS platform. With the help of the unified interface in ROS, these CNN-based methods can be easily used by other developers in different applications.

II. Related Work

To accelerate CNN, some previous works design frameworks to generate a specific hardware architecture for a target CNN, based on RTL [14] or HLS [16]. These works need to reconfigure the FPGA to switch between different CNN models. The reconfiguration consumes seconds [22], which is unacceptable for the real-time system. Some other works design instruction-driven accelerators [12], [13], [15], [21], making rapid switching possible by providing different instruction sequences. However, the CNN tasks on previous instruction-driven CNN accelerators are not interruptible, resulting in the latency-sensitive high-priority task waiting for the low-priority task to finish. This inability of CNN accelerators to support multi-task makes it difficult for robotic researchers to use embedded FPGA.

III. INCA Frame work

A. Hardware Resources Conflicts

1) Hardware Resources Conflicts In ROS: Although ROS is becoming the fundamental software platform for robotics, the independence between different ROS nodes brings **hardware resources conflicts challenge** to access the hardware accelerator. Figure 2 shows the time diagram of scheduling feature-point based visual odometry (VO) and Place Recognition in DSLAM system. The feature-point extraction (FE) and Place Recognition (PR) are impelmented in CNN and deployed to the CNN accelerator. In the native accelerator (the shadow part in Figure 2), the threads of FE and PR may need to process CNN at thesame time, and the simultaneous requests of the acceleratorwill lead to hardware resources conflicts.

Figure 2 also illustrates the idea of interrupt to schedule two CNN tasks. In the process of running a low-priority network (PR), the software may send an execution request for the high-priority task (FE). The interrupt enables the CNN accelerator to backup the running state of the low-priority PR network. Then the accelerator switches to the high-priority FE network. After the high-priority task (FE) completes, the low-priority task (PR) is restored to the accelerator and continues to execute.

B. Interruptible Accelerator Framework (INCA Framework)

Figure 3(a) details the INCA compilation step and runtime interrupt. Caffe [23] is a popular software framework for CNN, and the *.caffemodel/*.prototxt files define the network parameters and structure in Caffe. The previous deployment process, such as Angel-Eye [12] and DPU [21], quantizes the weights, and analyze the network topology. The original compiler translates the network topology and the quantization information into the original ISA sequence. INCA goes further than previous CNN compilers. It selects the optimized interrupt positions in the original instruction sequence, and adds virtual instructions at these positions to enable accelerator interrupt. After that, the original instruction sequence and the added virtual instructions are wrapped to the new interruptible VI-ISA. The wrapped VI-ISA instructions are dumped into a file (instruction.bin), and can be loaded into the instruction spaces on FPGA's DDR.

As illustrated in Figure 3(b), at runtime, an Instruction Arrangement Unit (IAU) in hardware listens to the interrupt request from ROS software, fetches the corresponding VI-ISA interruptible instructions and translates them to the original ISA executed on the CNN accelerator. Although INCA can be applied to various instruction-based CNN accelerators, we implement and evaluate it based on Angel-Eye [12].

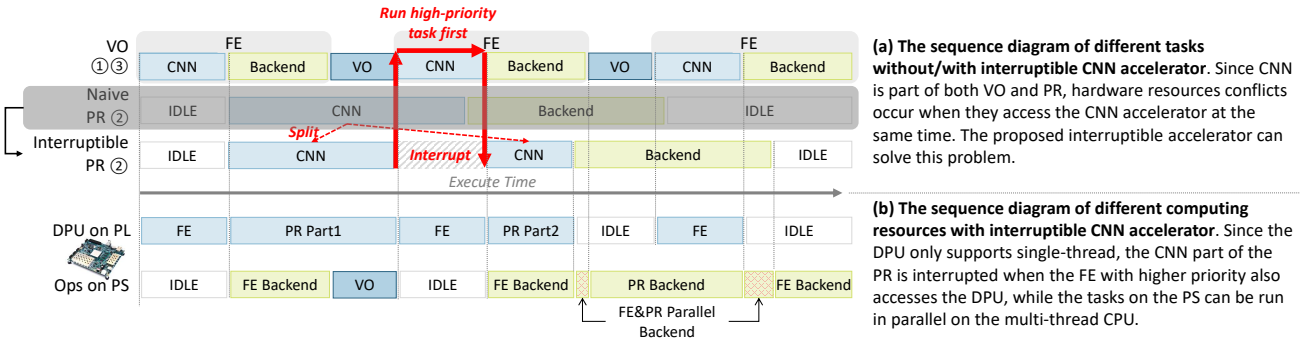
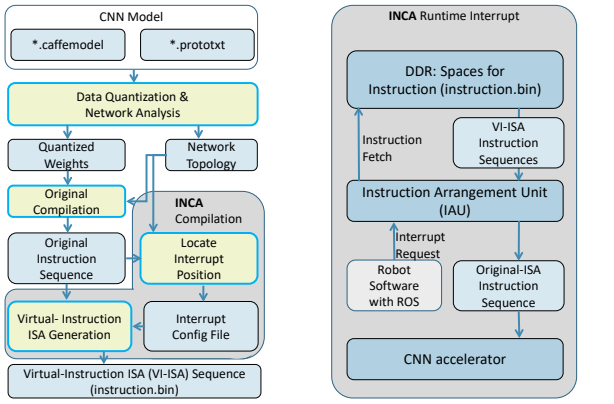


Fig. 2. Interruption to solve the hardware resources conflicts.

TABLE I
DESCRIPTION FOR THE BASIC INSTRUCTIONS

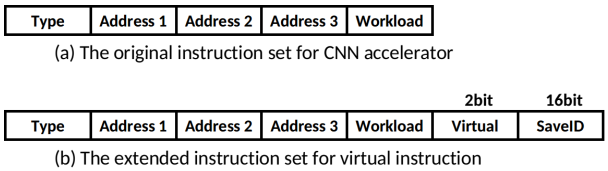
Category	Type	Description	Address 1	Address 2	Address 3	Workload	Backups	Recovery
LOAD	LOAD_W	Load weights/bias from DDR to on chip weight buffer.	Off-chip Addr	Weights-buffer Addr	-	Data Length	-	Weight / Inputdata
	LOAD_D	Load input data from DDR to on-chip data buffer.	Off-chip Addr	Data-buffer Addr	-	Data Length	-	Weight / Inputdata
CALC	CALC_I	Calculate intermediate results for some output channels from partial input channels.	Input Data Addr	Intermediate Data Addr	Weight Addr	Calc Size	Previous final results / Intemediate data	Weight / Inputdata / intemediate data
	CALC_F	Calculate the results for some output channels from all input channels.	Input Data Addr	Output Data Addr	Weight Addr	Calc Size	Finial results	Weight / Inputdata
SAVE	SAVE	Save the results from on-chip data buffer to DDR.	Off-chip Addr	Data-buffer Addr	-	Data Length	-	Weight / Inputdata



(a) At **compilation** step, INCA locates the interrupt location, adds virtual instructions, and generates the virtual-instruction ISA (VI-ISA) sequence.

(b) At **runtime**, IAU translates the VI-ISA sequence to original ISA sequence executed on the accelerator.

Fig. 3. INCA framework uses Virtual-Instruction (VI) to enable accelerator interrupt.



(a) The original instruction set for CNN accelerator

(b) The extended instruction set for virtual instruction

Fig. 4. Original and Virtual-Instruction ISA.

IV. Virtual-Instruction-based Accelerator Interrupt

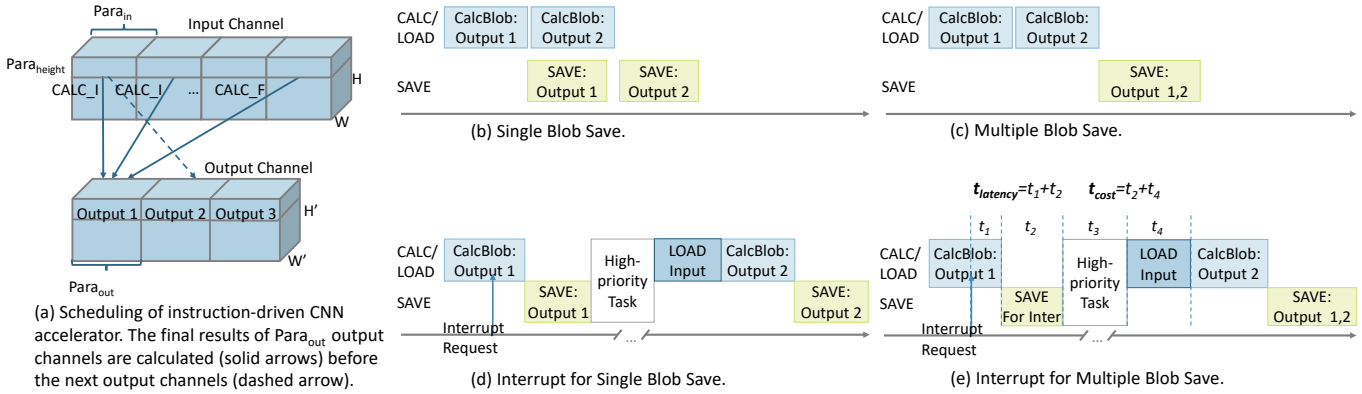
A. Instruction Driven Accelerator

There are three categories of instruction in the instruction-driven accelerator: LOAD, CALC, and SAVE [12], [13], [15]. The instruction description of each kind of instruction is listed in Figure 4(a) and Table I.

Each CALC instruction, including CALC_I and CALC_F, processes the convolution according to the hardware parallelism with $Para_{height}$ lines from $Para_{in}$ input channels to $Para_{out}$ output channels. $Para_{height}$, $Para_{in}$, and $Para_{out}$ are the parallelism along the height, input channel and output channel dimensions, which is determined by the hardware and original ISA. The convolution of the last $Para_{in}$ input channels is CALC_F, and the convolutions for the former input channels are CALC_I.

B. How To Interrupt: Virtual Instruction

There are four stages to handle interrupt. For the instruction flow illustrated in Figure 5(c), the interrupt stages are shown in Figure 5(e), including: (1) Time for finishing the current operation, $t1$. (2) Time to backup, $t2$. (3) Time for the high-priority task, $t3$. (4) Time to restore the low-priority task ,$t4$. The the latency to respond the interrupt is $t_{latency} = t_1 + t_2$. The extra cost for interrupt is $t_{cost} = t_2 + t_4$. There are different methods to implement interrupt in CNN accelerators.



(a) Scheduling of instruction-driven CNN accelerator. The final results of $Para_{out}$ output channels are calculated (solid arrows) before the next output channels (dashed arrow).

(b) Single Blob Save.

(c) Multiple Blob Save.

(d) Interrupt for Single Blob Save.

(e) Interrupt for Multiple Blob Save.

Fig. 5. Scheduling Illustration

CPU-Like. When an interrupt request occurs in CPU, CPU backs up all the on-chip registers to DDR. However, there are only tens of registers in CPU, and the volume of the backed-up data is less than 1 KB [24]. In CNN accelerators, there are hundreds of KB $\sim$ several MB on-chip caches [12], [15] to store input feature-maps or weights. Thus, the extra data transfer increases both the interrupt response latency($t_{latency}$) and the additional cost (t_{cost}).

Layer-by-layer. Most accelerators run the CNN layer by layer [12], [15]. There is no extra data transfer for the accelerator to switch between different tasks after each layer, thus, $t_{cost} = 0$. However, the position of the interrupt request is irregular and unpredictable. When an interrupt occurs inside a CNN layer, the CNN accelerator needs to finish the whole layer before switching, which leads to the high response latency($t_{latency}$).

We propose the **virtual-instruction-based** method (VI method) to enable low-latency interrupt. To reduce the interrupt response latency, our virtual-instruction-based method is interruptible inside each layer. We add some virtual instructions to the original instruction sequence to enable the interrupt. The virtual instructions, which contain the backup and recovery instructions, are responsible for backing up and restoring on-chip caches.

C. Where To Interrupt: After SAVE/CALC_F

We analyze the interrupt cost and select the positions of adding the virtual instructions. The backup/recovery data for different interrupt positions at each kind of instruction are listed in the Backup/Recovery columns of Table I.

D. Latency Analysis

In this subsection, we quantitatively analyze the impact of interruptible position on interrupt respond latency. As introduced in Section IV-A, each CALC instruction processes the convolution according to the hardware parallelism with $Para_{height}$ lines from $Para_{in}$ input channels to $Para_{out}$ output channels. We here note the computation of a CALC instruction an $instructionpulse$, or $pulse$. The compuataion time of each pulse is related to the hardware architecture and the width of the convolution layer. The larger the width, the larger the workload of a single calculation instruction, and thus the CALC instruction consumes more time. We note the time consumption of a pulse as a function of the width of the featuremaps, t_{pulse}:

$$t_{pulse}(W) = t_{Calc\_Hardware} \times W \tag{1}$$

$t_{Calc\_Hardware}$ indicates the time for hardware to produce the intermediate results of one pixel, which is defined by the hardware design and the clock frequency. W is the width of the featuremaps, indicating the workload of the instruction.

The worst case of waiting for finishing the current operation of the Layer-By-Layer interrupt method is that the interrupt request occurs at the beginning of the layer. In this case, the accelerator will wait until finishing the whole layer. The calculation of the whole layer consists of N_{pulse_layer} successive CALC instructions. We note the worst time of waiting to finish the current layer, which is the total time of these pulses, as t_{1_layer}.

$$N_{pulse\_layer} = \frac{Ch_{in} \times Ch_{out} \times H}{(Para_{in} \times Para_{out} \times Para_{height})} \tag{2}$$

$$t_{1\_layer} = N_{pulse} \times t_{pulse}(W) \tag{3}$$

Ch_{in} and Ch_{out} is the number of input channels and output channels. H is the height of featuremaps.

The calculation of the whole CalcBlob consists of N_{pulse_VI} successive CALC instructions. We note the worst waiting time in this case as t_{1_VI}.

$$N_{pulse\_VI} = \frac{Ch_{in} \times Para_{out} \times Para_{height}}{(Para_{in} \times Para_{out} \times Para_{height})} \tag{4}$$

$$t_{1\_VI} = N_{pulse\_VI} \times t_{pulse}(W) \tag{5}$$

Because our Virtual-Instruction method (VI method) only interrupts the execution after CALC_F and SAVE, there is no extra data transfer for the intermediate results. The backup operation in the VI method only transfers the final results,

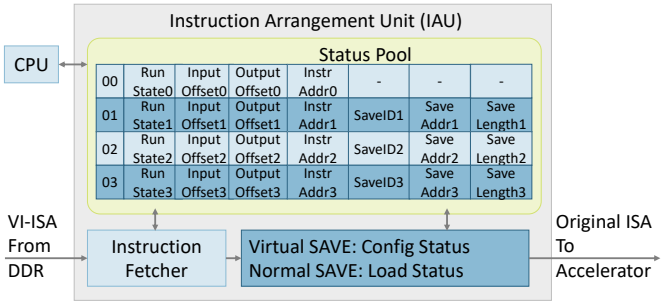


Fig. 6. Hardware architecture of IAU.

which are also transfered to DDR with the SAVE instructions in the Layer-By-Layer method. Experiment results show that the data transfer time for the final results is much less than the calculation time (less than 20%), in both the Layer-By-Layer method and the VI method. Thus the latency to respond to the interrupt request ($t_{latency}$) is mainly determined by the time of finishing the current operation.

Compared with the Layer-By-Layer method, the latency of our method is reduced to R_l.

$$\begin{aligned} R_l &= \frac{\bar{t}_{latency\_VI}}{\bar{t}_{latency\_layer}} \simeq \frac{\frac{1}{2} \times t_{1\_VI}}{\frac{1}{2} \times t_{1\_layer}} = \frac{N_{pulse\_VI} \times t_{pulse}(W)}{N_{pulse} \times t_{pulse}(W)} \\ &= \frac{N_{pulse\_VI}}{N_{pulse}} = \frac{Ch_{in} \times Para_{out} \times Para_{height}}{Ch_{in} \times Ch_{out} \times H} \\ &= \frac{Para_{out} \times Para_{height}}{Ch_{out} \times H} \end{aligned} \tag{6}$$

E. Instruction Arrangement Unit (IAU)

Instruction Arrangement Unit (IAU) is the hardware to handle the computing requirements from tasks with different priorities. The IAU monitors the interrupt status and generates the original ISA instruction sequence. The original CNN accelerator does not need to know the interrupt status, and only operates the instructions provided by IAU.

The hardware implementation of IAU is shown in Figure 6, which supports four tasks with different priorities. Task 0 has the highest priority and is not interruptible. InstrAddr records the address to fetch the instructions of the corresponding task. The InputOffset and the OutputOffset, which indicate base address offsets of the input and output data, are configured by the software. SaveID, SaveAddr, and SaveLength record the status when an interrupt occurs. Subsequent not-virtual SAVE instructions will be modified according to the recorded interrupt status (SaveID, SaveAddr, and SaveLength), to avoid duplicate output data transfer.

F. Example of Virtual Instruction

The example is based on a straightforward convolutional layer, which has only one input channel and two output channels. The convolution kernel size is 1×1. The shape of the input and output feature-maps is 2×16 (Figure 7(a)).

TABLE II
HARDWARE CONSUMPTION OF THE PROPOSED HARDWARE

	$\#DSP$	$\#LUT$	$\#FF$	$\#BRAM$
On-Board resource	2520	274080	548160	912
CNN accelerator	1282	74569	171416	499
IAU	0	2268	4633	4
FE post-processing	25	17573	29115	10

The parallelism of the CALC instruction in this example is $Para_{in} = 1$, $Para_{out} = 1$, $Para_{height} = 2$.

Thus, the two output channels are calculated by two CALC_F instructions (instruction 3 and 7 in Figure 7(c)). The addresses used in the instruction example are listed in Figure 7(b). Figure 7(c) is the instruction sequence from DDR with VI-ISA. Figure 7(d) is the executed original ISA instructions without interrupt. When an interrupt occurs at the first CalcBlob, Figure 7(e) illustrates the backup/recovery instructions (Blue) and the modified SAVE instruction (Red).

V. Evaluation and Results

A. Experiment Setup

The hardware-in-loop evaluate environment is illustrated in Figure 8(a). There is a simulation server providing the simulation environment based on AirSim [25], which is a high-fidelity visual and physical simulation for autonomous vehicles. The AirSim simulation server provides the camera data for each agent. Two Xilinx ZCU102 boards [26], with ZU9 MPSoC [27], are responsible for the calculation of each agent. The components in **??** for each agent are implemented in the ZCU102 board. The implementation of the FE (① in **??**), SuperPoint [5]), are introduced in former sections. GeM [9] is used to implement the PR module (②). GeM is a CNN-based method with ResNet101 [20] as the backbone, and the post-processing of GeM calculates the 3-norm of the output featuremaps. The VO module (③) in the experiment is the PnP [28] method, which is widely used in the feature-point based VO. The DOpt module (④) is proposed in [29] and also used in former distributed SLAM system [30]. The Map Merging [31] (⑤), Exploration [32] (⑥), and Navigation [33] (⑦) in this work are provided by the ROS framework.

The hardware resources are listed in Table II. The hardware resources are provided by Vivado after hardware implementation. Vivado [34] is the development toolchain for MPSoC provided by Xilinx. The CNN backbone is calculated by the Angel-Eye CNN accelerator [12] on the FPGA side of ZCU102 (Programmable logic, PL side). The FE post-processing steps run on our proposed accelerators, also on the PL side. The PL side has 2 clock frequencies. The CNN accelerator and the IAU are running at 300MHz. The accelerator for FE post-processing is running at 200MHz. Compared with the CNN accelerator, IAU and FE post-processing use very little hardware resources.

B. Virtual Instruction-based interrupt

1) Interrupt response latency and extra cost: We evaluate the latency to respond the interrupt and the performance

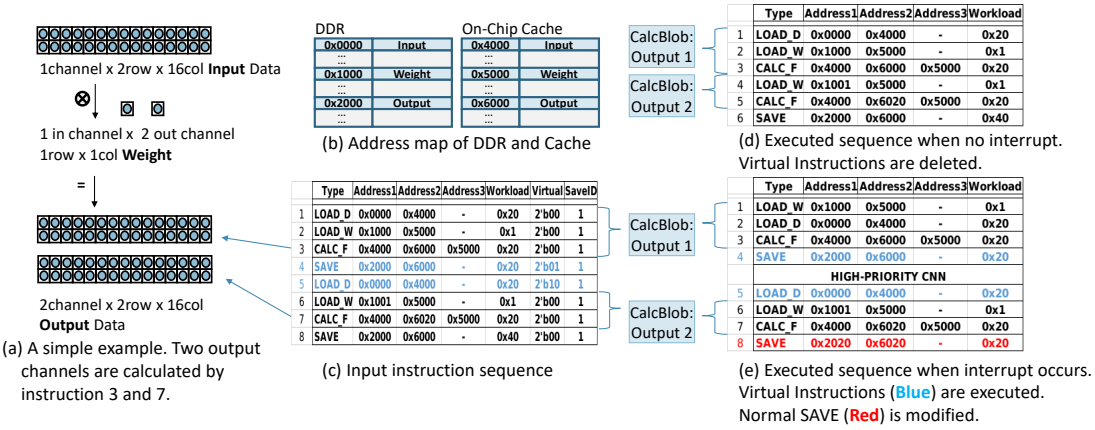


Fig. 7. A simple example of our proposed virtual-instruction-based interrupt.

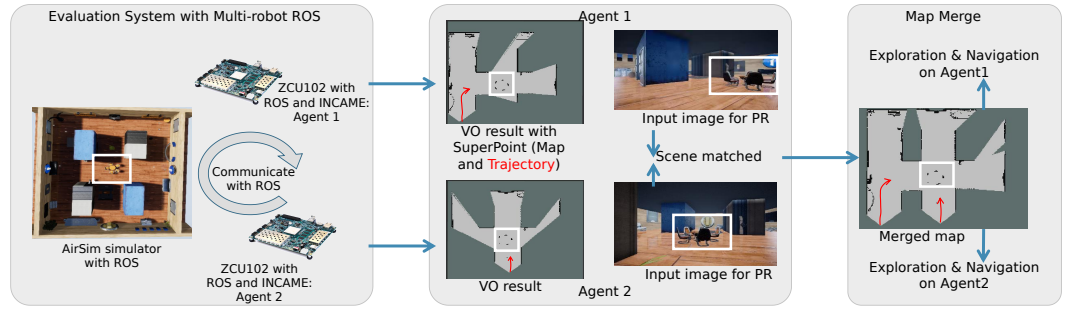


Fig. 8. Multi-robot exploration: environment and results.

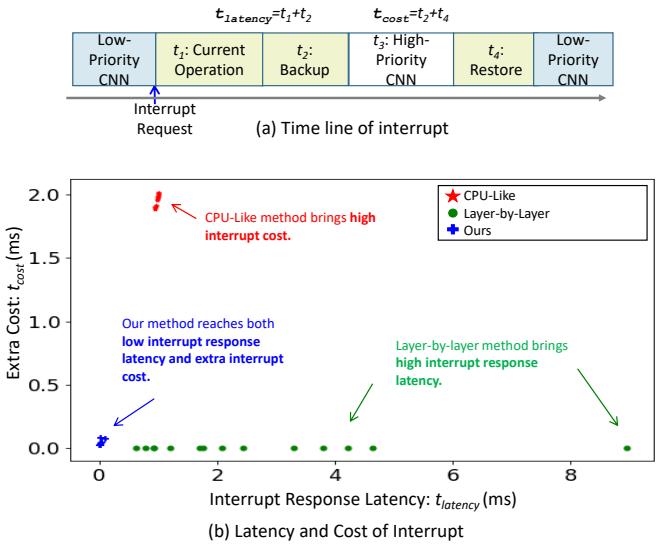


Fig. 9. The interrupt response latency & extra time cost.

degradation of different interrupt method. In MR-Exploration, only the low-priority PR task is interruptible, and the interrupt position is unpredictable. GeM [9] is used to implement the PR module in the experiment. The CNN backbone of the GeM is ResNet101 [20], which contains 101 convolution layers. The input shape of the CNN is $480 \times 640 \times 3$. The parallelism of the Angel-Eye is $Para_{height} = 8$, $Para_{in} = 16$, $Para_{out} = 16$, i.e. each CALC instruction processes 8 lines from 16 input channels to 16 output channels.

As shown in Figure 9(a), the latency to respond the interrupt in CPU-like method consists of the time to finish current executing instruction and the data backup time ($t_{latency} = t_1 + t_2$) for the on-chip data/weights caches (totally 2.2MB). The latency in layer-by-layer interrupt is the time to finish current layer. The latency of our virtual-instruction-based method is the time to finish current executing instruction and the backup time for the calculated output results.

The cost of CPU-like interrupt is the data transfer time of all the on-chip caches (totally 2.2MB) to/from DDR ($t_{cost} = t_2 + t_4$). The cost of virtual-instruction-based method is only the recovery of the input/weights from DDR to on-chip caches ($t_{cost} = t_4$). There is no extra cost for the layer-by-layer interrupt.

We randomly sample 12 positions of the ResNet101 CNN backbone. The interrupt response latency and the extra time

cost for different implementation of interrupt at the positions are listed in Figure 9(b). The CPU-like interrupt consumes the most extra cost (t_{cost}). Though the layer-by-layer interrupt consumes no extra time, the latency is much higher than our virtual-instruction-based interrupt. This is because the layer-by-layer interrupt need to wait for completion of a layer. The performance at same interrupt position in our proposed virtual interrupt can interrupt inside a layer, with lower latency.

Furthermore, though the network structures differ between different CNNs, the convolutional layers, which are the basic component in CNN, are similar between different CNNs. INCA monitors the running status inside each layer, and the interrupt respond latency and extra cost are only relevant to the currently operating layer. Thus, the latency and cost are also similar between different CNNs. In conclusion, the process for different CNN tasks are similar, and the cost of different tasks are similar.

C. ROS based MR-Exploration

The results of the Multi-Robot Exploration based on INCA is shown in Figure 8. The space in the AirSim [25] for the robots to explore is shown in Figure 8(a). It is a simple rectangle area with four different pillars, and some chairs at the center (in the white box). Figure 8(b) shows how PR works for map merging. The FE and VO of each agent produce the local map and trajectory on each ZCU102 board. When the PR threads on different agents find out a similar scene, the relative pose of the two agents at the similar scene is calculated. The map and the trajectory is merged with the calculated relative pose, as shown in Figure 8(c).

In this example, the FE and PR are both executed on the same Angel-Eye accelerator. The frequency of the input camera is 20fps, and each input frame is fed to the FE, and FE module would take up accelerator. While the CPU process VO with the feature-points from FE, the accelerator can switch to process the low-priority PR task. Because the executing time of VO varies, the time to finish a PR task is different. In this example, the time from the beginning of a PR to its end is 320$\sim$500 ms. Thus, the PR process one key frame every 7$\sim$10 input frames.

VI. Conclusion

In this paper, we propose an interruptible CNN accelerator and a deployment framework, INCA, for multi-robot exploration. With the help of the virtual-instruction-based interrupt method, the CNN accelerator can switch between different CNN tasks with low interrupt response latency and low extra cost. Note that the development of CPU task scheduling evolved from single-core multi-tasking to multi-core multi-tasking. Similarly, INCA currently focuses on interrupt support for single-core multi-tasking. We plan to investigate the multi-core multi-tasking for CNN accelerators as part of future work. INCA only needs to modify the instruction fetch module to IAU in hardware. Thus, it is easy to extend to handle other instruction-driven accelerators. Therefore, with the help of INCA, the independent software in ROS can access the accelerator without hardware resources conflicts, on various CNN accelerators.

References

[1] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[2] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring r-cnn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6409–6418.

[3] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[4] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," *arXiv preprint arXiv:1806.08548*, 2018.

[5] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *CVPR Workshops*, 2018, pp. 224–236.

[6] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *ICCV*, 2015, pp. 118–126.

[7] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *ECCV*. Springer, 2016, pp. 467–483.

[8] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *CVPR*, 2016, pp. 5297–5307.

[9] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[10] R. Mur-Artal and J. D. Tards, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, pp. 1255–1262, 2016.

[11] H. Jégou and A. Zisserman, "Triangulation embedding and democratic aggregation for image search," in *CVPR*, June 2014, pp. 3310–3317.

[12] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 35–47, 2017.

[13] J. Yu, G. Ge, Y. Hu, X. Ning, J. Qiu, K. Guo, Y. Wang, and H. Yang, "Instruction driven cross-layer cnn accelerator for fast detection on fpga," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, p. 22, 2018.

[14] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *FPL*. IEEE, 2016, pp. 1–9.

[15] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *FPGA*. ACM, 2016, pp. 26–35.

[16] L. Lu, Y. Liang, Q. Xiao, and S. Yan, "Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs," in *FCCM*, Apr. 2017, pp. 101–108.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[18] R. Ramsauer, J. Kiszka, D. Lohmann, and W. Mauerer, "Look mum, no VM exits! (almost)," *CoRR*, vol. abs/1705.06932, 2017. [Online]. Available: http://arxiv.org/abs/1705.06932

[19] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *CVPR*, 2016, pp. 1646–1654.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[21] "DNNDK User Guide - Xilinx," 2019. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf

[22] K. Papadimitriou, A. Dollas, and S. Hauck, "Performance of partial reconfiguration in fpga systems: A survey and a cost model," *Acm Transactions on Reconfigurable Technology & Systems*, vol. 4, no. 4, pp. 1–24, 2011.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[24] S. B. Furber, *ARM system-on-chip architecture*. pearson Education, 2000.
[25] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.
[26] "Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit," 2019. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html
[27] "UltraScale MPSoC Architecture," 2019. [Online]. Available: https://www.xilinx.com/products/technology/ultrascale-mpsoc.html
[28] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Ep n p: An accurate o (n) solution to the p n p problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.
[29] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *The International Journal of Robotics Research*, vol. 36, pp. 1286–1311, 2017.
[30] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual slam," in *ICRA*. IEEE, 2018, pp. 2466–2473.
[31] T. Andre, D. Neuhold, and C. Bettstetter, "Coordinated multi-robot exploration: Out of the box packages for ROS," in *GLOBECOM WiUAV Workshop*, Dec. 2014.
[32] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *IROS*, Sept 2017, pp. 1396–1402.
[33] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *ICRA*, 2010.
[34] "Vivado Design Suite," 2019. [Online]. Available: https://www.xilinx.com/support/university/vivado.html