CNN-based Monocular Decentralized SLAM on embedded FPGA

Jincheng Yu[1], Feng Gao[1], Jianfei Cao[3], Chao Yu[1], Zhaoliang Zhang[1], Lu Tian[2], Zhengfeng Huang[3], Yu Wang[1] and Huazhong Yang[1]

***Abstract*— Decentralized visual simultaneous localization and mapping (DSLAM) can share locations and environmental information between robots, which is an essential task for many *multi-robot* applications. For the keyword *"robot"*, the visual odometry (VO) is a basic component to estimate the 6-DoF absolute pose, and for the keyword *"multi"*, decentralized place recognition (DPR) is a fundamental element to produce candidate place matches. Previous works develop fixed-point CNN accelerators for many tasks. Different from other applications, VO task needs higher numerical precision. Thus the traditional quantization method does not work for VO, so We propose a fixed-point fine-tune method for the CNN-based monocular VO. Due to the limited resources, it is challenging to do DPR for every input frame. We explore the influence of the DPR frequency on the DSLAM results, and find out a proper DPR frequency to balance the accuracy and speed. We propose a cross-component pipeline scheduling method, improving the DPR frequency, and further promoting the final accuracy of DSLAM.**

I. Introduction

Decentralized visual simultaneous localization and mapping (DSLAM) can share locations and environmental information between robots, which is an essential task for many multi-robot applications. Cieslewsk et al. [1] conclude the basic procedure of DSLAM as fig 1.

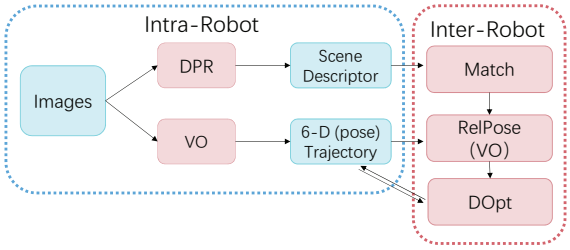


Fig. 1. DSLAM framework. **VO** is used to calculate the intra-robot 6-DoF absolute pose from the input frames. **DPR** produces a compact image representation to be communicated among robots. **Match** stage finds out candidate inter-robot place recognition matches. **RelPose** requires data from the matched robots and establishes relative poses between the robots trajectories. **DOpt** does optimization with the trajectories, intra-robot pose measurements from VO and inter-robot relative poses from RelPose, and updates the trajectories.

This framework contains five modules: Decentralized Place Recognition (DPR), Visual Odometry (VO), Match, Relative Pose estimation (RelPose) and Decentralized Optimization (DOpt). DPR and VO are intra-robot operations, requiring high computing resources on embedded system. The results of DPR can be used for both intra- and inter-robot loop-closure detection. Match, RelPose and DOpt are inter-robot operations, and consume most of the communication resources of DSLAM sytem. The RelPose part relies on the VO components since it can benefit from re-using VO's data and computing resources.

[1]Electronic Engineering Department, Tsinghua University, Beijing, China yjc16@mails.tsinghua.edu.cn, yu-wang@tsinghua.edu.cn
[2] Xilinx,Inc.(Beijing), Beijing, China
[3]School of Microelectronics, Hefei University of Technology, Hefei, China

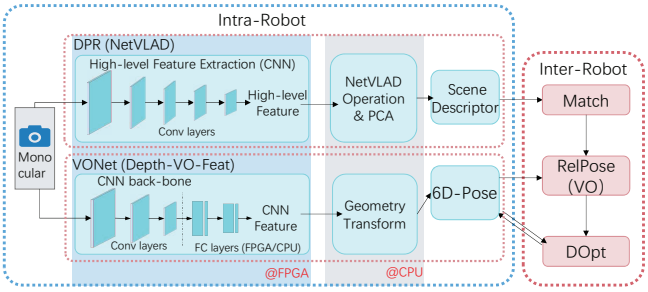


Fig. 2. Our framework adopt Depth-VO-Feat [4] in DSLAM system as the monocular VO, and we use NetVLAD, the same method in [1], to do DPR. The CNN backbones of DPR and VO are calculated on CNN accelerator in fixed-point number at the FPGA side. The post processing operations, such as PCA and geometry transformation, are calculated at the CPU side. In order to reduce the precision loss of VO, a small part of VONet (FC layers) is also calculated in floating-point number on CPU.

Cieslewsk et al. [1] use ORB-SLAM [2] for VO and NetVLAD [3] as the DPR component. These two algorithms both consume a large amount of computing and storage resources, and pose a huge challenge to the DSLAM in embedded systems.

With the development of convolutional neural network (CNN), we can reconstruct the depth and pose with the absolute scale directly from a monocular camera, making monocular VO more robust and efficient. And monocular VO methods, like Depth-VO-Feat [4], make DSLAM system much easier to deploy than stereo ones.

In order to take advantage of CNN in the embedded DSLAM system, we adopt Depth-VO-Feat [4] as the monocular VO, and we use NetVLAD [3], which is also used in previous DSLAM system [1], to do DPR. We build up a CNN-based monocular real-time DSLAM system on the embedded FPGA platform with following contributions:

- Fully fixed-point number work for many applications, such as image classfication [5] and object detection [6]. Different from these applications, VO task needs higher numerical precision. Thus, the traditional quantization method does not work for our monocular VO, so we propose a fixed-point fine-tune method for the CNN-

based VO.

- The DPR in previous DSLAM system [1] is operated only for key frames. Different from the ORB-SLAM, the CNN-based VO cannot tell the key frames. Thus, we need to execute DPR as frequently as possible. Due to the limited hardware resources, it is challenging to do DPR for every input frame. We explore the influence of the DPR frequency on the DSLAM results and propose a cross-component pipeline scheduling method, improving the DPR frequency.
- To the best of our knowledge, this is the first work to implement all components of monocular DSLAM with CNN. We deploy the system on the Xilinx ZCU102 MPSoC [7] hardware platform with DNN Processing Unit (DPU) [8]. ZCU102 is a evaluation board for MPSoC [9] provided by Xilinx and DPU is a CNN accelerator. The proposed DSLAM system is illustrated in fig 2.

II. Hardware Architecture

The CNN-based VO and DPR components consume more than tens of billions operations and thus are difficult to deploy on embedded systems. The Xilinx MPSoC [9] is a chip with ARM cores and FPGA fabric. The architecture of MPSoC is illustrated in fig 3. Our target hardware platform, ZCU102 [7], is a popular evaluation board for the MPSoC chip.

The ARM cores with an embedded Linux operating system are called Processing System (PS). The FPGA fabric is called Programmable Logic (PL). The peripherals like cameras and communication units (WiFi or others) are accessible to PS. The high-bandwidth on-chip AXI interface is used to communicate between PS and PL. PS and PL can also share DDR to transfer large amounts of data such as each frame of photos. Xilinx CNN accelerator, called DNN Processing Unit (DPU) [8], is one of the state-of-the-art CNN accelerators and is known for its energy efficiency in running various CNN structures. We deploy the DPU on the PL side of Zynq SoC.

Though FPGA can significantly improve the performance and energy efficiency of CNN inference, it cannot efficiently calculate the floating-point number and thus requires fixed-point parameters and intermediate data in CNN.

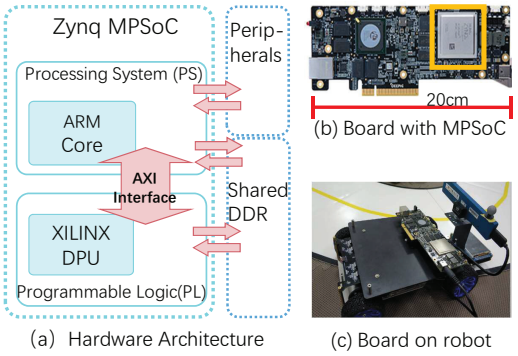


Fig. 3. Xilinx Zynq MPSoC Platform

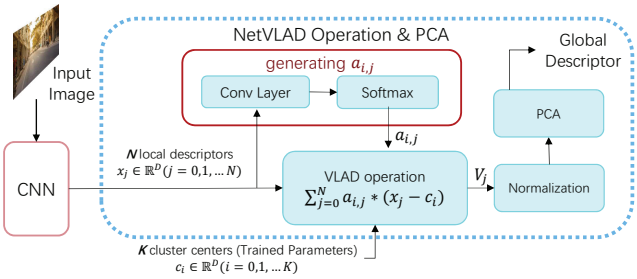


Fig. 4. NetVLAD Operation. The components of NetVLAD operation are deployed on PS side, including the Conv Layer to generate $a_{i,j}$ and PCA.wq

III. Methodology

A. Decentralized Place Recognition (DPR) with NetVLAD

The goal of place recognition (DPR) is to calculate a given frame into a limited set of places. Every place can be encoded as a compacted code that can be easily transferred at low communication costs. Recent advances of CNNs make it possible to build a robust end-to-end place recognition method, and NetVLAD[3] is one of the best CNN-based DPR methods. The dataflow of NetVLAD is illustrated in fig 4.

The CNN backbone extracts the features map from the input image, and then the $W \times H \times D$ feature map is transformed into $N \times D$ local descriptors. Vector of Locally Aggregated Descriptors (VLAD) is a popular operation to capture statistical information of local descriptor aggregated over the input image. There are totally N descriptors $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)$. There are K cluster centers $(\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_K)$ as VLAD's parameters, which are determined in the training phase. Each descriptor is a D dimension vector as well as each cluster center. The weighted sum of local descriptors to each cluster center reflects the "coefficient" of the cluster center. Concatenating the coefficient of each cluster center encodes the whole input picture.

In the VLAD step, the coefficient of the i^{th} cluster center $(\boldsymbol{V_i})$ is calculated as follows, and each coefficient is a vector whose length is D.

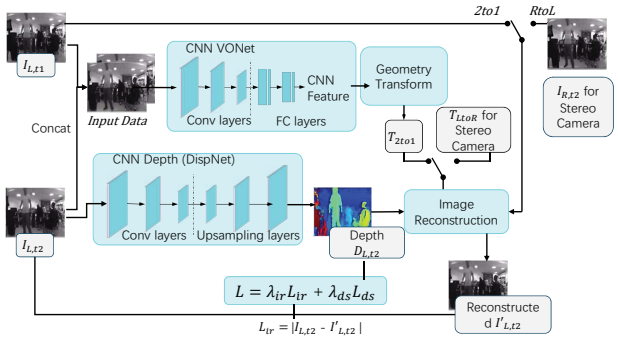
$$\boldsymbol{V_i} = \sum_{j=0}^{N} a_{i,j} * (\boldsymbol{x_j} - \boldsymbol{c_i}) \tag{1}$$

A convolution layer also generates the weight parameters $a_{i,j}$. The global descriptor $(\boldsymbol{V_{all}})$ which encodes the whole image is the concatenation of the coefficient of each cluster center, and $\boldsymbol{V_{all}}$ is a vector whose length is $K \times D$ length.

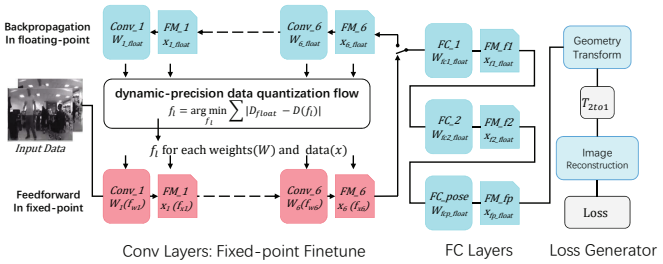
$$\boldsymbol{V_{all}} = \{\boldsymbol{V_1}, \boldsymbol{V_2}, ..., \boldsymbol{V_K}\} \tag{2}$$

The global descriptor is then compressed by Principal Component Analysis (PCA) to obtain the final compact descriptor $(\boldsymbol{V_{compact}})$ of the image.

$$\boldsymbol{V_{compact}} = PCA(\boldsymbol{V_{all}}) \tag{3}$$



(a) Illustration of training framework for visual odometry. The image reconstruction loss (L_{ir}) is used to jointly train two networks for depth (DispNet) and odometry estimation (VONet).



(b) Fixed-point fine-tune method. The blue elements are the general fine-tune method in floating-point number, including loss generation and backpropagation. The red elements are the fixed-point feedforward convolutional layers. FM is short for feature map. The **dynamic-precision data quantization flow** find out the optimal fractional length (f_l) for weights ($w1, w2, ..., w6$) and intermediate featuremaps($x1, x2, ..., x6$) respectivaly.

Fig. 5. The training and fixed-point fine-tune method for CNN based VO.

To generate the weighted parameters $\boldsymbol{a}$, a softmax operation, which is not supported by the DPU, is processed following a convolution layer with 1×1 kernels. The NetVLAD operations, including 1×1 convolution, softmax, VLAD and normalization, are not efficiently supported by the DPU or need floating-point number system. Therefore, the NetVLAD operations are running on the PS side.

B. Quantilization for NetVLAD

To deploy the NetVLAD CNN backbone on the DPU, we need to quantize the floating-point CNN model to the FPGA friendly fixed-point model. Previous works use the **dynamic-precision data quantization flow**[5] to find the optimal fractional length (f_l) of the weights and activations for each layer of CNN respectively.

For a fixed-point number n, its value can be expressed as:

$$n = \sum_{i=0}^{bw-1} B_i \cdot 2^{-f_l} \cdot 2^i \tag{4}$$

where bw is the bit width and f_l is the fractional length which can be negative. B_i is the digit on the i^{th} bit, which is 0 or 1. For the parameters or activations of a layers, the data quantization flow aims to find the optimal fractional length for the corresponding data:

$$f_l = \arg\min_{f_l} \sum |D_{float} - D(bw, f_l)| \tag{5}$$

where D is layer parameters (in fig 5(b) is W) or the activations (in fig 5(b) is x) of a layer, and $D(bw, f_l)$ is the fixed-point representation under given bw and f_l. We analyze the dynamic range of the data, and then find the optimal f_l.

In our hardware design, the bit width of all weights and intermediate feature maps are fixed to 8. The bw in eqs. (4) and (5) is set as $bw = 8$.

$$D(f_l) = D(8, f_l) \tag{6}$$

In the dynamic-precision data quantization flow, we avoid to fine-tune the CNN model. Previous work shows that this quantization method leads to neglectable accuracy decline in many applications, such as classification [5] and object detection [6].

This method also works for NetVLAD, the run the CNN layers in fixed-point number and keep the VLAD operations in floating-point number. Quantitative results will be given in section IV.

C. CNN-based Visual Odometry

We adopt Depth-VO-Feat [4] in the DSLAM system to estimate the pose from the input monocular camera. The training and forwarding framework is illustrated in fig 5(a).

Depth-VO-Feat uses image reconstruction loss (L_{ir}) as a self-supervised signal and jointly trains two networks for depth (DispNet) and odometry estimation (VONet) without external supervision. DispNet predicts the depth of each pixel in the reference frame ($I_{L,t2}$). The reconstructed frame ($I'_{L,t2}$) is reconstructed from the time-adjacent frames ($I_{L,t1}$) or the frame of the other camera of the binocular sensor ($I_{R,t2}$) with the predicted VO results (T_{2to1}) or the known camera motion between stereo cameras (T_{RtoL}). The predicted depth ($I_{L,t2}$) is also used in the reconstruction

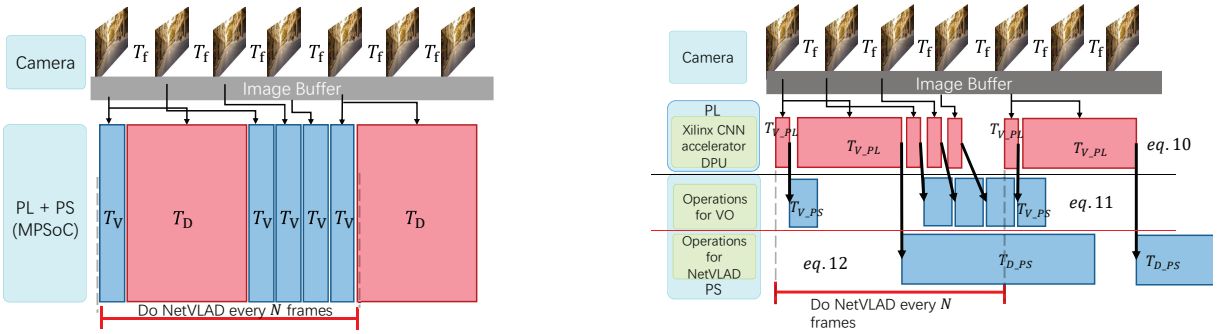
The reconstruction loss is defined as the sum of the difference of each pixel in $I'_{L,t2}$ and $I_{L,t2}$.

$$L_{ir} = \sum_{p} |I_{L,t2}(p) - I'_{L,t2}(p)| \text{ , } p \text{ for each pixel} \tag{7}$$

By including this reconstruction loss for both spatially adjacent input pair $\{I_{L,t2}, I_{R,t2}\}$ and temporally adjacent pair $\{I_{L,t2}, I_{L,t1}\}$, we use stereo sequences in the training phase and monocular sequences in the testing phase. With the calibrated spatial relationship between the left and right cameras (T_{RtoL}), our VONet can learn the real world scale for time-adjacent frames (T_{2to1}).

D. Fixed-Point fine-tune for CNN-based VO

The VO task requires much higher computational precision than other applications, and the VO errors accumulate during the DSLAM. If we directly quantize the entire VONet with the dynamic-precision data quantization flow introduced in section III-B, the accuracy declines sharply.



(a) Scheduling without pipeline. There are only two threads: Camera read and computation. The black lines indicate the data dependance across different threads.

(b) Scheduling with cross-module pipeline. There are four threads: Camera read, DPU core at PL, PS Operations for VO, and PS Operations for NetVLAD.

Fig. 6. The computation pipeline with/without cross-component scheduling.

To keep the accuracy of the VONet, we adopt the **fixed-point fine-tune method**[6] to fine-tune the VONet. There are two steps in the general fine-tune method: 1) feedforward and 2) backpropagation, both of which are calculated in floating-point numbers. In the fixed-point fine-tune method, fixed-point feedforward is applied to some layers, while the back-propagation step still runs in floating-point. After each back-propagation, the floating-point weights (W_{float}) and the intermediate featuremaps (x_{float}) are updated. According to eq. (5), we find out the optimal f_l for each weight and intermediate featuremap.

In fig 5(b), we fine-tune the convolution layers ($Conv\_1$ - $Conv\_6$) with the fixed-point fine-tune method and directly fine-tune the fully connected (FC) layers (FC_1, FC_2, FC_pose) in floating-point number. To balance the speed and the accuracy, we also attempt to fine-tune some of FC layers in fixed-point. The results of speed and accuracy will be shown in detail in section IV.

E. Deployment of VO and NetVLAD on MPSoC

We are here to remind readers that although there is a fixed-point fine-tune method for VO, its purpose is to obtain a high-precision fixed-point network. In actual deployment, we only need to deploy the feedforward phase of the fixed-point network to the embedded system.

As introduced in section II, the CNN parts of VO and NetVLAD, especially the convolutional layers, are transferred to fixed-point and deployed to the PL part of MPSoC. Other parts, such as the VLAD operation in NetVLAD and some FC layers in VO, are calculated on the PS part of MPSoC.

Different from the previous DSLAM system [1], whose DPR is operated only for key frames. Our CNN-based VO cannot tell the key frames. Thus, we need to execute DPR as frequently as possible.

Because the DPU kernel does not support multi-threading, we need to prioritize VO calculations, which are calculated for each input frame. Netvlad is assigned to be executed at the time interval of VO calculation, so that the execution pipeline of VO and NetVlad is as shown fig 6(a). The time interval for reading the camera is T_f, the VO run time is T_V and the NetVLAD run time is T_D.

VO needs to be executed once for every **1** input frame. The NetVLAD operation is operated for every N VO frames. The operation of the VO frames ($N \times T_V$) and the NetVLAD operation (T_D) should be finished within the time of reading N frames ($N \times T_f$). N is constrained by eq. (8).

$$N \times T_f > T_D + N \times T_V \tag{8}$$

F. Cross-Components Scheduling

We optimize the pipeline of two components on Zynq MPSOC to schedule them effectively. The pipeline is illustrated in fig 6(b). Both NetVLAD (T_D) and VO (T_V) time can be divided into two parts:

$$\begin{aligned} T_D &= T_{D\_PL} + T_{D\_PS} \\ T_V &= T_{V\_PL} + T_{V\_PS} \end{aligned} \tag{9}$$

T_{D_PL} is the CNN time of NetVLAD deployed on the PL side (Programmable Logic, FPGA side). T_{D_PS} is the VLAD operations of NetVLAD, which is operated on the PS side (Processing System, CPU side). T_{V_PL} is the CNN time of VO on the PL side. T_{V_PS} is the time of the FC layers in floating-point number and the geometry transformation of VO, which is operated on the PS side.

Considering the thread on PL, the PL part of the N VO frames ($N \times T_{V\_PL}$) and the NetVLAD operation (T_{D_PL}) should be finished within the time of reading N frames ($N \times T_f$). The time constraint is given as eq. (10).

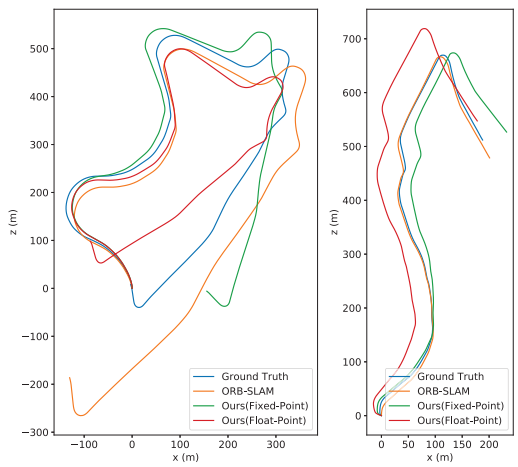
$$N \times T_f > T_{D\_PL} + N \times T_{V\_PL} \tag{10}$$

The thread for VO on PS constrains the NetVLAD frequency as eq. (11).

$$N \times T_f > T_{D\_PL} + T_{V\_PL} + (N-1) \times T_{V\_PS} \tag{11}$$

The PS part of current NetVLAD should finish before computing the PS part of the next NetVLAD frame. This constraint can be written as eq. (12).

$$N \times T_f > T_{D\_PS} \tag{12}$$



(a) Seq. 09 (b) Seq. 10

Fig. 7. Qualitative evaluation of visual odometry on the KITTI Odometry test sequences (09, 10).

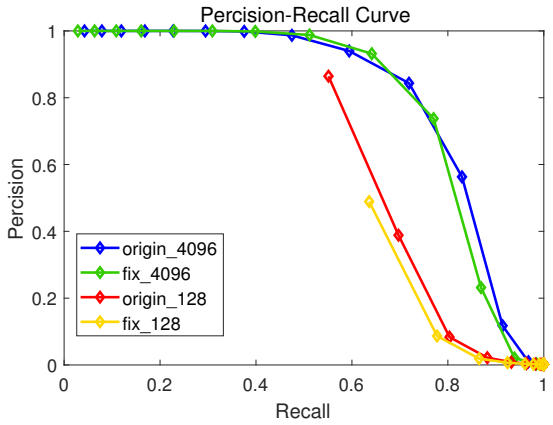


Fig. 8. PR curves on sequence 00. The original floating-point NetVLAD with an output of 4096 dimensions (Blue). The fixed-point NetVLAD 4096-D output (Green). The original floating-point NetVLAD with 128-D output (Red). The fixed-point NetVLAD 128-D output (Yellow). The 128-D shorter code consists of the first 128 elements of the 4096-D longer code.

The execution time of our design will be given in section IV.

IV. Experiments

In this section, we will evaluate the speed and accuracy of each proposed CNN components, as well as the impact of our optimization on final DSLAM result.

A. VO with Fixed-Point finetune

We evaluate our VO approach with DPU on KITTI dataset [10]. The dataset contains 61 labeled video sequences with stereo pairs, with original image size at 1242×375 pixels. In the following experiments, we resize the image to 608×160 in training and testing processes, just like the original Depth-VO-Feat [4] does. We use the 00-08 sequences from KITTI as the training set, and evaluate the network on sequence 09 and 10 on the sub-sequences at the length of [100,200,...,800] and report the average translational and rotational errors in table I. The comparison of the estimated trajectory for different methods is illustrated in fig 7. We use the popular SLAM system, ORB-SLAM [2], as the baseline.

The initial learning rate is $1e^{-5}$ and the number of fine-tune iterations is 240000, which is sufficient to train Depth-VO-Feat from random initialization. We snapshot the network weights every 5000 iterations and list the accuracy of the best snapshot in table I.

As introduced in section III-D, we use the fixed-point fine-tune method to guarantee the accuracy of CNN-based VO. Besides running the convolutional layers in fixed-point number on the PL side, we also tried to make some FC layers in fixed-point number. The *Quant. Strategy* indicates the different configurations of fixed-point fine-tune. The *Fixed Part* is the fixed-point layers on PL, and for example, $(Conv+FC1, 2)$ means running all of the convolutional layers as well as the first two FC layers(FC1 and FC2) on in fixed-point number. The *Float Part* is the floating-point layers for accurate pose prediction on PS. The last FC layer (FC_{pose}) should always be operated in floating-point number. Otherwise the VO fails and the predicted rotation is always stationary.

Because the computation volume of FC layers is much less than that of CONV layers, most of the computation time is spent on the fixed-point CONV layers on the PL side. Computing FC layers on PL or PS has little effect on VO computing speed. However, computing FC layers in floating-point or fixed-point strongly affects VO accuracy. For this reason, we deploy all of the convolutional layers in floating-point number on PL and all of the FC layers in floating-point number on PS.

B. NetVLAD with DPU

We evaluate the NetVLAD performance on the loop-closure dataset based on KITTI [11]. The dataset labels the ground truth of loop closure for these sequences based on the metric positions of each image. Specifically, it compares the position of each image to the others in the sequence and regards the one as a loop if it lies within a radius of $6m$, which is suggested in [12].

The CNN model is quantized with the **dynamic-precision data quantization flow** introduced in section III-B and is deployed on the DPU on the PL side. The other VLAD operations are deployed on the PS side.

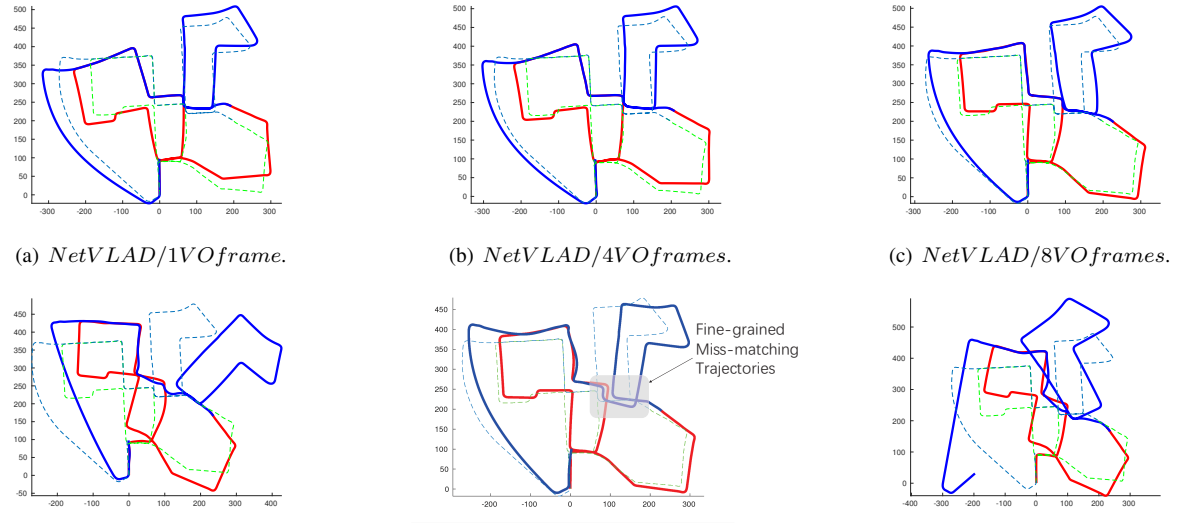
The precision-recall curves (PR curves) of the original NetVLAD with an output of 4096 dimensions (Blue), fixed-point NetVLAD 4096-D output (Green), original NetVLAD with 128-D output (Red), and fixed-point NetVLAD 128-D output (Yellow) are shown in fig 8. We take sequence 00 as the test sequence, and achieve similar accuracy with the floating-point model of the same output dimensions. The 128-D shorter code consists of the first 128 elements of the 4096-D longer code. As illustrated in fig 4, the 4096-D code is generated by the PCA operation, and the more advanced elements are in coding, the more critical they are. The first 128 elements can sufficiently describe the scenes in our following experiment.

TABLE I

VISUAL ODOMETRY (VO) RESULTS ON TEST SEQUENCES (09, 10)

Method	Quant. Strategy		Seq. 09		Seq. 10		run time
	Fixed Part (PL side)	Float Part (PS side)	t^1_{err}	r^1_{err}	t_{err}	r_{err}	$(ms/frame)$
ORB-SLAM	-		15.30	0.26	3.68	0.48	230 [2]
Depth-VO-Feat[4]	-		11.92	3.60	12.62	3.43	-
Ours	Conv+FC1,2	FC_pose	13.27	5.27	14.75	7.78	8
	Conv+FC1	FC2+FC_pose	13.80	4.38	11.3	4.30	8
	Conv	FC1,2+FC_pose	10.27	4.08	8.84	4.01	13

[1] $t_{err}(\%)$ is the average translational drift error. $r_{err}(^\circ/100m)$ is average rotational drift error.
[2] We run the monocular ORB-SLAM [2] on the PS part of Xilinx MPSoC.



(a) $NetVLAD/1VOframe$. (b) $NetVLAD/4VOframes$. (c) $NetVLAD/8VOframes$.

(d) $NetVLAD/10VOframes$. (e) $NetVLAD/12VOframes$. (f) $NetVLAD/14VOframes$.

Fig. 9. The DSLAM result of two robots(red and blue, the dashed is the ground truth). The higher NetVLAD frequency is, the better the DSLAM performs.

Empirical results show that the dynamic-precision data quantization flow without fine-tuning not only works for classification and detection tasks, it also works for scene encoding tasks. In this paper, two lengths (128 and 4096) is used for scene encoding. In the shorter code experiment, the fixed-point results are always worse than the floating-point results. In the longer code experiment, the fixed-point results may reach a better precision under some special recall rates.

C. Frequency Exploration of DPR in DSLAM

We adopt Depth-VO-Feat [4] as the VO, and we use NetVLAD [3] to do DPR in our DSLAM system. Due to the limited hardware resources, it is challenging to do DPR for every input frame. In this section, we explore the influence of the DPR frequency on the DSLAM results.

We evaluate our DSLAM system on KITTI odometry dataset. Firstly we divide the KITTI sequence 00 into 2 subsequences with a 10-frame overlap, and resize the input image to 608×160 for VO and DPR. We treat each subsequence as the raw data for each robot. We use the same method in [1] to simulate the DOpt and merge the two trajectories. The DOpt method used in this work is proposed in [13]. The DOpt method is based on the factor graph optimization method, which a popular method to estimate the trajectory in SLAM. The pose of each input camera frame is a node in the factor graph, and the VO provides the constrains between the intra-robot successive frames, in which the estimation error may accumulate. The NetVLAD provides the constrains between different robots at the same scene, or the constrains inside the same robot with loop-closure, which can correct accumulated errors.

The VO is peroated for every input frame, yet the NetVLAD can only be operated once for N input frames ($NetVLAD/Nframes$). The higher the operating frequency of NetVLAD, the smaller the N, the more constraints between different robots or intra-robot loop closure can be provided, and the better the accumulated VO errors can be corrected. The merged trajectory at different NetVLAD frequency is shown in fig 9.

As illustrated in fig 9, the less frequently the NetVLAD is operated, the trajectory merging performance is worse. When the N is higher 14, i.e., the frequency of NetVLAD is less than running NetVLAD on every 14 frames, the trajectory is divergent, and the loop-closure of the blue trajectory is totally missing.

Fig 9(e) looks not bad, and the merged trajectory seems

better than a higher NetVLAD frequency (fig 9(d)). It is because that in fig 9(e), some of the critical inter-robot loop-closures are detected, making the overall shape of the trajectory looks correct. However, as illustrated in the shadowed area in fig 9(e), the trajectories are not well merged due to the loss of intra-robot loop-closure detection. The partially detected intra-robot loop-closures detected in fig 9(d) would distort the trajectory and make the merged trajectory seems worse than fig 9(e). When the frequency of NetVLAD goes high to running once NetVLAD every 8 VO frames (fig 9(c)) or higher (figs 9(a) and 9(b)), the loop-closures between different robots and inside the same robot are adequately detected, and so the trajectories are well merged.

The empirical results shows that, in the KITTI 00 sequence, doing NetVLAD every 8 input frames is plenty for trajectory merging. The higher frequency of NetVLAD brings limited improvements in the final DSLAM system accuracy.

D. Run-Time with/without Cross-Module Scheduling

We evaluate our VO and NetVLAD implementation simultaneously on a Xilinx ZCU102 MPSOC platform [7]. There is an embedded Debian 9 (Stretch) operation system on the PS side (Processing System, CPU side). The drivers of the DPU and a popular middleware for robot ROS are also installed in the Debian 9 OS. The drivers take charge of communication between software and DPU. The PS operations, such as VLAD operations in NetVLAD and the FC layers in VO, are running over the ROS middleware. Table II shows the run time of each part of our DSLAM system.

TABLE II
RUN TIME OF EACH PART IN OUR DSLAM

	VO (T_V)	NetVLAD (T_D)	DPR, PL (T_{D_PL})	VO, PL (T_{V_PL})	DPR, PS (T_{D_PS})	VO, PS (T_{V_PS})
Time (ms)	13	422	66	3	356	340

	Without Cross-Component	With Cross-Component
Constraint	$N \times T_f > T_D + N \times T_V$	$N \times T_f > T_{D\_PS}$
N value	N = 12	N = 8

* We read the camera at 20fps, so the T_f in fig 6 is 50ms.
* We use the NetVLAD method to do DPR in this paper.

Since the NetVLAD frequency (operate once NetVLAD for every N VO frames) before cross-module scheduling is constrained by eq. (8), NetVLAD can be calculated once every 12 VO frames. We divide the running time of the DSLAM into three threads: PL, VO on PS, NetVLAD on PS, and we calculate the running time of each part, which satisfies eqs. (10) to (12) respectively. Then we find out that run-time of the NetVLAD on PS (T_{D_PS}) becomes the bottleneck of the DSLAM system and eq. (12) constrains the NetVLAD frequency (N) to 8.

The merged trajectory without cross-component scheduling is shown in fig 9(e) and the trajectory with our proposed cross-component scheduling method is shown in fig 9(c). The empirical results in section IV-C show that our cross-component scheduling method can significantly improve the DSLAM accuracy.

V. CONCLUSION

Though DSLAM can benefit from CNN, the deployment of CNN on embedded FPGAs faces significant challenges. We propose a CNN-based monocular DSLAM system deployed on a Xilinx ZCU102 MPSoC platform, and a fixed-point fine-tune method to balance the accuracy and speed of CNN-based Visual Odometry on embedded FPGA. We also explore the impact of DPR frequency on DSLAM results, and propose a cross-component scheduling method to increase the operating frequency of NetVLAD to improve DSLAM performance. The CPU operations on NetVLAD limit the speed of the DSLAM system. In future work, accelerators for the VLAD operations could be designed for higher performance.

REFERENCES

[1] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-Efficient Decentralized Visual SLAM," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2466–2473, 2018.

[2] 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.

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 1437–1451, 2017.

[4] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[5] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, and S. Song, "Going deeper with embedded fpga platform for convolutional neural network," *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016.

[6] 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 Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 22:1–22:23, Dec. 2018. [Online]. Available: http://doi.acm.org/10.1145/3283452

[7] "Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit," 2019. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

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

[9] "UltraScale MPSoC Architecture," 2019. [Online]. Available: https://www.xilinx.com/products/technology/ultrascale-mpsoc.html

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[11] "Kitti groundtruth," 2019. [Online]. Available: https://github.com/ZhangXiwuu/KITTI_GroundTruth

[12] E. Stumm, C. Mei, S. Lacroix, J. Nieto, M. Hutter, and R. Siegwart, "Robust Visual Place Recognition with Graph Kernels," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4535–4544, 2016.

[13] 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.