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 "robot", the Visual Odometry (VO) is a basic component to estimate the 6-DoF absolute pose, and for "multi", Decentralized Place Recognition (DPR) is a fundamental element to produce candidate place matches. Although some CNN-based VO and DPR methods have made significant progress in performance compared to feature-based methods, such as Depth-VO-Feat [1] and NetVLAD [2], they focus primarily on the accuracy of CNNs, yet consider little about the deployment of CNNs on embedded systems.

Since the embedded system usually only supports fixed-point CNN, we propose a pose-sensitive fixed-point finetune method for the CNN-based monocular VO, and accelerate the per-frame VO from 230ms to 10ms with similar accuracy. In addition, empirical experiments show that the frequency of DPR has a large impact on the final result of DSLAM. So we propose a cross-component pipeline scheduling method to increase the computational speed of NetVLAD from once every 12 frames to once every 8 frames, and further improve the final accuracy of DSLAM.

I. Introduction

In recent years, the intelligence of a single agent have been significantly improved. To further expand the capabilities of intelligent robots, using multiple robots simultaneously can accelerate many tasks, such as localization, exploration, and mapping. 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. [3] conclude the basic procedure of DSLAM as fig 1.

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.

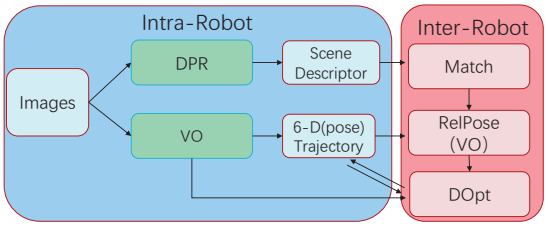


Fig. 1. DSLAM framework. VO is used to calculate the intra-robot 6-DoF absolute pose from the input frames. DPR produces a compact image representation to be communicated among robots. Match stage finds out candidate inter-robot place recognition matches. RelPose requires data from the matched robots 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.

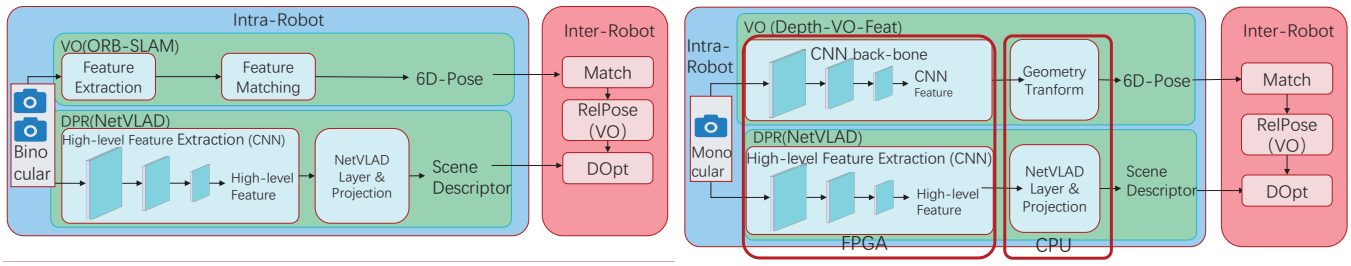
Cieslewsk et al. [3] use ORB-SLAM [4] for VO and NetVLAD [2] as the DPR component. These two algorithms both consume a large amount of computing and storage resources, and pose a huge challenge to the DSLAM in embedded systems. The DSLAM framework in [3] is illustrated in fig 2(a).

With the development of CNN, we can reconstruct the depth and pose with the absolute scale from the monocular camera directly, making monocular VO more robust and efficient. And monocular VO methods, like Depth-VO-Feat [1], make DSLAM system much easier to deploy than stereo ones. Besides, some recent advances in deep learning and large-scale labeled vision datasets have significantly improved the accuracy of place recognition (DPR), such as NetVLAD [2]. However, these previous works only focus on the improvement of network performance, neglecting the deployment of the network on embedded systems.

Although CNN-based DSLAM systems have many advantages, there are still some key issues to be solved: 1) Embedded systems usually only support fixed-point models. 2) When running multiple CNN models at the same time, the running speed of the system on embedded platforms will decrease, and the slowdown of DPR will lead to the degradation of the final DSLAM accuracy. Therefore, we build up a CNN-based monocular DSLAM system on the embedded FPGA platform with following contributions:

- To the best of our knowledge, this is the first work

[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



(a) DSLAM in [3]

(b) Our CNN-based DSLAM.

Fig. 2. (a) The previous work uses ORB feature-point-based method to estimate the trajectory from sequences of the stereo camera, and use NetVLAD [2] to do DPR. (b) Our framework adopt Depth-VO-Feat [1] in DSLAM system as the monocular VO, and we use the same method of NetVLAD in [3] to do DPR. We use the Xilinx Zynq ZU9 MPSoC hardware platform to for deployment.

to implement all components of monocular DSLAM with CNN. We deploy the system on the Xilinx Zynq ZU9 MPSoC hardware platform with DPU [5], which is an embedded CNN accelerator. The proposed DSLAM system is illustrated in fig 2(b). We also propose a two-stage framework to optimize and deploy a given DSLAM system on the embedded FPGA.

- As embedded systems impose restrictions on the size of the model, we propose a Pose-Sensitive fixed-point finetune method to divide the VO network into subsets, and adapt different finetune strategies to achieve similar accuracy to the original floating-point VO network. We schedule the fixed-point layers on DPU and the floating-point layers on CPU, so that we can accelerate VO to 10ms/frame.
- In order to improve the NetVLAD frequency, we propose a cross-component method to schedule the calculation process of VO and DPR, and complete the calculation across the PL and PS of MPSoC to make full use of the embedded platform. And we increase the calculation frequency of NetVLAD from every 12 frames to once every 8 frames.

The rest part of this article is organized as follows. Section II will give the basic idea of CNN-based VO and DPR, as well as the hardware platform. Section III will detail the implementation of our two-stage DSLAM framework. The experimental results will be given in Section IV. Section V will conclude this paper.

II. Background

A. CNN-based methods for DSLAM

As mentioned above, there are two essential components on each robot: 1) Visual Odometry (VO) and 2) Decentralized Place Recognition (DPR).

1) Visual Odometry (VO): Visual odometry estimation is the task to infer ego-motion from a sequence of images and an essential component of the DSLAM system. Some feature-based SLAM systems have enjoyed great success, like ORB-SLAM [6] and ORB-SLAM2 [4]. Recently, several studies have shown that these feature-based SLAM systems require high computing resources.

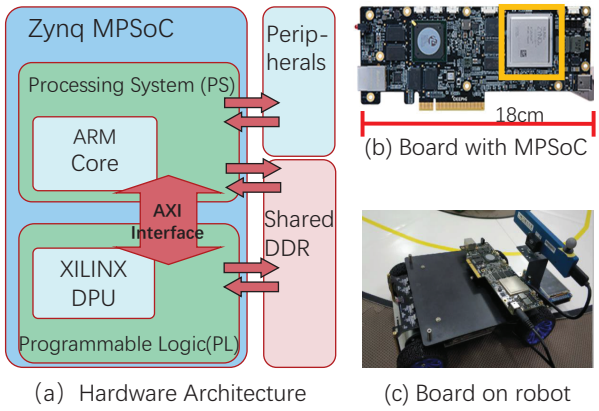


Fig. 3. Xilinx Zynq Zu9 MPSoC Platform

Fang et al. [7] illustrate that the feature extraction stage is the most computation-intensive phase, consuming >50% of CPU resources. In order to make full use of the parallelism of the embedded platform, a recent work [1] introduces CNN for end-to-end pose 6-DoF estimation.

2) Decentralized Place Recognition (DPR): The goal of place recognition or DPR is to calculate a given frame into a limited set of places. Every place can be encoded as compact code that can be easily transferred at low communication costs. Traditional location recognition methods typically convert input frames into a collection of hand-crafted feature points and local descriptors, such as SIFT [8] or ORB [4], using vectorization techniques like bag-of-words (BoW) [9]. Recent advances [2], [10] in the CNN enable powerful end-to-end model for place recognition, and NetVLAD is one of the best CNN-based methods to do DPR.

B. Hardware architecture of Zync MPSoC

The CNN-based VO and DPR components consume more than tens of billions of operations and thus are difficult to deploy on embedded systems. The Xilinx ZU9 MPSoC is a chip with ARM cores and FPGA fabric, and can be used to run CNN on embedded systems. The architecture of MPSoC is illustrated in fig 3.

The ARM cores with an embedded Linux operation

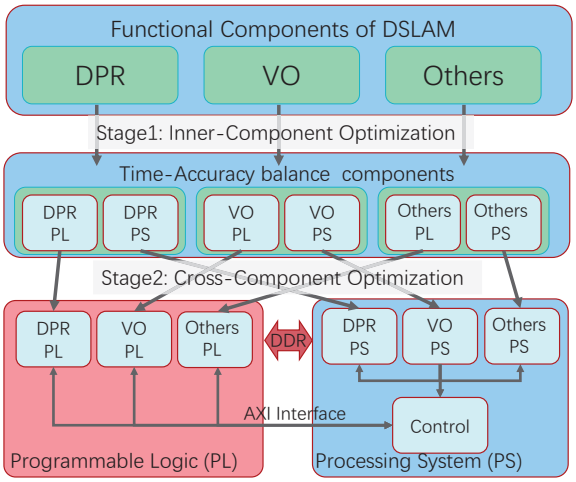


Fig. 4. The two-stage CNN-based DSLAM framework. Every CNN-based components can be split into two subnets running on PL and PS respectively can can be optimize by the inner-component optimization. The cross-component optimization can effectively schedule the components across PL and PS.

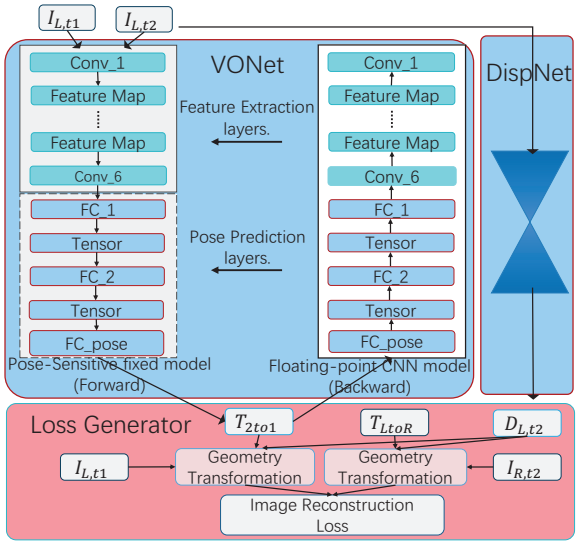


Fig. 5. Illustration of training framework for visual odometry, where T_{LtoR} is the relative camera pose transformations between left and right views. To speed up the inference, we attepmt different quantization strategies to fixed-finetune the network for VO with fixed-point feed forwarding and floating-point backpropagation.

system are called Processing System (PS). The FPGA fabric is called Programmable Logic (PL). The peripherals like camera 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 every frame of the camera. Xilinx CNN accelerator, called DPU [5], is one of the state-of-the-art accelerators and is known for its energy efficiency in running various CNN structures. We deploy the DPU accelerator on the PL side of Zynq SoC.

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

III. Methodology

In this section, we will firstly introduce the two-stage general CNN-based DSLAM framework to leverage the computation resources on embedded FPGA. Then we will take the DSLAM frame shown in fig 2(b) as an example and introduce the specific implementation of the two stages in this example.

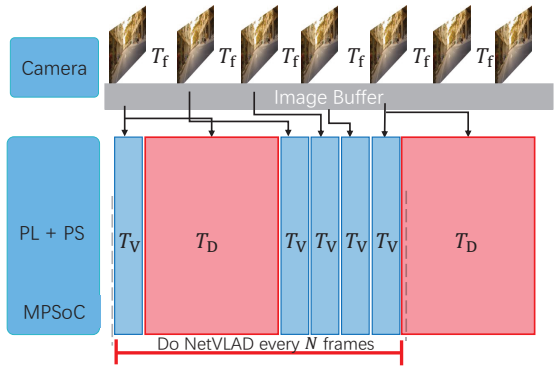
A. CNN-based DSLAM framework on FPGA

As shown in fig 4, our CNN-based FPGA framework contains two major steps: 1) inner-component optimization and 2) cross-component optimization. A DSLAM system has several CNN components such as VO, DPR, and some other components like semantic segmentation which can support the VO and DPR. Each CNN component can and should be divided into two subnets, running on PL and PS side individually. Given the definitive CNN components, the inner-component optimization stage will find out the balance between the speed and accuracy, and will produce the best split method of each component to PL subnet and PS subnet. After generating the PL and PS subnet for all CNN components, the cross-component optimization stage will schedule the components across the PS and PL in pipeline, so that we can leverage the limited on chip resources and reach the overall optimal system performance. The scheduling configuration and control unit, which will be stored and executed on the PS side, will be generated as the output of this framework.

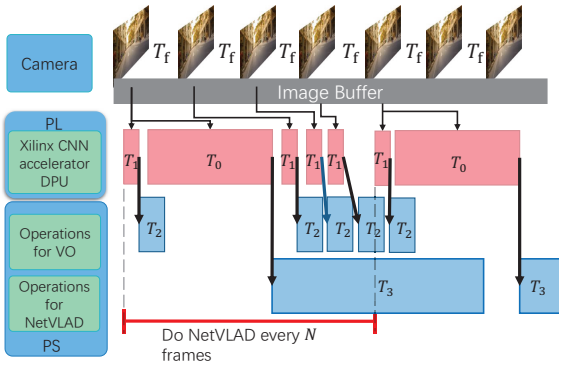
In the following subsections, based on the proposed DSLAM method in fig 2(b), we will introduce two inner-component method: Pose-Sensitive fixed-point finetune method for VO optimization and the NetVLAD implementation on FPGA for DPR optimization. The cross-component example to scheduling the Depth-VO-Feat and the NetVLAD will be detailed as an example.

B. Pose-Sensitive Fixed-Point Finetune

We adopt Depth-VO-Feat [1] in the DSLAM system to estimate the pose from the input monocular camera. The training and forwarding framework is illustrated in fig 5. Depth-VO-Feat uses image reconstruction loss as a self-supervised signal to train the convolutional neural networks and jointly trains two networks for depth (DispNet) and odometry estimation (VONet) without external supervision. In order to run our networks efficiently on the FPGA platform, we use fixed-point arithmetic units on hardware. Previous works use the fixed-point finetune method [11] to run CNN in an 8-bit fixed-point arithmetic system without accuracy decline in many applications, such as classification [12] and object detection [11]. Fixed-point fine-tune uses the



(a) Scheduling without pipeline. There are only two threads: Camera read and computation.



(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.

fixed-point number representation in the feed forward phase and keep floating-point number representation for backpropagation, and both weights and data will be re-quantized after each backpropagation. We find that if the entire VONet is treated as a fixed-point model, the accuracy drops dramatically because VO task requires much higher computational precision than other applications.

Pose-Sensitive fixed-point finetune method can explore the run time of VO at different accuracy. We split the VONet into 2 subnets: 1) fixed-point layers for feature extraction. 2) floating-point layers for pose prediction. We propose the Pose-Sensitive fixed-point finetune method, that trains the feature extraction layers with fixed-point finetune method, and trains the pose prediction layers with floating-point.

The Pose-Sensitive fixed-point finetune method will construct some possible network split plans, improve the accuracy under each split plan, and evaluate the run time on the embedded system. With the help of Pose-Sensitive fixed-point finetune, we can balance the accuracy and the execution time. The experiment results of the accuracy and the run time will be exhibited in section IV.

C. DPR on FPGA with NetVLAD

The CNN-based DPR methods give the global descriptor of a camera frame in a two-step manner: 1) Firstly, a CNN encoder fetches the high-level feature map. 2) A vectorization component that aggregates the feature map into a shot global descriptor. In the original NetVLAD [2], the feature extraction encoder is a typical CNN named VGG-16 [13], and the vectorization component consists of VLAD layers. The PCA method can drastically reduce the output dimension. The previous works [3], [14] show that the 128-D code after PCA is plenty for DSLAM. The VLAD layer and the PCA projection require floating-point number, so we implement the NetVLAD layer and the projection on the PS side of Zynq MPSoC.

Unlike the fixed-point finetune method used for pose estimation, we simply analyze the dynamic range of the weight and intermediate feature map of each CNN layer, and figure out the optimal decimal point position for each layer respectively to minimize the truncation error of each layer.

Since the DPU dose not support multi-threading, we can only schedule the whole module of VO and NetVLAD serially. fig 6(a) illustrate the serial schedule strategy. 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. We do VO every input frame and do NetVLAD every N frames. The N is constrained by eq. (1).

$$N \times T_f > T_D + N \times T_V \tag{1}$$

D. Cross-Components Scheduling

The time consumption of NetVLAD and VO is unbalanced. We do pipeline optimization to schedule the two components on Zynq MPSoC efficiently. The pipeline is illustrated in fig 6(b). Both NetVLAD and VO time can be divided into two parts: $T_D = T_0 + T_3, T_V = T_1 + T_2$.

The CNN time for NetVLAD and VO is T_0 and T_1. The other computation time cost on PS for VO and NetVLAD is T_2 and T_3. The frequency of NetVLAD can be expressed as do NetVLAD every N frames $(NetVLAD/\sharp frames)$.

Considering the thread on PL, the time constraint is given as eq. (2).

$$N \times T_f > T_0 + N \times T_1 \tag{2}$$

The thread for VO on PS constrains the NetVLAD frequency as eq. (3).

$$N \times T_f > T_0 + T_1 + (N-1) \times T_2 \tag{3}$$

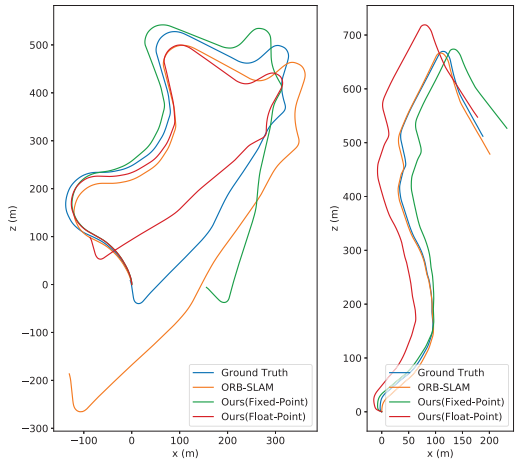
The PS part of current NetVLAD should finish before computing the PS part of next NetVLAD frame. This constraint can be written as eq. (4).

TABLE I

Visual odometry (VO) results on test sequences (09, 10)

Method	Quant. Strategy Fixed Part	Float Part	Seq. 09 t^1_{err}	r^1_{err}	Seq. 10 t_{err}	r_{err}	run time $(ms/frame)$
ORB-SLAM	-		15.30	0.26	3.68	0.48	230 [2]
Depth-VO-Feat[1]	-		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 on the PS part of Xilinx ZU9 MPSoC. The run time of reconstructing absolute scale of VO is included.



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

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

$$N \times T_f > T_3 \tag{4}$$

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 Pose-Sensitive Fixed-Point Finetune

We evaluate our VO approach with DPU on KITTI dataset [15]. 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 [1] 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 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 [4], as the baseline.

We use the Pose-Sensitive fixed-point finetune method to balance the speed and accuracy. We use different strategies to split the VONet, and finetune the VONet with the same training superparameters. The initial learning rate is $1e^{-5}$ and the number of finetune iterations is 240000, which is sufficient to train Depth-VO-Feat from random initialization. We snapshot the network weights every 5000 iteration, and list the accuracy of the best snapshot in table I. The Quant. Strategy indicates the configuration of Pose-Sensitive fixed-point finetune. The Fixed Part is the feature extraction layers on PL, and the Float Part is the pose prediction layers on PS. Because the computation volume of FC layers is much less than that of CONV layers, we can split all of the FC layers into pose prediction network, significantly improving the accuracy by consuming little extra time.

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

We evaluate our VO and NetVLAD implementation simutaneously on a Xilinx ZU9 MPSOC. Table II shows the run time of each part of our DSLAM system.

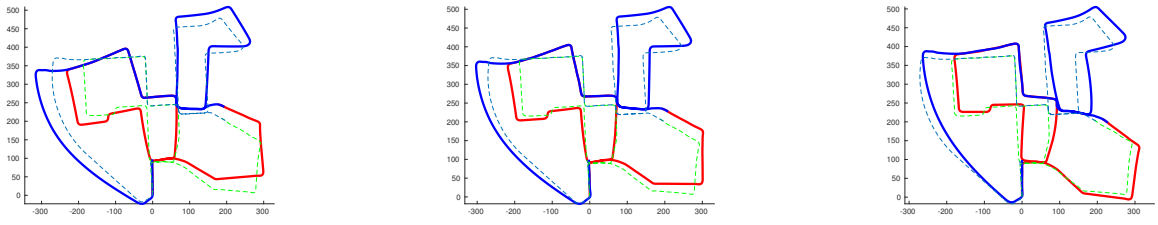
TABLE II

Run time of each part in our DSLAM

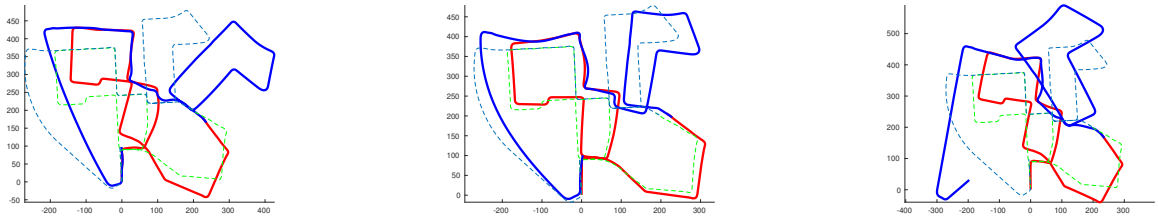
	VO (T_V)	DPR (T_D)	DPR, PL (T_0)	VO, PL (T_1)	VO, PS (T_2)	DPR, PS (T_3)
Execute time (ms)	13	422	66	3	10	356

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

Since the NetVLAD frequency (N) before cross-module scheduling is constrained by eq. (1), NetVLAD can be calculated once every 12 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. (2) to (4) respectively. Then we find out that run-time of the NetVLAD on PS (T_3) is the bottleneck of DSLAM system and eq. (4) constrains the NetVLAD frequency (N) to 8.



(a) NetVLAD/1frame. (b) NetVLAD/4frames. (c) NetVLAD/8frames.



(d) NetVLAD/10frames. (e) NetVLAD/12frames. (f) NetVLAD/14frames.

Fig. 8. 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.

The effect of NetVLAD frequency on the DSLAM result will be further discussed in section IV-D.

C. NetVLAD with DPU

We evaluate the NetVLAD performance on the loop-closure dataset based on KITTI [16]. 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 $6m$.

The precision-recall 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 9. We take sequence 00 as the test sequence, and achieve similar accuracy with the floating-point model of the same output dimensions.

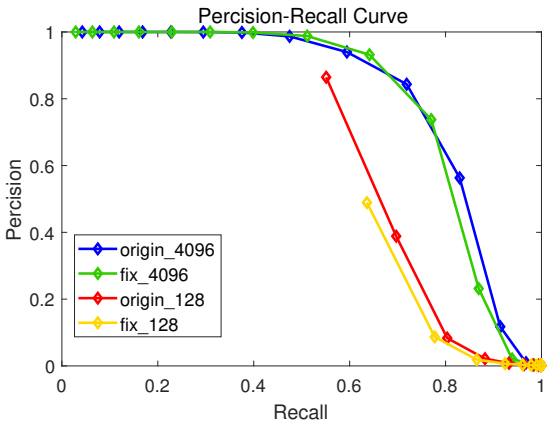


Fig. 9. ROC curves on sequence 00.

D. DSLAM Evaluation

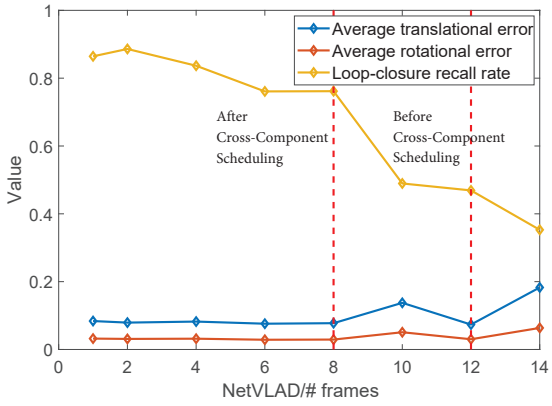


Fig. 10. ATE, ARE and LCR curves on sequence 00. For ATE and ARE, lower is better, and for LCR, higher is better. Lower $NetVLAD/\sharp frames$ means higher NetVLAD frequency. The NetVLAD frequency is $NetVLAD/12frames$ for serial scheduling and $NetVLAD/8frames$ for Cross-Component scheduling.

We evaluate our DSLAM system on KITTI odometry dataset. Firstly we divide the KITTI sequence 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 [3] to simulate the DOpt and merge the two trajectories. The DOpt method used in this work is proposed in [17].

We run each subsequence on the Xilinx MPSoC ZU9 platform to get the results of VO and DPR. The merged trajectories before and after cross-module scheduling are shown in figs 8(c) and 8(e). The results of other NetVLAD frequencies, illustrated as figs 8(a), 8(d) and 8(e), are simulated with the fixed-point CNN model

on desktop computer.

Tranditional indicators, such as average translational error (ATE) and average rotational error(ARE), can not demonstrate the performance of trajectory merging in DSLAM, so we propose a new metric called loop-closure recall rate (LCR) to evaluate the DSLAM system. LCR indicates the success rate of loop-closure detection on the merged trajectory, which can reflect the performance of trajectory merging. LCR is calculated by the following formula:

$$LCR = \frac{\eta_{merged\ trajectory}}{\eta_{groundtruth}}$$

, where η is the number of frame pairs with successful place recognition. In our experiments, when the Euclidean distance between one point and the other point on the trajectory is less than 6 meters, just like [18], we think that this is the place that has been reached before, that is, the location matches, forming a loop.

Fig 10 shows that with the increase of the NetVLAD frequency, there is a downward trend on ATE and ARE. However, the ATE and ARE strongly depend on the loop-closure result, and a single accidentally detected loop-closure will dramatically change the trajectory. That's why fig 8(e) performers better than fig 8(d) with a lower NetVLAD frequency. The LCR can efficiently indicate the trajectory merging performance: as the NetVLAD frequency increases, the LCR curve also increases.

From fig 8 and fig 10, we can conclude that the NetVLAD frequency higher than $NetVLAD/8frames$ reach a similar high level performance in different indicators. Continuous improvement of the NetVLAD frequency from $NetVLAD/8frames$ has limited effect on the final DSLAM performance. Our cross-component scheduling method can improve the NetVLAD frequency from $NetVLAD/12frames$ to $NetVLAD/8frames$, and thus significantly evolve the DSLAM performance.

V. Conclusion

Though DSLAM can benefit from CNN, the deployment of CNN on embedded systems faces big challenges. We propose a CNN-based monocular DSLAM system deployed on Xilinx ZU9 MPSoC, and a Pose-Sensitive fixed-point finetune method to balance the accuracy and speed of CNN-based Visual Odometry on embedded FPGA. We also explore the impact of NetVLAD frequency on DSLAM results, and propose a cross-component scheduling method to increase the operating frequency of NetVLAD to improve DSLAM performance.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147–1163, 2015. [Online]. Available: https://doi.org/10.1109/TRO.2015.2463671

[7] W. Fang, Y. Zhang, B. Yu, and S. Liu, "Fpga-based orb feature extraction for real-time visual slam," 2017 International Conference on Field Programmable Technology (ICFPT), pp. 275–278, 2017.

[8] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, vol. 60, pp. 91–110, 2004.

[9] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," IEEE Transactions on Robotics, vol. 28, 2012.

[10] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," Proceedings of the IEEE International Conference on Computer Vision, 2017.

[11] 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

[12] 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.

[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.

[14] P.-E. Sarlin, F. Debraine, M. Dymczyk, R. Siegwart, and C. Cadena, "Leveraging Deep Visual Descriptors for Hierarchical Efficient Localization," arXiv preprint arXiv:1809.01019, 2018.

[15] 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.

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

[17] 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.

[18] X. Zhang, L. Wang, Y. Zhao, and Y. Su, "Graph-based place recognition in image sequences with cnn features," Journal of Intelligent & Robotic Systems, pp. 1–15, 2018.