# Data-Efficient and Hardware-Efficient Decentralized Monocular Visual SLAM

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

***Abstract*— Decentralized visual simultaneous localization and mapping (DSLAM) can share location and environmental information between robots and is the essential task for many multi-robot applications. For *"robot"*, the Visual Odometry (VO) is a basic component to estimate the 6-D absolute pose, and for *"multi"*, Decentralized Place Recognition (DPR) is a basic component to produce candidate place recognition matches. With the development of Convolutional Neural Network(CNN), both the VO and DPR can be significantly improved in performance, such as Depth-VO-Feat[1] and NetVLAD[2]. However, previous works concentrate on the accuracy of the CNNs, yet consider little about the deployment CNNs on the embedded system.**

**Because the embedded system usually support fixed-point CNN only, we propose a pose-sensitive fixed-point finetune method for the CNN-based monocular VO, and accelerate the VO from 230ms to 10ms with the similar accuracy. As the frequency of DPV influences the final result of DSLAM, we propose a module-mixed pipeline scheduling method to calculate the NetVLAD every 4 input frames from every 6 frames, and improve the final accuracy of DSLAM.**

## I. Introduction

In recent years, the capabilities of a single agent have been significantly improved. To further expand the capabilities of intelligent robots, using several robots can accelerate many tasks, such as localization, exploration, and mapping. Decentralized visual simultaneous localization and mapping (DSLAM) can share location and environmental information between robots and is the essential task for many multi-robot applications. [3] concludes the basic procedure as fig 1

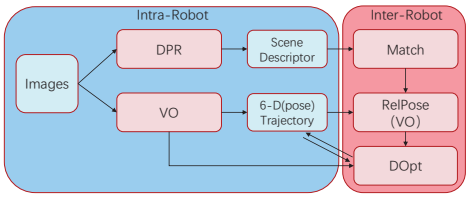



Fig. 1. DSLAM framework[3]. **VO** is used to calculate the intra-robot 6-D absolute pose from the input frames. **DPR** produces a compact image representation that communicate among robots. **Match** stage finds out candidate inter-robot place recognition matches. **RelPose** requires data from the matched robots and establishes relative poses between the robots trajectories. **DOpt** obtains the trajectories, intra-robot pose measurements from VO and inter-robot relative poses from RelPose, and updates the trajectories.

[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

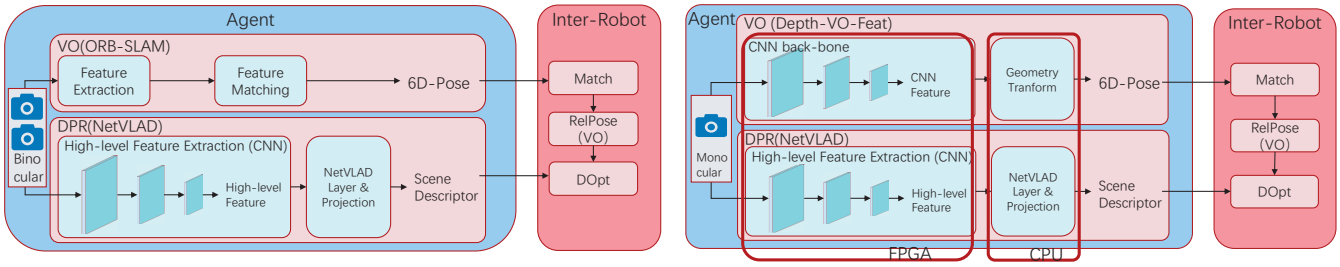
There are five components: DPR, VO, Match, RelPose and DOpt. DPR and VO are intra-robot operations, and require high computation resources on embedded system. Match, RelPose and DOpt are inter-robot operations, and consume most of the communication of DSLAM sytem. The RelPose components can and should depend the VO components since it can benefit from re-using the data and computation resources of VO.

The previous work [3] uses ORB-SLAM[4] as the VO and NetVLAD[2] as the DPR component. These two algorithms both consume a large amount of computation and storage, and pose a great challenge to DSLAM on the embedded system. The DSLAM frame in [3] is illustrated in fig 2(a).

Monocular systems are much easier to deploy than binocular systems. The transitional monocular feature point based VO should use complex depth reconstruction methods to compute the absolute scale of the pose scale[5], leading to speed-down. 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[1]. ecent advances in deep learning and the availability of large labelled visual datasets have significantly improved the accuracy of place recognition, such as NetVLAD[2]. However, previous works concentrate on the accuracy of the CNNs, yet consider little about the deployment CNNs on the embedded system.

Though DSLAM system can benefit from the development of CNN, the fully CNN-based DSLAM system faces some key issues: 1) The embedded system usually support fixed-point CNN. 2) The speed will decline in the embedded system when running several CNN models simultaneously, the speed-down of DPR will lead to the decline of the final DSLAM accuracy. Therefore, we build up a CNN-based monocular DSLAM system on embedded FPGA platform, with following contributions:

- To the best of our knowledge, this work is the first to implement all components of monocular DSLAM with CNN. We deploy the system on Xilinx Zynq MPSoC hardware platform with DPU [6], which is an embedded CNN accelerator. The proposed DSLAM framework is illustrated in fig 2(b).
- As the embedded system usually support fixed-point CNN, we propose a pose-sensitive fixed-point fine-tune method to make the feature extraction layers fixed-point and remain the pose prediction layers floating-point, reaching the same accuracy with the original floating-point VO network. We schedule the fixed-point layers



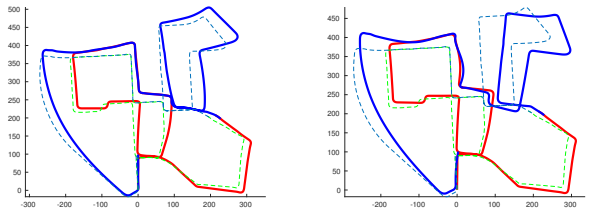

(a) DSLAM in [3]

(b) Our CNN-based DSLAM.

Fig. 2. (a) The previous work uses ORB feature-point based method to estimate the trajectory from the sequence 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 Xilinx Zynq MPSoC hardware platform to for deployment.

on DPU and the floating-point layers on CPU, so that we can accelerate the VO to 10ms.

- To increase the NetVLAD frequency, we propose a cross-components scheduling method to scheduling the computation flow across VO and DPR, as well as across the PL and PS of MPSoC to make full use of the embedded platform. We calculate the NetVLAD every 4 input frames from every 6 frames, improving the final accuracy of DSLAM. We also propose a new indicator called loop-closure recall (LCR), which indicates the remaining rate of loop-closure after trajectory merging, to evaluate the performance of trajectory merging. The output result of DSLAM with different NetVLAD frequency is illustrated in fig 3. The tranditional average trajectory error (ATE) can not indicate the performance of trajectory merging in DSLAM.



(a) NetVLAD/8 frames. ATE=9 (b) NetVLAD/12 frames. ATE=9

Fig. 3. The DSLAM result of two robots(red and blue, the dashed is the ground truth the robots). (a) When do NetVLAD every 8 frames, the trajectories can be merged correctly. (b) When do NetVLAD every 12 frames, the trajectories can not be commendably merged. However, the ATE of these two results is similar.

## II. Background

### A. CNN based methods in DSLAM

As described before, there are two essential components on each agent: 1) Visual Odometry (VO) and 2) Place Recognition (DPR).

*1) Visual Odometry (VO):* Visual odometry estimation is the task to infer ego-motion from a sequence of images and is an essential component in the SLAM system. Some feature-based SLAM systems have enjoyed great success, like ORB-SLAM[7] and ORB-SLAM2[4]. Recently, several studies have shown that these feature-based SLAM systems require high computing resources. [8] shows that the feature extraction stage is the most computation-intensive, consuming over 50% of the CPU resources.

As FPGA is one of the most promising platforms as the accelerator for VO, the SLAM system on FPGA has become a hot research topic. However, FPGA-accelerated feature extraction still consumes a lot of time and computing resources, which cannot be deployed simultaneously with an FPGA-accelerated neural network.

*2) Place Recognition (DPR):* The goal of place recognition is to calculate a given frame into a limited set of places. Each place can be encoded as a concise code which can be easy transferred with low communication cost. Traditional place recognition method usually translate the input frame as the aggregation of handcrafted feature point and local descriptors, like SIFT [9] or ORB [4], using vectorization techniques like bag-of-words (BoW) [10] or vector of locally aggregated descriptors(VLAD) [11].

Recent advances in the deep learning and the convolution neural network (CNN) enable powerful end-to-end mode for place recognition [12], [2], and the NetVLAD method is one of the most accurate methods based on CNN. The NetVLAD algorithm based on VGG-16 model [13] consumes more than $80G$ operations for a single $300 \times 300$ input image (each operation means addition or multiplication). It is very challenging to deploy the NetVLAD on a traditional embedded hardware platform.

### B. Hardware architecture of Zync MPSoC

The Xilinx Zync MPSoC is a chip with ARM cores and FPGA fabric. The ARM cores with an embedded Linux operation system are called Processing System (PS). The FPGA fabric is called Programmable Logic (PL). The peripherals like camera and communication unit (WiFi or others) are accessible with PS. The high-bandwidth on-chip AXI interface is used to communicate between PS and PL. PS and PL can also share the DDR to transfer a large volume of data such as each frame of the camera. Deephi CNN accelerator, which is called DPU [6], is one of the state-of-the-art accelerators and is famous for high energy efficiency on various CNN structure. We deploy the accelerator on the

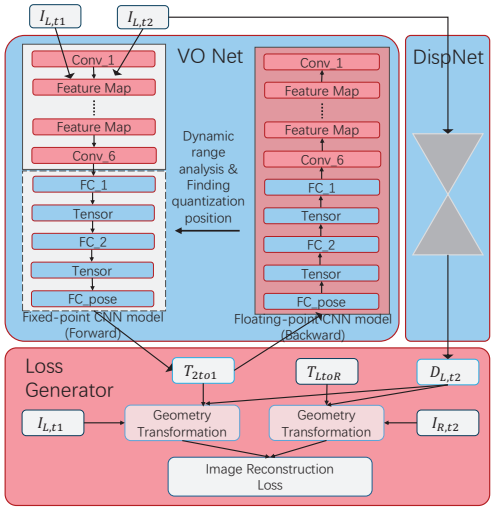



Fig. 4. Illustration of training framework for visual odometry, where $T_{LtoR}$ is the relative camera pose transformations between left and right views. To speed up the inference, we attepmt different quantization strategies to fixed-finetune the network for VO with fixed-point feed forwarding and floating-point backpropagation.

PL side of Zynq SoC with the help of DPU.

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

Our hardware-software co-design DSLAM system contains two essential improvements in the pose estimation and place recognition tasks. As illustrate in fig 2(b), both of these two components are divided into two stages: 1) CNN front end to extract features which is deployed to the DPU on PL and 2) Algebraic operations to present final results which are deployed on the PS ARM cores. To make full use of the Zynq MPSoC, we optimize the data follow for both of these components.

### A. Pose Estimation

We adopt Depth-VO-Feat [1] in DSLAM system to estimate the pose from the input monocular camera. Monocular visual SLAM is a key issue in the field of robotics, while there are two challenging problems: 1) it is difficult and expensive to obtain accurate labeled data, 2) the methods that use monocular sequences in training always suffer from the scale-ambiguity problem, i.e., the actual scale of translations is missing, and only the direction is learned. In Depth-VO-Feat [1], we use image reconstruction loss as a self-supervised signal to train the convolutional neural networks and jointly train two networks for depth and odometry estimation without external supervision, which can be used independently in the testing phase. Besides, to fix this scale-ambiguity issue, we use stereo sequences in the training phase and monocular sequences in the testing phase. With the known spatial relationship between the left and right cameras, our neural networks can learn the real world scale. Moreover, we use depth smoothness loss to encourage the predicted depth to be smooth, which demonstrated success in prior works. Then the final loss becomes:

$$L = \lambda_{ir} L_{ir} + \lambda_{ds} L_{ds} \tag{1}$$

where $L_{ir}$ and $L_{ds}$ are image reconstruction loss and depth smoothness loss respectively, $\lambda_{ir}$ and $\lambda_{ds}$ are the loss weightings for each loss term. The training framework is illustrated in section III.

In order to run our networks efficiently on the FPGA platform, we use fixed-point arithmetic units in the hardware to replace the floating-point number format in GPU and CPU. Many previous works have shown that 8-bit quantization for weights and featuremaps can make the networks run faster on FGPA. Here we adopt the fixed-point finetune method in [14], in that we use the 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. The fixed-point method will lead to a slight accuracy loss of the model, and the performance of the fixed-finetuned Depth-VO-Feat will be shown in detail in section IV.

### B. Place Recognition

The place recognition method provide the encoded vector transferred to the central agent for inter-robot place matching. As described in section II, CNN has achieved significant improvements in place recognition tasks, and NetVLAD [2] is one of the most impressive methods. The CNN-based place recognition 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. The VLAD layer [2] is a recently proposed plug-and-play operation that greatly improves the performance of place recognition. In the original work with the VLAD layer [2], the feature extraction encoder is a typical CNN named VGG-16 [13]. The output dimension of original NetVLAD is usually tens of thousands, which is very difficult to be stored on the embedded system, not to mention in the communication-constrained environment. The PCA and the projection method can drastically reduce the output dimension. The previous works[3] show that 128 dimension is plenty for DSLAM. The data flow and operations of the NetVLAD layer and the projection are complex and require the floating-point number, which cannot be supported with Deephi DPU. 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. The training procedure with huge non-public datasets is very complicated, and also we cannot finetune the NetVLAD model because of the lack of training data. 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

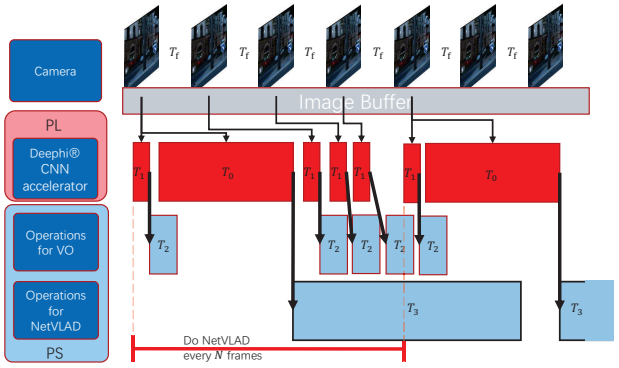



Fig. 5. Scheduling pipeline. There are four threads: Camera read, Deephi core at PL, PS Operations for VO, and PS Operations for NetVLAD.

to minimize the truncation error of each layer. This method is proposed in [15] and is used in many tasks such as image classification and image detection.

### C. Parallel Scheduling for VO and NetVLAD

The time consumption of NetVLAD and VO is imbalanced. We do pipeline optimization to schedule the two components on Zynq MPSoC efficiently. The pipeline is illustrated in fig 5. The interval time for reading camera is $T_f$. The CNN time for NetVLAD and VO is $T_0$ and $T_1$. The computation time cost on PS for VO and NetVLAD is $T_2$ and $T_3$. We do VO every input frame and do NetVLAD every $N$ 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 NetVLAD should finish before computing the PS part of next NetVLAD frame. This constraint can be written as eq. (4).

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

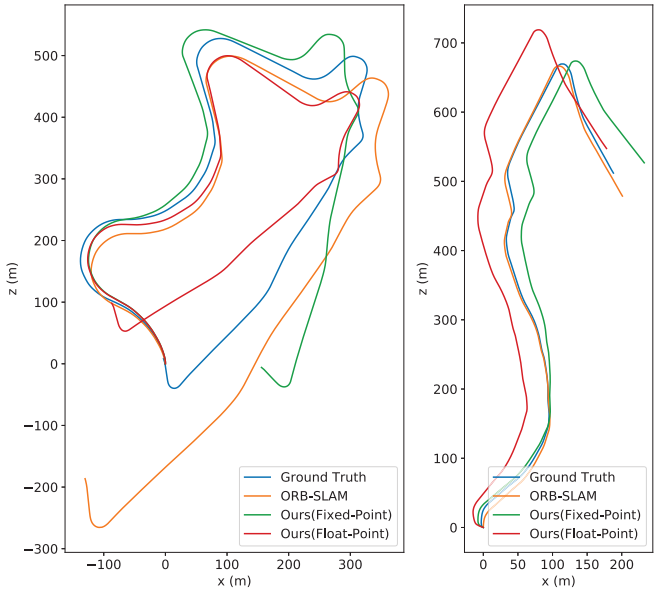
The execution time of our design will be given in section IV.

## IV. Experiments

The speed and the accuracy of our proposed hardware-software co-design DSLAM will be evaluated in this section.

### A. VO with pose-sensitive fixed-point fine-tune

We evaluate our VO approach with DPU on KITTI dataset [16]. The dataset contains 11 labeled video sequences with stereo pairs, with original image size at $1242 \times 375$ pixels. In our design, we resize the image to $608 \times 160$ as the original Deepth-VO-Feate does [1].




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

Fig. 6. Qualitative evaluation of odometry on the KITTI Odometry test sequences (09, 10).

At the fixed-finetune procedure, we use the stereo pairs in sequences 01 to 10 as the training set. The training super parameters in eq. (1) are $[\lambda_{ir}, \lambda_{fr}, \lambda_{ds}] = [1, 0.1, 10]$. At the evaluation procedure, we use the left-eye input images of the stereo pairs of the 00 sequence of KITTI as the test set.

We evaluate the VO on the sub-sequences at length of [100,200,...,800] and report the average translational and rotational errors for the testing sequence 09 and 10 in table I. The comparison of the estimated trajectory for the methods is illustrated in fig 6.

Because of the fixed-point number used in our design, which sacrifices the precision of the CNN, the VO accuracy is not as good as the previous works [4], [1]. On the other hand, our VO can calculate the 6-D pose within $20ms$ on a resource-constrained embedded platform.

### B. Run-Time

We evaluate our DSLAM on two intelligent cars which are controlled by the Deephi Aristotle board with a ZU9 MPSOC. Table II shows the run time of each part of our DSLAM system. The pipeline of scheduling of these parts is shown in section II.

Substituting the run-time of each part into eqs. (2) to (4). We find out that the run-time of NetVLAD on PS ($T_3$) becomes the bottleneck of the system and eq. (4) constrains the NetVLAD frequency ($N$) to 8.

We calculate the relative 6-D pose between every two successive frames and the NetVLAD code every 8 frames in the following experiments.

### C. NetVLAD with DPU

We evaluate the NetVLAD performance on the loop close dataset for KITTI[17]. The dataset labels the ground truth of loop closure for these sequences based on the metric

TABLE I

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

| Method | Quant. Strategy Fixed Part | Float Part | Seq. 09 $t_{err}$ | Seq. 09 $r_{err}$ | Seq. 10 $t_{err}$ | Seq. 10 $r_{err}$ | run time $(ms/frame)$ |
|---|---|---|---|---|---|---|---|
| ORB-SLAM | - | | 15.30 | 0.26 | 3.68 | 0.48 | |
| 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 | 12 |

* $t_{err}(\%)$ is the average translational drift error. $r_{err}(^\circ/100m)$ is average rotational drift error.

TABLE II

RUN-TIME OF EACH PART IN OUR DSLAM

| | NetVLAD, PL $(T_0)$ | VO, PL $(T_1)$ | VO, PS $(T_2)$ | NetVLAD, PS $(T_3)$ |
|---|---|---|---|---|
| Execute time (ms) | 66 | 3 | 10 | 354 |

* We read the camera at 20fps, so the $T_f$ in fig 5 is 50ms.

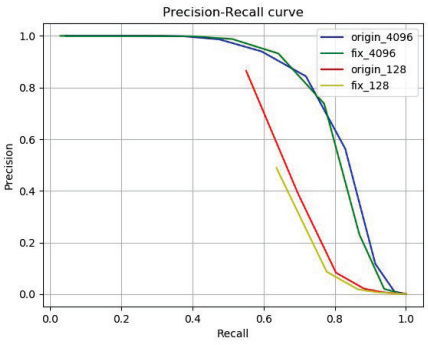



Fig. 7. ROC curves on sequence 00.

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 7. We use sequence 00 as the test sequence.

Our NetVLAD with DPU performs similarly to the original NetVLAD with the same output dimension but runs much faster with the help of the acceleration techniques on FPGA and architecture design. Even if there are sightly accuracy reduction on the ROC curves, the following experiment shows that the slight drawback would not make the overall DSLAM performance decline.

### *D. DSLAM Evaluation*

With the help of our hardware-software co-design VO and place recognition components on the embedded system, we can build up the DSLAM system like the previous work [3]. However, unlike the feature point based method in previous [3], our approach does not necessarily compute the feature point for each frame. The previous work [3] transfers the ORB feature points among agents to do inter-robot pose estimation and trajectory merging.

A possible way to solve the inter-robot pose estimation is to calculate the feature points and descriptors of the corresponding robot and frames when the inter-robot loop closure occurs. However, on the one hand, the traditional feature point detection and description method cannot get the absolute scale from a single image. On the other hand, it is resource consuming to do feature point algorithms like SIFT[11] and ORB[4]. The agents may stop and solve the feature points. We directly transfer the down-sampled $608 \times 160$ image among the agents and use the same method as the proposed VO with DPU to do the inter-robot pose estimation. The result of the merged trajectory is illustrated in fig 8.

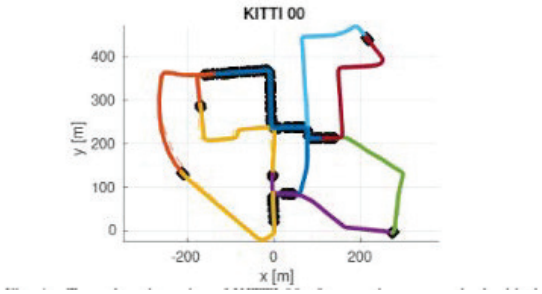



Fig. 8. ??? The sub-trajectory of each agent and the merged trajectory.

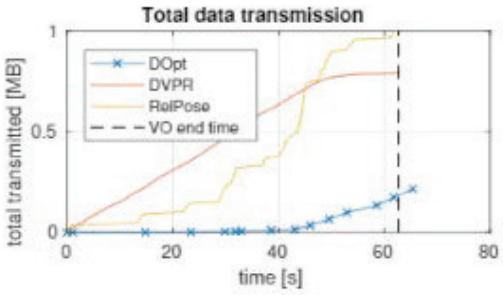



Fig. 9. ??? The total communication traffic of the propsoed DSLAM

There are three kinds of data need to be transferred

among the agents and the server: 1) the VO results of each frame (VO), 2) the NetVLAD results of every 8 frames (NetVLAD), 3) the image need for inter-robot relative pose estimation (RelPose).

At the beginning of the task, there is no inter-robot loop closure detected, so there is no data traffic for RelPose. When inter-robot loop closure occurs, the data traffic for RelPose increases rapidly. The previous work[3] also faces this problem though it uses feature point for inter-robot relative pose estimation.

The total communication traffic of the proposed DSLAM system on KITTI 00 is increasing with time and is shown in fig 9.

## V. Conclusion

We propose a hardware-software co-design DSLAM system with the help of Xilinx Zynq MPSoC and Deephi DPU. We optimize two essential components of the DSLAM system, 1)Visual Odometry(VO) and 2) Place Recognition on the embedded system. From the aspect of calculation, the vectorization and projection operations on the PS side of Zynq MPSoC needed by NetVLAD is the bottleneck in doing more frequent place recognition. From the aspect of communication, the data traffic for inter-robot relative pose estimation consumes the most communication resources. The accelerator for vectorization based on FPGA and more data-efficient inter-robot pose estimation method could be researched in future work.

## Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant No.61874156)

## 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] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2609–2616.

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

[7] 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

[8] 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.

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

[10] 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.

[11] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, 2010.

[12] 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.

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

[14] 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

[15] 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.

[16] 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.

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