

Data-Efficient and Hardware Decentralized Monocular Visual SLAM

Jincheng Yu¹ and Feng Gao¹

Abstract—Decentralized simultaneous localization and mapping (DSLAM) is essential to a multi-robot system, especially in environments lacking absolute positioning equipments like GPS. Visual based SLAM is a widely adopted solution in industry for its low cost and high flexibility. There are two essential components need to be efficiently deployed on each agent: 1) Visual Odometry(VO) and 2) Place Recognition. However, both of these components require intensive computation and storage on embedded system. The place recognition task is usually done with CNN based methods. We adopt CNN as the VO to provide 6-D pose between different frames, for both intra-robot or inter-robot. Thus we can use the CNN accelerator based on FPGA to execute these two components. In this work, we propose a hardware-software co-design DSLAM framework and use embedded FPGA to accelerator these two components.

We evaluate our framework on the hardware platform Xilinx ZU9 SoC and we can perform DSLAM in real time on each agent. We also evaluate our system on publicly available dataset.

I. INTRODUCTION

In recent years, with the development of the hardware and algorithms, the capabilities of a single agent have been greatly improved. To further expand the capabilities of intelligent robots, using several robots can accelerate many tasks, such as localization, exploration, and mapping. As simultaneous localization and mapping(SLAM) is an essential component in many tasks, it is important to do SLAM across different robots in many multi-agent applications. The camera is a widely used sensor in SLAM for its rich information and low cost. However, in many scenarios, communication is limited, so that there is no server or an agent can stably collect all of the visual data from each robot.

Therefore, to reduce communication requirements, the previous work [1] proposes a data-efficient decentralized SLAM(DSLAM) system. The DSLAM frame in [1] is illustrated in fig 1(a). It makes improvements in three typical components in the DSLAM system: 1) Using ORB-SLAM [2] in stereo configuration as the visual odometry (VO) algorithm which provides basic intra-robot pose estimation. 2) Using NetVLAD [3] algorithm to do place recognition which relates the current observation to previous scenes and other robots. 3) Using distributed Gauss-Seidel algorithm [4] as the optimization back-end which optimizes the intra-robot position and fuses the inter-robot locations and maps. Each agent executes the ORB-SLAM which contains three steps for each input frame: feature extraction, feature matching and RANSAC. The NetVLAD method can encode the camera frame to a short vector which can be transformed to the server or a central agent with low communication cost.

However, both ORB-SLAM and NetVLAD require tremendous computation and storage resources, and thus, the deployment of DSLAM on embedded system is challenged by the limited resources and power supply.

Though NetVLAD consumes huge computation, with the development of FPGA accelerators, we use the embedded CNN accelerator on FPGA [5] to perform NetVLAD for each frame. We also notice that there are also some previous works regression the 6-D pose directly from the input stereo camera [6] or monocular camera [7], [8], With the development of CNN. We adopt Depth-VO-Feat [8] in DSLAM system to estimate the pose from the input monocular camera. Because Depth-VO-Feat is trained with stereo input frames and inferenced with monocular camera, the CNN method can provide absolute scale from monocular camera, and also be accelerated with our CNN accelerator. Thus we do not need to execute ORB-SLAM on embedded CPUs.

The proposed DSLAM framework is illustrated in fig 1(b). To make the DSLAM system more energy efficient and hardware friendly, we propose a novel hardware-software co-design DSLAM framework with the following contributions:

- We implement NetVLAD on an embedded SoC platform with CPU cores and FPGA fabric.
- We use an end-to-end CNN based method to estimate the 6-DoF pose between intra-robot successive frames and matched scenes between different robots.
- We demonstrate that our proposed hardware-software co-design decentralized SLAM system can achieve a similar accuracy with the current state-of-the-art DSLAM system without increase of communication.

The rest part of this article is organized as follows. Section II will give the basic idea of CNN based methods and the hardware architecture of embedded FPGA. Section III will detail the implementation of our hardware-software co-design DSLAM system. The experiment result will be given in Section IV. Section V will conclude this paper.

II. BACKGROUND AND MOTIVATION

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.

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 SLAM system. Some feature-based SLAM systems have enjoyed great success, like ORB-SLAM[9] and ORB-SLAM2[2]. Recently, several studies have shown that these feature-based SLAM systems require

¹Electronic Engineering Department, Tsinghua University, Beijing, China
yjc16@mails.tsinghua.edu.cn

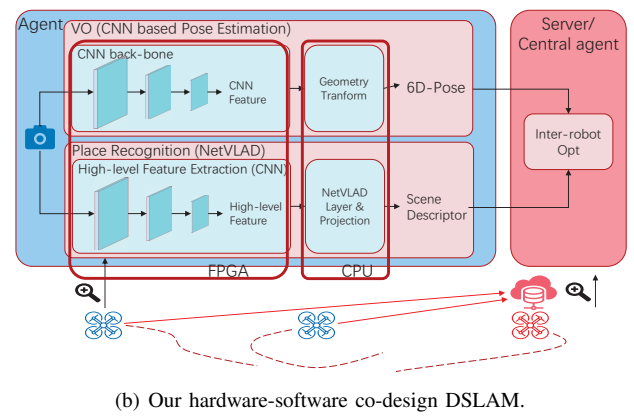
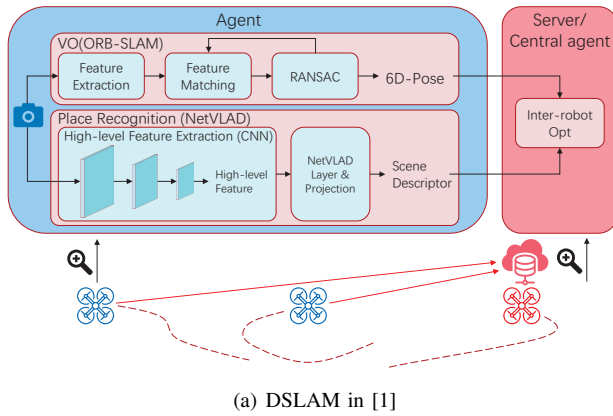


Fig. 1. Overview of the DSLAM in [1] and our hardware-software co-design DSLAM. Each agent (blue drones) will send the result of 6-D pose estimation and scene descriptor to a server or a central agent (red server or drone in figure) to do inter-robot place recognition and optimization. We use CNN instead of feature points to do pose estimation so that we can use CNN accelerator to speed up the whole process.

high computing resources. Fang et al.[10] 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 accelerator, the SLAM system on FPGAs 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*: The goal of place recognition is to calculate a given frame into a limited set of places. Each places can be encoded as a very short code which can be easy transformed 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 [11] or ORB [2], using vectorization techniques like bag-of-words (BoW) [12] or vector of locally aggregated descriptors(VLAD) [13].

Recent advances in the deep learning and the convolution neural network (CNN) enable powerful end-to-end mode for place recognition [14], [3], and the NetVLAD method is one of the most accurate method based on CNN. The NetVLAD algorithm based on VGG-16 model [15] consumes more than 80G operations for a single 300×300 input image (each operation means an addition or a multiplication). It is very challenging to deploy the NetVLAD on a traditional embedded hardware platform.

B. Hardware architecture of Zynq MPSoC

The Xilinx Zynq MPSoC is a chip with ARM cores and FPGA fabric. The system is illustrated in fig 2. 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 large volume of data such as each frame of camera. Deepphi CNN accelerator [5] is one of the state-of-

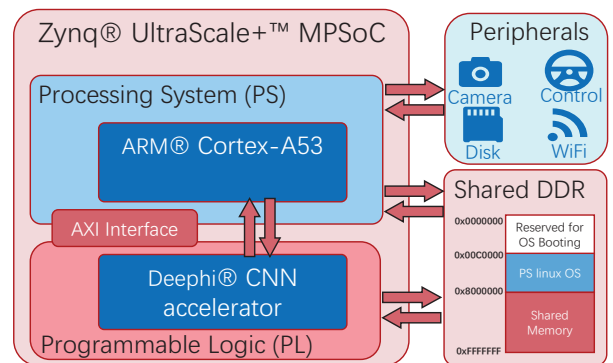


Fig. 2. Hardware architecture of Zynq SoC

the-art accelerators and is famous for high energy efficiency on various of CNN structure. We deploy the accelerator on the PL side of Zynq SoC.

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

C. Motivation

Though previous work [1] proposes data-efficient DSLAM system, it is difficult to implement the two essential components, VO and place recognition simultaneously on a communication-limited and energy-constrained embedded hardware platform on a real robot. We propose this hardware-software co-design DSLAM system to use Xilinx Zynq MPSoC and Deepphi accelerator to execute these two components on real system.

III. HARDWARE-SOFTWARE CO-DESIGN DSLAM

Our hardware-software co-design DSLAM system contains two essential improvements in the pose estimation and the place recognition tasks. As illustrate in fig 1(b), both of these two components are divide into two stages: 1) CNN front end to extract features which is deployed to the CNN

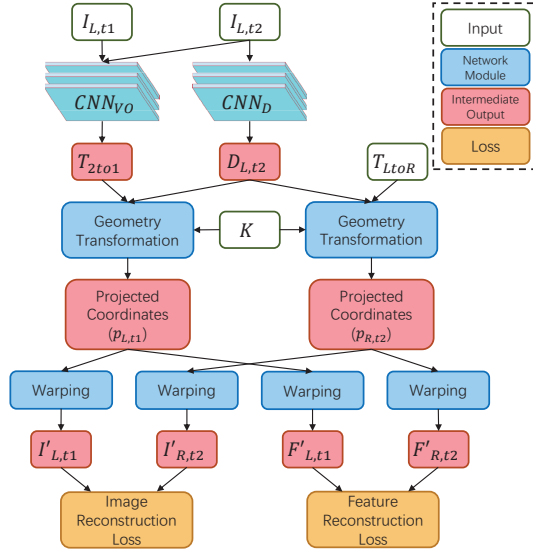


Fig. 3. Illustration of Depth-VO-Feat framework in training phase, where T_{LtoR} is the relative camera pose transformations between right and left views, and K denotes the known camera intrinsic matrix. CNN_{VO} and CNN_D are convolutional nets for visual odometry and depth estimation respectively, which can be used independently in testing phase.

accelerator on PL and 2) geometric operations to present final results which is deployed on the PS ARM core. To make full use of the Zynq MPSoC (illustrated in fig 2), we optimize the data flow for both of these components.

A. Pose Estimation

We adopt Depth-VO-Feat [8] 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's 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 direction is learned. In Depth-VO-Feat [8], 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 testing phase. Besides, to fix scale-ambiguity issue, we use stereo sequences in training phase and monocular sequences in testing phase. With the known spatial relationship between the left and right cameras, our neural networks can learn the real world scale. Feature reconstruction loss is an additional supervision signal, used to improve the robustness of this framework. And we use depth smoothness loss to encourage the predicted depth to be smooth, which demonstrated success in prior works. Then the final loss becomes:

$$L = \lambda_{ir} L_{ir} + \lambda_{fr} L_{fr} + \lambda_{ds} L_{ds} \quad (1)$$

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

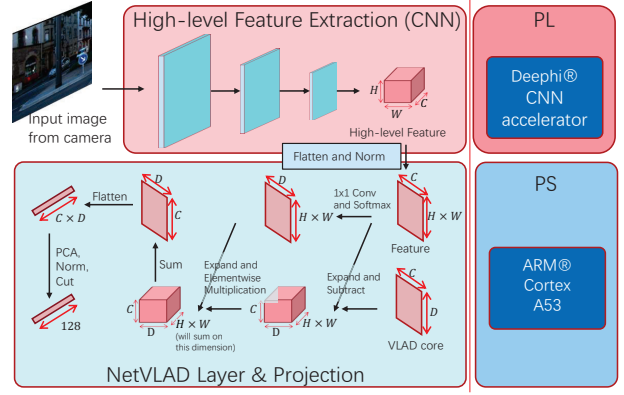


Fig. 4. Process of NetVLAD. The CNN encoder is running at the CNN accelerator on PL side, and the VLAD layer as well as the PCA is running at the ARM core at PS side.

In order to run 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 FPGA. Here we adopt the fixed-point finetune method in [16], 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. As the fixed-point method will lead to the accuracy loss of the model, We attempt several different quantization strategies to balance speed and accuracy, which will be shown in detail in section IV.

B. Place Recognition

The place recognition method provide the encoded vector transformed to the central agent for inter-robot place matching. As described in section II, CNN has achieved great improvements in place recognition tasks, and NetVLAD [3] is one of the most impressive methods. The computation flow of NetVLAD is illustrated in fig 4. 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 [3] is a recently proposed plug-and-play operation that greatly improve the performance of place recognition. In the original work with the VLAD layer [3], the feature extraction encoder is a typical CNN VGG-16 [15]. The output dimension of original NetVLAD is usually tens of thousands, which is very difficult to stored on 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[1] show that 128 dimension is plenty for DSLAM. The data flow and operations of the NetVLAD layer and the projection are complex and require floating-point number, which cannot be supported with DeepPhi DPU. We implement the NetVLAD layer and

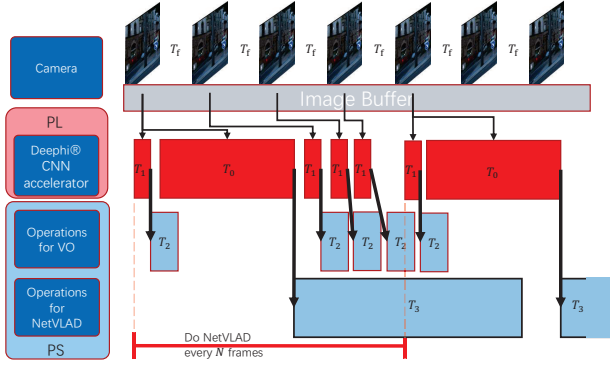


Fig. 5. Scheduling pipeline. There are 4 threads: Camera read, Deepphi core at PL, PS Operations for VO, and PS Operations for NetVLAD.

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 complex, and also we cannot finetune the NetVLAD model because of the lack of training data. We simply analyse 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. This method is proposed in [17] 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 imbalance. We do pipeline optimization to schedule the two components on Zynq MPSoC. 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 constrain is given as eq. (2).

$$N \times T_f > T_0 + N \times T_1 \quad (2)$$

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

$$N \times T_f > T_0 + T_1 + (N - 1) \times T_2 \quad (3)$$

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

$$N \times T_f > T_3 \quad (4)$$

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

IV. EXPERIMENTS

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

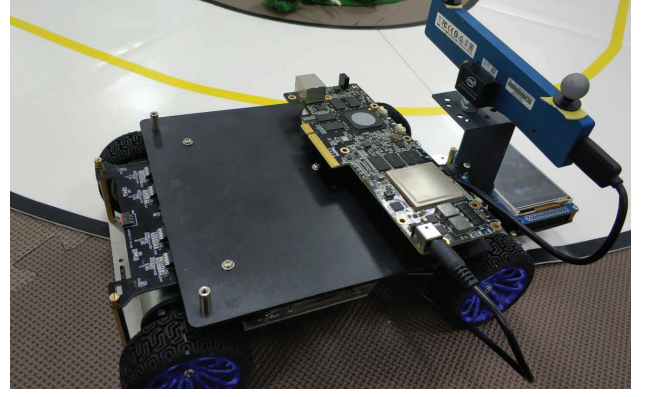


Fig. 6. The intelligent car with the Zynq MPSoC board.

A. Run-Time

We evaluate our DSLAM on two intelligent cars which are controlled by the Deepphi Aristotle board with a ZU9 MPSoC. The intelligent car and the board is illustrated in fig 6. Table I shows the run time of each part of our DSLAM system. The pipeline of scheduling of these parts is shown in section II.

TABLE I
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.

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.

B. VO with DPU

We evaluate our VO approach with DPU on KITTI dataset [18]. The dataset contains 11 video sequences with stereo pairs, with original image size at 1242×375 pixels. In our design, we resize the image to 608×160 as the original Depth-VO-Feat dose [8].

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 500 and report the average translational and rotational errors for the testing sequence 00 in table II. The comparison of the estimated trajectory for the methods is illustrated in fig 7.

Because of the fixed-point number used in our design, which sacrifice the precision of the CNN, the VO accuracy is not as good as the previous works [2], [8]. On the other

TABLE II
VISUAL ODOMETRY (VO) RESULT

method	t_{err}	r_{err}	run time (ms/frame)
ORB-SLAM			
Depth-VO-Feat			
Ours			12

* t_{err} is the average translational drift error.
 r_{err} is average rotational drift error.

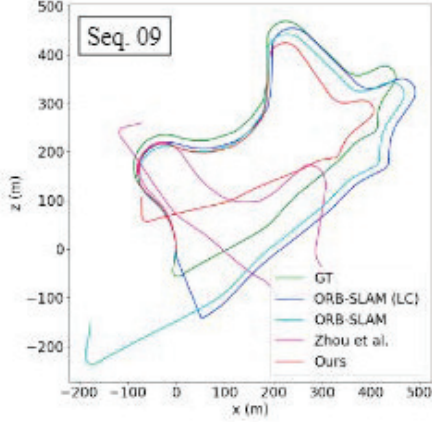


Fig. 7. Trajectory of different VO methods on sequence 00.

hand, our VO can calculate the 6-D pose within $20ms$ on a resource constrained embedded platform.

C. NetVLAD with DPU

We evaluate the NetVLAD performance on the loop close dataset for KITTI[19]. The dataset labels the ground truth of loop closure for these sequences based on the metric positions of each images. Specifically, it compares the position of each image to the others in the sequence and regard 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 8. 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 slightly accuracy reduction on the ROC curves. The following experiment shows that the slightly 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 embedded system, we can build up the DSLAM system like the previous work [1]. However, unlike the feature point based method in previous [1], our approach dose not necessarily compute the feature point for each frame. The previous work [1] transfers the

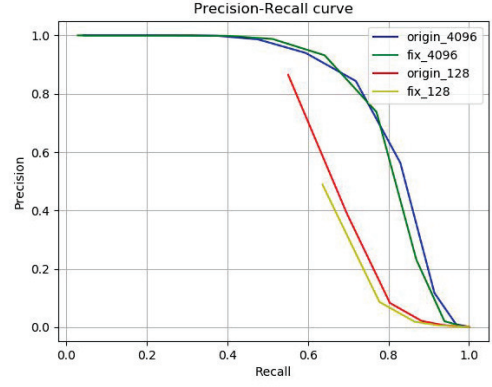


Fig. 8. ROC curves on sequence 00.

ORB feature points among agnets 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 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[13] and ORB[2]. The agents may stop and solve the feature points. We directly tranform the down-sampled 608×160 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 9.

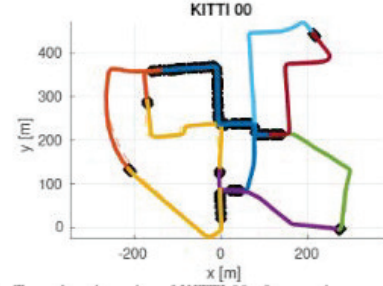


Fig. 9. The sub-trajectory of each agent and the merged trajectory.

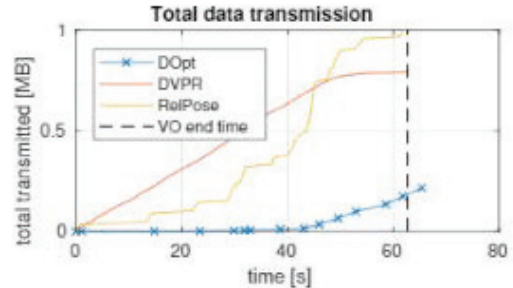


Fig. 10. The total communication traffic of the proposed DSLAM

There are three kinds of data need to be transfers 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 iamge need for inter-robot relative pose estimation (RelPose).

At the begining 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[1] also faces this problem though it use feature point for inter-robot relative pose estimation.

The total communication traffic of the propsoed DSLAM system on KITTI 00 is increasing with time, and is shown in fig 10.

V. CONCLUSION

We propose a hardware-software co-design DSLAM system with the help of Xilinx Zynq MPSoC and DeepPhi DPU. We optimize two essential components of 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 to do 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 is not supported by any fund.

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. Tardós, "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] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, 1989.
- [5] "Boards - DeepPhi Tech," 2019. [Online]. Available: <http://www.deepphi.com/technology/boards>
- [6] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 66–75.
- [7] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVO: Monocular Visual Odometry Through Unsupervised Deep Learning," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7286–7291, 2018.
- [8] 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.
- [9] 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>
- [10] 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.
- [11] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Key-points," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.
- [12] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, 2012.
- [13] 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.
- [14] 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.
- [15] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [16] 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>
- [17] 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.
- [18] 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.
- [19] "Kitti groundtruth," 2019. [Online]. Available: <https://github.com/ZhangXiwu/KITTI.GroundTruth>