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

Jincheng Yu[1], Feng Gao[1], Jianfei Cao[1], Zhaoliang Zhang[1] and Yu Wang[1]

Abstract— Decentralized simultaneous localization and mapping (DSLAM) is essential to a multi-robot system, especially in environments lacking absolute positioning equipment like GPS. Monocular visual-based SLAM is a widely adopted solution in the industry for its low cost and high flexibility. 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 the 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 the publicly available dataset.

## I. Introduction

In recent years, with the development of the hardware and algorithms, 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. 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 [?] proposes a data-efficient decentralized SLAM(DSLAM) system. The DSLAM frame in [?] is illustrated in ??. It makes improvements in three typical components in the DSLAM system: 1) Using ORB-SLAM [?] in stereo configuration as the visual odometry (VO) algorithm which provides basic intra-robot pose estimation. 2) Using NetVLAD [?] algorithm to do place recognition which relates the current observation to previous scenes and other robots. 3) Using distributed Gauss-Seidel algorithm [?] 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 [?] to perform NetVLAD for each frame. We also notice that there are also some previous works regress the 6-D pose directly from the input stereo camera [?] or monocular camera [?], [?], With the development of CNN. We adopt Depth-VO-Feat [?] in DSLAM system to estimate the pose from the input monocular camera. Because Depth-VO-Feat is trained with stereo input frames and feedforward with the monocular camera, the CNN method can provide the absolute scale from the monocular camera, and also be accelerated with the CNN accelerator [?]. Thus we do not need to execute ORB-SLAM on embedded CPUs.

The proposed DSLAM framework is illustrated in ??. 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.

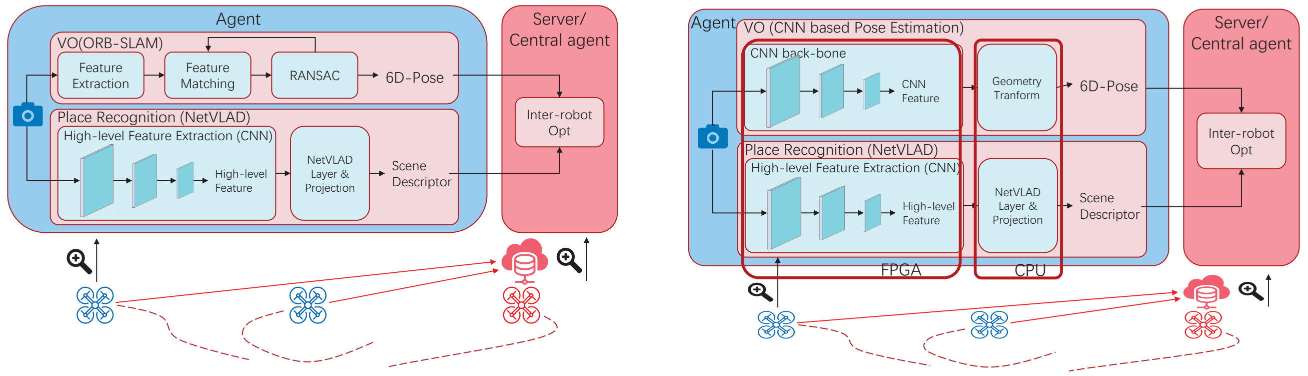
The rest part of this article is orgnized as follows. ?? will give the basic idea of CNN based methods and the hardware architecture of embedded FPGA. ?? will detail the implementation of our hardware-software co-design DSLAM system. The experiment result will be given in ??. ?? will conclude this paper.

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



(a) DSLAM in [?]

(b) Our hardware-software co-design DSLAM.

Fig. 1. Overview of the DSLAM in [?] 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.

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[?] and ORB-SLAM2[?]. Recently, several studies have shown that these feature-based SLAM systems require high computing resources. Fang et al.[?] 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: 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 [?] or ORB [?], using vectorization techniques like bag-of-words (BoW) [?] or vector of locally aggregated descriptors(VLAD) [?].

Recent advances in the deep learning and the convolution neural network (CNN) enable powerful end-to-end mode for place recognition [?], [?], and the NetVLAD method is one of the most accurate methods based on CNN. The NetVLAD algorithm based on VGG-16 model [?] 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 system is illustrated in ??. The

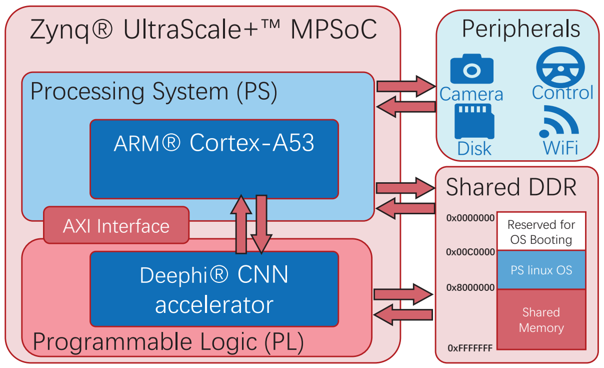


Fig. 2. Hardware architecture of Zynq SoC

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 [?], 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 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.

## C. Motivation

Though previous work [?] proposes data-efficient DSLAM system, it is difficult to implement the two essential components, VO and place recognition si-

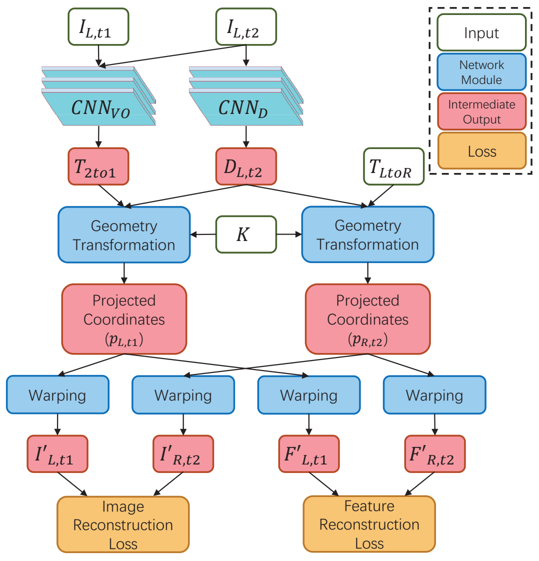


Fig. 3. Illustration of Depth-VO-Feat framework in the training phase, where $T_{LtoR}$ is the relative camera pose transformations between right and left views, and $K$ denotes the known camera intrinsic matrix. $CNN_{VO}$ and $CNN_D$ are convolutional nets for visual odometry and depth estimation respectively,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

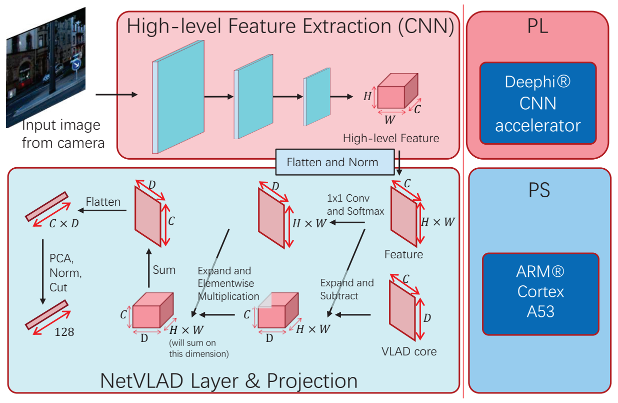


Fig. 4. Process of NetVLAD. The CNN encoder is running at the CNN acclerator on PL side, and the VLAD layer as well as the PCA is running at the ARM core at PS side.

multaneously 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 Deephi DPU to execute these two components on the real system.

## III. Hardware-Software Co-design DSLAM

Our hardware-software co-design DSLAM system contains two essential improvements in the pose estimation and place recognition tasks. As illustrate in ??, 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) geometric operations to present final results which are deployed on the PS ARM cores. To make full use of the Zynq MPSoC (illustrated in ??), we optimize the data follow for both of these components.

### A. Pose Estimation

We adopt Depth-VO-Feat [?] 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 [?], 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. Feature reconstruction loss is an additional supervision signal, used to improve the robustness of this framework. 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:

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

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

In order to run Depth-VO-Feat 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 [?], 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 ??.

### B. Place Recognition

The place recognition method provide the encoded vector transferred to the central agent for inter-robot place matching. As described in ??, CNN has achieved significant improvements in place recognition tasks, and NetVLAD [?] is one of the most impressive methods. The computation flow of NetVLAD is illustrated in ??. 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 [?] is a recently proposed plug-and-play operation that greatly improves the performance of place recognition. In the original work with the VLAD layer [?], the feature extraction encoder is a typical CNN named VGG-16 [?]. 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[?] 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 to minimize the truncation error of each layer. This method is proposed in [?] 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 ??. 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 ??.

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

The thread for VO on PS constrains the NetVLAD frequency as ??.

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

The PS part of NetVLAD should finish before computing the PS part of next NetVLAD frame. This constraint can be written as ??.

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

The execution time of our design will be given in ??.

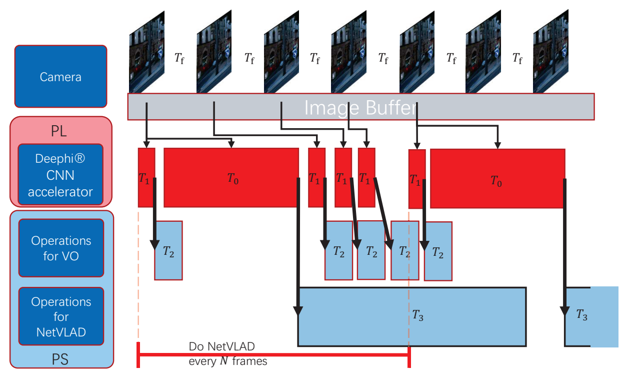


Fig. 5. Scheduling pipeline. There are four threads: Camera read, Deephi core at PL, PS Operations for VO, and PS Operations for NetVLAD.

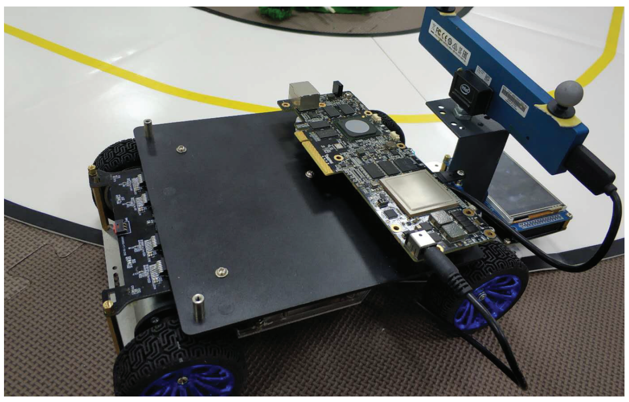


Fig. 6. The intelligent car with the Zynq MPSoC board.

## IV. Experiments

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

### A. Run-Time

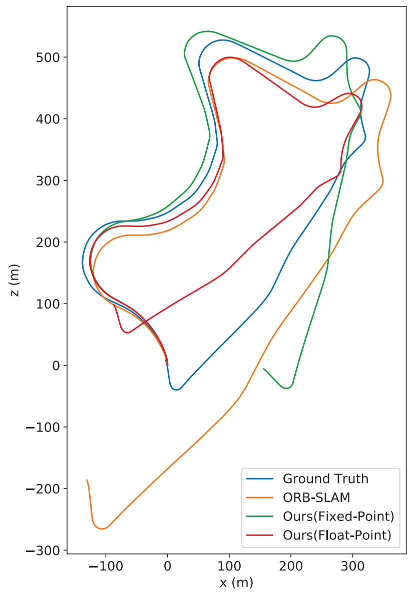
We evaluate our DSLAM on two intelligent cars which are controlled by the Deephi Aristotle board with a ZU9 MPSOC. The intelligent car and the board is illustrated in ??. ?? shows the run time of each part of our DSLAM system. The pipeline of scheduling of these parts is shown in ??.

TABLE I
Run-Time of each part in our DSLAM

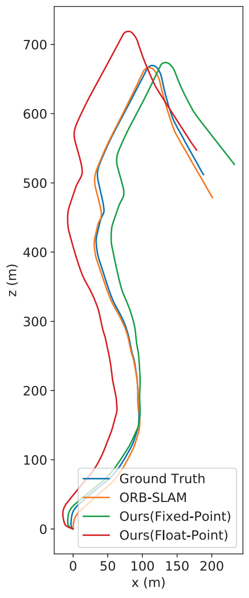
| | NetVLAD, PL ($T_0$) | VO, PL ($T_1$) | VO, PS ($T_2$) | NetVLAD, PS ($T_3$) |
|---|---|---|---|---|
| Execute time (ms) | 66 | 3 | 10 | 354 |

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

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



(a) Traj. on Seq. 09



(b) Traj. on Seq. 10

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

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 [?]. 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 [?].

At the fixed-finetune procedure, we use the stereo pairs in sequences 01 to 10 as the training set. The training super parameters in ?? 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.

TABLE II

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

| method | 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 | |
| Ours(float) | 10.49 | 3.34 | 11.64 | 3.14 | |
| Ours(fixed) | 10.27 | 4.08 | 8.84 | 4.01 | 12 |

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

We evaluate the Viusal Odometry on KITTI odometry dataset and report the average translational and rotational errors for the testing sequences 09 and 10 in ??. The comparison of the estimated trajectory for the methods is illustrated in ??.

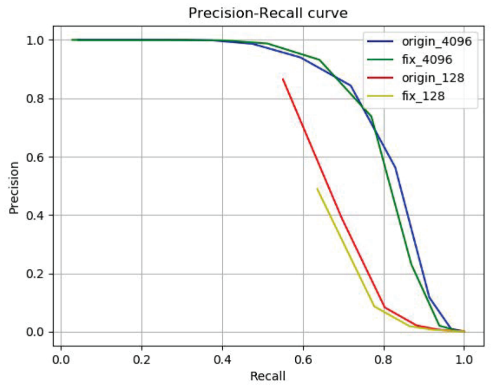


Fig. 8. ROC curves on sequence 00.

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 [?], [?]. On the other 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[?]. 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 ??. 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 [?]. However, unlike the feature point based method in previous [?], our approach does not necessarily compute the feature point for each frame. The previous work [?] 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[?] and ORB[?]. 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 ??.

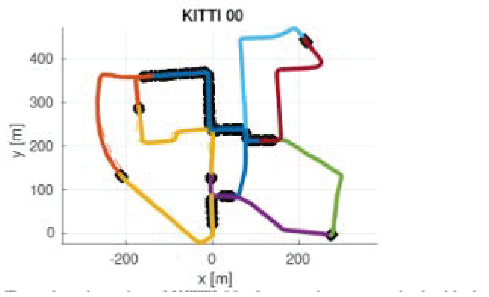


Fig. 9. ??? The sub-trajectory of each agent and the merged trajectory.

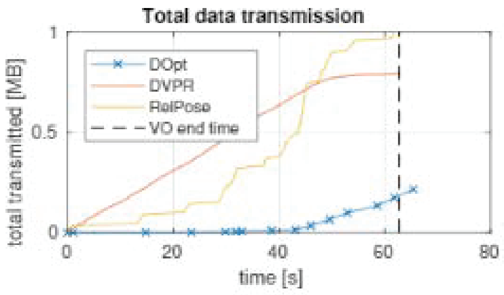


Fig. 10. ??? The total communication traffic of the propsoed DSLAM

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[?] 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 ??.

## 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 is not supported by any fund.