MobileSP: An FPGA-Based Real-Time Keypoint Extraction Hardware Accelerator for Mobile VSLAM

Ye Liu, *Student Member, IEEE*, Jingyuan Li, Kun Huang, *Graduate Student Member, IEEE*, Xiangting Li, Xiuyuan Qi, Liang Chang, *Member, IEEE*, Yu Long, and Jun Zhou, *Senior Member, IEEE*

***Abstract*— Keypoint extraction is a key technique for Visual Simultaneous Localization and Mapping (VSLAM). Recently, Convolutional Neural Network (CNN) has been used in the keypoint extraction for improving the accuracy. As one of the state-of-the-art CNN based keypoint extraction techniques, the SuperPoint ranked top in the CVPR2020 image matching challenge. However, the use of complex CNN makes it difficult to meet the real-time performance on a mobile platform with limited resource such as mobile robots and wearable Augmented Reality (AR) devices. In this work, based on the SuperPoint, we proposed an FPGA-based real-time keypoint extraction hardware accelerator through algorithm-hardware co-design for mobile VSLAM applications, which is named as MobileSP. Several algorithm and hardware level design techniques have been proposed to reduce the computation and improve the processing speed while maintaining high accuracy, including a partially shared detection & description encoding architecture, a pre-sorting based Non-Maximum Suppression (NMS) engine and a software-hardware hybrid pipeline computing technique. The design has been implemented and evaluated on a ZCU104 FPGA board. It achieves real-time performance of 42 fps with low Absolute Trajectory Error (ATE) of 1.82 cm simultaneously, outperforming several state-of-the-art designs.**

***Index Terms*— Keypoint extraction, CNN, FPGA, hardware accelerator, mobile VSLAM.**

I. Introduction

KEYPOINTS (or feature point) extraction, consisting of keypoint detection [1] and descriptor generation [2], is a pivotal module for VSLAM [3]–[5], which is widely used in applications such as autonomous vehicle [6], mobile robot [7] and AR [8]. The keypoint detection is to detect repeatable points that contain rich geometric information (e.g., endpoints of lines, center of circles, junctions and corners) and the keypoint description is to represent local features of each detected point with a vector (named descriptor). After the keypoint extraction, the correspondences between the keypoints of the image pair can be established during the matching, as shown in Fig. 1.

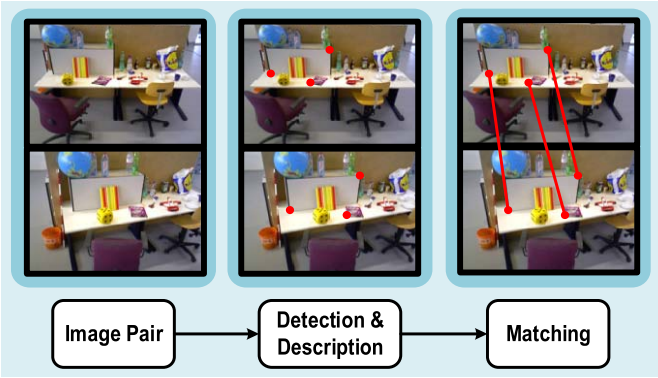

Fig. 1. Keypoint detection and description.

The classical keypoint extraction approaches such as SIFT [9], SURF [10], ORB [11] follow a detect-then-describe pipeline that firstly detect geometrically distinctive points and then encode the image patch centered at each point into a descriptor. In the recent years, keypoint extraction methods using CNN [12], [13] have shown higher accuracy by leveraging the powerful feature extraction capability of deep learning. For example, LIFT [14], LF-Net [15], D2-Net [16] and RF-Net [17]. They show superiority than the classical methods, especially for wide-baseline matching with large viewpoint change. As one of the state-of-the-art keypoint extraction methods using CNN, the SuperPoint [18] features a self-supervised learning framework with Homographic Adaptation to boost rich points detection and robust descriptor. It is combined with a matching algorithm SuperGlue [19] and ranked the first in the CVPR2020 image matching challenge. However, the use of CNN in the keypoint extraction has caused high computational complexity, making these methods difficult to achieve real-time performance when implemented on mobile platforms with limited computing resource. This poses challenges for them to be used in mobile VSLAM applications such as mobile robots and wearable AR devices.

Recently, some FPGA-based hardware accelerators have been designed to speed up the keypoint extraction. For example, [20] implemented an ORB based keypoint extraction accelerator with Rotationally Symmetric BRIEF [2] and applied it in VSLAM system. [21] proposed a customized

Manuscript received 24 March 2022; revised 13 June 2022; accepted 5 July 2022. Date of publication 21 July 2022; date of current version 9 December 2022. This work was supported by the National Natural Science Foundation of China (NSFC) through NSAF under Grant U2030204. This article was recommended by Associate Editor A. James. *(Corresponding author: Jun Zhou.)*

The authors are with the University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China (e-mail: liuye2018@std.uestc.edu.cn; zhouj@uestc.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2022.3190300.

Digital Object Identifier 10.1109/TCSI.2022.3190300

hardware architecture for sparse algebra in VSLAM. In [22], [23], hardware accelerators have been proposed to speed up neural network. [24] implemented a SuperPoint based keypoint extraction hardware accelerator using CNN on FPGA. However, due to the lack of algorithm-hardware co-design, only limited processing speed or accuracy is achieved.

In this work, based on the SuperPoint, we proposed an FPGA-based real-time keypoint extraction hardware accelerator (named MobileSP) through algorithm-hardware co-optimization for mobile VSLAM applications. The main contributions of this work are summarized as below:

- A partially shared detection and description encoding architecture is proposed to reduce the computation and facilitate the parallel processing;
- A pre-sorting based NMS engine is proposed to reduce the computation and improve the processing speed while maintaining high accuracy;
- A software-hardware hybrid pipeline computing technique is proposed to further improve the processing speed;
- The proposed design has been implemented and evaluated on a ZCU104 FPGA board. It achieves real-time performance of 42 fps with high accuracy, outperforming several state-of-the-art designs;

The rest of the paper is organized as follows: Section II reviews the related works. Section III introduces the SuperPoint which is our baseline. Section IV presents the details of proposed algorithm and hardware design techniques of the MobileSP. Section V shows and discusses the experimental results, and Section VI concludes the paper.

II. Related Works

As a basic research topic of geometric computer vision, keypoint extraction is to detect repeatable keypoints and encodes local feature of each detected points as a descriptor. In the past decades, a number of keypoint extraction algorithms based on feature engineering or deep learning have been proposed.

Moravec [1] is one of the earliest keypoint detection algorithms, proposed by Moravec in 1981. The algorithm measures the difference between each image block and its eight directions with variance, and take the center pixel of the image block with high variance as the keypoint. But this algorithm is not isotropic and hence has a higher response to textures in specific directions. In 1988, the Harris [24] keypoint detection improved Moravec, which take the gray-scale change in any direction into account. This improvement makes the algorithm invariant to brightness, contrast, and rotation. In 1994, the Shi-Tomasi [26] corner detection algorithm proposed by Shi Jianbo and Carlo Tomasi further improved Harris, and introduces eigenvalues of the matrix into the confidence score of keypoints to achieve a higher robustness than Harris.

As the first widely recognized integrated keypoint detection and description method, SIFT [9] on the one hand, shows advantageous performance that offers a baseline for many subsequent works, and on the other, prevents itself from wide application for its high computational complexity. Thus, some subsequent works aim at reducing the computational complexity. For example, the SURF [10] leverages square-shaped filter instead of Difference of Gaussian as an approximate replacement of scale-normalized Laplacian of Gaussian, which significantly speeds up the keypoint detection. The BRIEF [2] introduces binary descriptor which is matched with Hamming Distance to reduce the memory consumption and computational cost of matching. The DAISY [27] applies convolution with Gaussian kernel to oriented gradient map to realize efficient descriptor generation. The FAST [28] is the first attempt of using machine leaning to approximate a keypoint classifier that significantly increase the detection speed. The ORB [11] combines and optimizes FAST detector and BRIEF descriptor to further propose a lightweight replacement for SIFT.

The above-mentioned methods mainly aim to reduce the complexity, but the improvement on the accuracy is very limited. To further improve the accuracy, especially in the cases with significant illumination and viewpoint changes, some new methods have been proposed. For example, the TILDE [29] detector learns from repeatable point across images sequences captured under different illumination to improve the accuracy and robustness of the detection. Early attempts with machine learning mainly focus on detector. In the following years, learned descriptor gains increasing attention. For instance, as a CNN trained with ground truth generated by SIFT and SFM [30], [31], LIFT [14] shows better performance than SIFT across a number of metrics. L2-NET [32] learns features of image patches from Euclidean space with CNN to further improve the accuracy. With the same network structure as L2-NET, HardNET [33] proposed a novel loss function to enhance patch-based learning that achieves better performance.

Different from above works that compute detection and description with separate modules, the merged framework simultaneously computes the detection and description. For instance, the D2-NET [16] uses a completely shared CNN pipeline for detection and description where keypoints are determined as the local maximum on descriptor map. the R2D2 [34] takes reliability into account while determining keypoint and leverages shared backbone for both tasks. Recently, SuperPoint [18] has been proposed which achieves significant performance improvement compared to prior works. SuperPoint uses a shared CNN backbone and separate explicit decoders for detector and descriptor and proposes a self-supervised framework that boosts detector and descriptor learning with Homographic Adaptation. With the self-supervised framework, training of SuperPoint is initialized from synthetic dataset to avoid the limitation of artificially annotated datasets. SuperPoint achieves a superior performance compared to previous methods, especially for the image pairs with significant illumination change and viewpoint change. It ranked the first together with a state-of-the-art matching method (named SuperGlue [19]) in the image matching challenge of CVPR 2020.

The keypoint extraction methods are widely applied in mobile platforms such as robots and drones, where the computing resources are very limited. Thus, over the past few years, a number of works aim at accelerate keypoint extraction on FPGA. By introducing Rotationally Symmetric BRIEF,

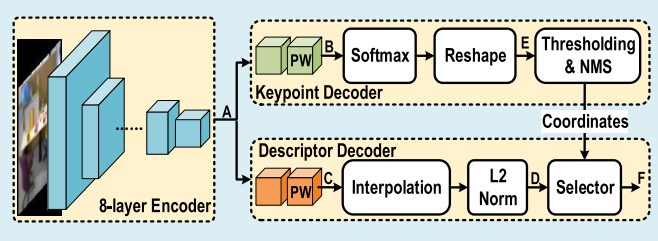

Fig. 2. Architecture of the SuperPoint.

[20] customized ORB to be more hardware friendly and implemented an accelerator on FPGA as the front end of VSLAM system. As an accelerator for PRB-SLAM, [21] also implemented a keypoint extraction with customized hardware. [24] implemented a SuperPoint based keypoint extraction hardware accelerator on FPGA.

Compared to the conventional methods, the CNN based detection and description methods have significantly improved the performance. However, the introduction of neural network also incurs large computational complexity, which causes long processing time especially on the resource constrained computing platforms such as smartphone, robots and drones. A number of existing architecture attempt to accelerate keypoint extraction on FPGA, but due to the lack of algorithm-hardware co-design, only limited processing speed or accuracy is achieved. Thus, how to further speed up keypoint detection and description on edge platforms while achieving high accuracy needs to be investigated.

III. SuperPoint

In this work, we choose the SuperPoint as our baseline for its state-of-the-art performance [18]. The SuperPoint consists of 3 modules, which are respectively 8-layer VGG-like encoder, keypoint decoder and descriptor decoder, as shown in Fig. 2. With a grayscale image input $I \in \mathbb{R}^{H\times W}$, the 8-layer encoder extracts features for both detection and description, and outputs a tensor $A \in \mathbb{R}^{H_c\times W_c\times 128}$, where $H_c = \frac{H}{8}$, $W_c = \frac{W}{8}$. Both keypoint decoder and descriptor decoder take tensor A as input. The keypoint decoder firstly maps tensor A to a tensor $B \in \mathbb{R}^{H_c\times W_c\times 65}$ through two convolution layers, in which the second layer is pointwise convolution, then performs Softmax logical regression along channels. The first 64 channels of regressed tensor will be reshaped to $\mathbb{R}^{H\times W}$, which can be seen as the confidence score map of keypoints, and the last channel is dropped. The final keypoints detection results are obtained by applying thresholding and NMS to the confidence scores. The descriptor decoder first maps tensor A to a coarse descriptor tensor $C \in \mathbb{R}^{H_c\times W_c\times 256}$ with two convolution layers, in which the second layer is pointwise, then interpolates C to $\mathbb{R}^{H\times W\times 256}$. By applying a channel-wise L2 normalization to the interpolated tensor, the dense descriptor tensor $D \in \mathbb{R}^{H\times W\times 256}$ is obtained, where the 256-dimension vector at each position is exactly the descriptor of the corresponding point on original image. Finally, the descriptor decoder selects the corresponding descriptors from dense descriptor tensor D according to the keypoints coordinates resulted from keypoint decoder. The coordinates of detected keypoints together with their corresponding descriptors are the final output of the SuperPoint. The parameters of the SuperPoint backbone are shown in Table I. Three MaxPooling layers are inserted into the previous eight layers of the network to reduce the dimension of image size and aggregate the feature information. The remaining four layers are used for keypoint detection and descriptor generation respectively.

Compared to the previous methods, the SuperPoint achieved superior accuracy especially when it comes to difficult scenarios such as large viewpoint change or illumination change. This helps the SuperPoint win the top ranking in the CVPR2020 image matching challenge.

IV. Proposed MobileSP

Although the SuperPoint achieved excellent accuracy, it has large computational complexity due to the use of CNN, resulting in high computation latency as shown in Table I. This makes it difficult to achieve real-time performance when implemented on a mobile platform with limited computing resource. To address this issue, we have proposed an FPGA-based real-time keypoint extraction hardware accelerator named MobileSP. In this design, both algorithm and hardware level design techniques have been proposed to reduce the computation and improve the processing speed while maintaining high accuracy, including a partially shared detection and description encoding architecture, a pre-sorting based NMS engine, and a software-hardware hybrid pipeline computing technique. The details of these techniques are described as follows.

A. Partially Shared Detection and Description Encoding Architecture

Firstly, we have noticed that the SuperPoint computes a dense keypoints confidence score map E and a dense descriptor tensor D as shown in Fig. 2, but only a small proportion (normally less than 1%) of keypoints and their corresponding descriptors are selected as the outputs F. This means that a large number of operations for the keypoint extraction in the SuperPoint are actually redundant, resulting unnecessary computation.

The redundancy in the description can be removed by performing keypoint description decoding after the detection. Then the description decoding module only needs to compute the descriptors for the obtained keypoints. However, this will slow down the flow as the detection and description will be in serial in this case. We have observed that the keypoint detection mainly relies on low-level geometric local features such as corners, edges, endpoints of lines and center of circles while the description needs high-level global semantic features. It indicates that the depth of neural network required for detection and description are different. The detection requires shallower network than the description to achieve satisfactory performance. Therefore, instead of using the same layers of encoder for both detection and description as in the SuperPoint, the layers of the two tasks can be different.

Based on this consideration, as shown in Fig. 3, we have proposed a partially shared detection and description encoding architecture. The encoder stage 1 extracts low-level geometric

TABLE I
PARAMETERS OF THE SUPERPOINT

TABLE I
PARAMETERS OF THE SUPERPOINT

Layer	Kernel Size	Output Size*
Conv_1	64*[1, 3, 3]	[64, H, W]
Conv_2	64*[64, 3, 3]	[64, H, W]
Pooling_1	2*2 MaxPooling	[64, H/2, W/2]
Conv_3	64*[64, 3, 3]	[64, H/2, W/2]
Conv_4	64*[64, 3, 3]	[64, H/2, W/2]
Pooling_2	2*2 MaxPooling	[128, H/4, W/4]
Conv_5	128*[64, 3, 3]	[128, H/4, W/4]
Conv_6	128*[128, 3, 3]	[128, H/4, W/4]
Pooling_3	2*2 MaxPooling	[128, H/8, W/8]
Conv_7	128*[128, 3, 3]	[128, H/8, W/8]
Conv_8	128*[128, 3, 3]	[128, H/8, W/8]
Conv_Keypoint_1	256*[128, 3, 3]	[256, H/8, W/8]
Conv_Keypoint_2	65*[256, 1, 1]	[65, H/8, W/8]
Conv_Descriptor_1	256*[128, 3, 3]	[256, H/8, W/8]
Conv_Descriptor_2	256*[256, 1, 1]	[256, H/8, W/8]

*The size of input image is H*W.

features which are shared by both detection and description. Different from the SuperPoint, which use maxpooling to perform subsampling, we use convolution to achieve a similar effect. The encoder stage 1 contains 4 convolution layers, among which the second and the forth layer is strided convolution with a stride of 2. Thus the encoder stage 1 realized a 4 times subsampling. With the output tensor $L \in \mathbb{R}^{(H/4)\times(W/4)\times 128}$ from the stage 1, the encoder stage 2, which contains 3 convolution layers, extracts high-level features from the tensor L to compute a tensor $H \in \mathbb{R}^{H_c\times W_c\times 256}$ containing global and semantic high-level features for descriptor generation. In the detection decoder, we use two convolution layers to decode tensor L to the confidence map, where the first layer is convolution with a stride of 2, which is different from that of the SuperPoint. The parameters of the proposed backbone are shown in Table II.

The advantage of this encoding architecture is that the detection decoding will complete earlier than the description decoding which allows the description decoding to only compute the descriptors for the obtained keypoints to significantly reduce the redundant computation. Furthermore, the parallelism of the algorithm is not affected since the detection decoding can operate in parallel with the encoder stage 2. In addition, since the keypoint detection mainly relies on low-level features as aforementioned, the impact of this architecture revision on the detection accuracy is negligible, which can be seen from the experimental results later.

To realize this, the operations in the description decoding including selective interpolation, pointwise convolution and L2 normalization are now performed only on the detected keypoints. For example, in the SuperPoint, the pointwise convolution in the description decoding is performed on a 2D

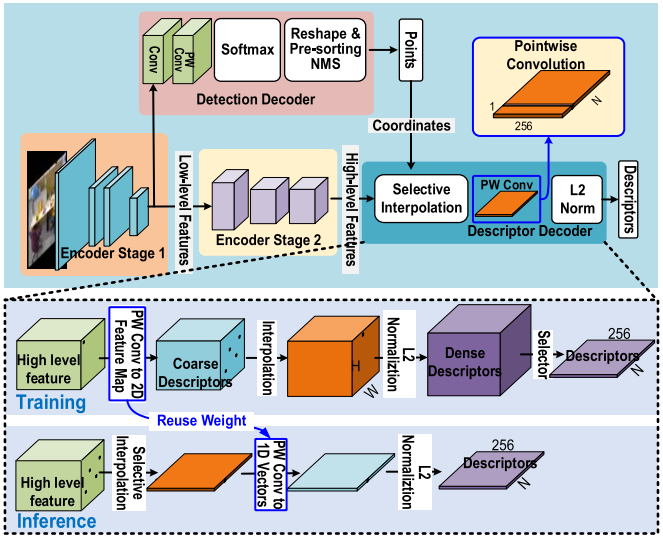

Fig. 3. Proposed partially shared detection and description encoding architecture.

TABLE II
PARAMETERS OF THE MOBILESP

Layer	Kernel Size	Output Size*
Conv_1	64*[1, 3, 3]	[64, H, W]
Conv_2	64*[64, 3, 3]	[64, H/2, W/2]
Conv_3	64*[64, 3, 3]	[64, H/2, W/2]
Conv_4	128*[64, 3, 3]	[128, H/4, W/4]
Conv_Keypoint_1	256*[128, 3, 3]	[256, H/8, W/8]
Conv_Keypoint_2	65*[256, 1, 1]	[65, H/8, W/8]
Conv_Descriptor_1	128*[128, 3, 3]	[128, H/4, W/4]
Conv_Descriptor_2	128*[128, 3, 3]	[128, H/8, W/8]
Conv_Descriptor_3	256*[128, 3, 3]	[256, H/8, W/8]

*The size of input image is H*W.

feature map containing the information of keypoints and non-keypoints, but in the proposed method it should be performed only on the detected keypoints which compose a 1D vector.

However, the training loss function in the original method requires information of both keypoints and non-keypoints in the pointwise convolution. Therefore, during the training phase, we still keep the information of non-keypoints in the description decoding part to facilitate the training using the original loss function. During the inference phase, we remove the redundant computation by only performing the description decoding on the detected keypoints. As shown in Fig. 3, we have 2 modes respectively for the training and inference. While training the model, to enable the backward propagation, we compute point wise convolution before selective interpolation and L2-normalization. However, while inferring the model, we directly perform selective interpolation on feature-level instead of descriptor-level, that is, before the pointwise convolution. Pointwise convolution does not result any information propagation between pixels. Our derivation also shows the two modes are mathematically equivalent.

Specifically, in terms of training mode, for channel c of a resulted vector V after bilinear interpolation, we assume

that the 4 adjacent values for V_c to interpolate to be $a_{11} = f(x_1, y_1), a_{12} = f(x_1, y_2), a_{21} = f(x_2, y_1)$ and $a_{22} = f(x_2, y_2)$. Thus, the bilinear interpolation firstly computes:

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} a_{11} + \frac{x - x_1}{x_2 - x_1} a_{21} \tag{1}$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} a_{12} + \frac{x - x_1}{x_2 - x_1} a_{22} \tag{2}$$

then computes V_c after pointwise convolution and interpolation as:

$$V_c = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \tag{3}$$

that is,

$$V_c = \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} a_{11} + \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} a_{21} + \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} a_{12} + \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} a_{22} \tag{4}$$

where a_{11}, a_{12}, a_{21} and a_{22} are the results of pointwise convolution.

We further assume input tensor consists of high-level feature to be $P \in \mathbb{R}^{H_c \times W_c \times 256}$ and $P_{ij} \in \mathbb{R}^{256}$ to be the 256-dimensional vector corresponding to coordinates i and j in tensor P. The weight of pointwise convolution kernel corresponding to channel c of output tensor is assumed to be vector $W \in \mathbb{R}^{256}$. It will be evident to find that:

$$a_{11} = W^T \cdot P_{x_1 y_1} \tag{5}$$

$$a_{12} = W^T \cdot P_{x_1 y_2} \tag{6}$$

$$a_{21} = W^T \cdot P_{x_2 y_1} \tag{7}$$

$$a_{22} = W^T \cdot P_{x_2 y_2} \tag{8}$$

Therefore, the resulted value V_c after pointwise convolution and interpolation can be represented as:

$$\begin{aligned} V_c &= \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} W^T \cdot P_{x_1 y_1} + \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} W^T \cdot P_{x_2 y_1} \\ &\quad + \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} W^T \cdot P_{x_1 y_2} + \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} W^T \cdot P_{x_2 y_2} \\ &= W^T \cdot \left[\frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} P_{x_1 y_1} + \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} P_{x_2 y_1} + \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} P_{x_1 y_2} \right. \\ &\quad \left. + \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} P_{x_2 y_2} \right] \\ &= W^T \cdot P' \end{aligned} \tag{9}$$

where P' is exactly the bilinear interpolation of 4 adjacent vectors $P_{x_1 y_1}, P_{x_2 y_1}, P_{x_1 y_2}$ and $P_{x_2 y_2}$ in input tensor P. Consequently, we noticed that with same weight of pointwise convolution, exchanging the computation order of bilinear interpolation and pointwise convolution does not result any change to V_c, which ensure that no accuracy loss is introduced for the inference mode.

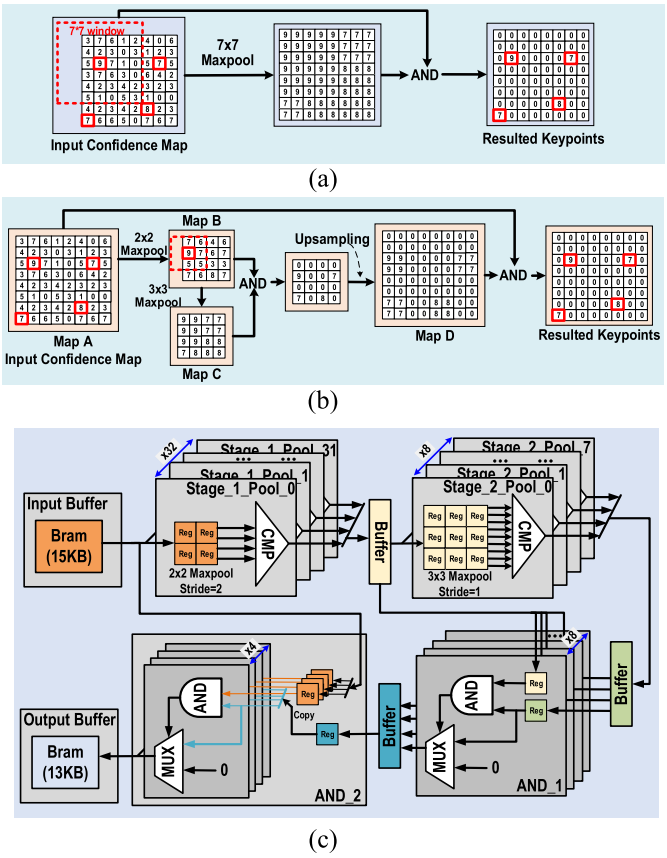

Fig. 4. (a) Conventional NMS (b) Proposed pre-sorting based NMS (c) Hardware architecture of the NMS engine.

B. Pre-Sorting Based NMS Engine

In the SuperPoint, the NMS is applied to obtain a sparse and uniform keypoint distribution. Conventional NMS algorithms compare the center pixel with other pixels in a square window to determine whether the center pixel is local maximum. The values on confidence map will be suppressed to zero if it is not the local maximum. This operation is repeated for a sliding square window with a stride of one pixel. On the PC, this operation is usually implemented through a common MaxPooling function with stride of 1 which is provided in most deep learning frameworks and well optimized for parallel computation on the GPU. Then a special AND function is applied between the generated feature map with the original feature map. In the AND function each pixel is compared with that in the original feature map and if they are equal the pixel will be kept. Otherwise, it is suppressed to zero, as shown in Fig. 4(a). In the MaxPooling function, every time when the window is shifted, local maximum calculation needs to be performed to all the data in the window. However, when the NMS window is shifted, only a column of pixels is changed. Thus, the local maximum calculation for the unchanged pixels in each NMS window are actually redundant.

In the MobileSP, we proposed a pre-sorting based NMS engine to reduce the redundant computation and improve the processing speed. The data flow and the detailed hardware implementation are shown in Fig. 4(b) and 4(c), respectively. For example, when performing NMS with 7×7 window and

stride of 1, we first perform a 2×2 MaxPooling with stride of 2 on the input feature map A, as shown in Fig. 4(b). This acts as a pre-sorting step and results in a subsampled feature map B. After that, we perform a 3×3 MaxPooling with stride of 1 on the sub-sampled feature map, then apply the AND function between the resulted feature map C and the sub-sampled map B. The above operation is effectively similar to the NMS with 7×7 window (the impact of approximation on the accuracy is negligible), while it significantly reduces the computation as the NMS is performed to the sub-sampled feature map after pre-sorting. As the above operation results in a 4 times sub-sampled feature map, it is transformed back by applying a nearest upsampling in which one pixel is copied 4 times and applying the AND function again between the upsampled feature map D with the original input feature map A. After being transformed to the original size, the keypoints distribution map is obtained.

By using the pre-sorting based NMS, the reduction of computation can be calculated as below. For a 7×7 NMS applied to $H \times W$ confidence score map, the total number of comparison operations for the conventional implementation is:

$$(H \times W) \times (7 \times 7 - 1) + (H \times W) = 49HW \quad (10)$$

With the proposed method, the total number of comparison operations is:

$$\left(\frac{H}{2} \times \frac{W}{2}\right) \times (2 \times 2 - 1) + \left(\frac{H}{2} \times \frac{W}{2}\right) \times (3 \times 3 - 1) + \left(\frac{H}{2} \times \frac{W}{2}\right) + (H \times W) = 4HW \quad (11)$$

Therefore, with the proposed method, 91.84% of comparison operations could be saved.

To further improve the processing speed, parallel processing architectures have been designed to accelerate the MaxPooling and AND computation, as shown in Fig. 4(c). The Stage_1 in Fig.4(c) corresponds to the 2∗2 MaxPooling with stride of 2 from Map A to Map B in Fig.4(b), meaning that the Stage_1 module completes the 2∗2 MaxPooling with stride of 2. The Stage_2 in Fig.4(c) corresponds to the 3∗3 MaxPooling with stride of 1 from Map B to Map C in Fig.4(b), meaning that the Stage_2 module performs the 3∗3 MaxPooling with stride of 1. The Stage_1 is designed with 32 parallel processing channels and the Stage_2 is designed with 8 parallel processing channels to improve the parallelism while considering their own workload. The module AND_1 and AND_2 correspond to the two AND functions as aforementioned. They are designed with 8 and 4 parallel processing channels, respectively. Adding the AND function in the NMS causes a slight increase in the processing time. To address this issue, we have designed pipelined hardware for the NMS where the AND function is performed in parallel with other operations. This design has eliminated the increase in the processing time.

C. A Software-Hardware Hybrid Pipeline Computing Technique

Fig. 5(a) shows the hardware architecture of the MobileSP. As like some previous works [24], [35], the neural network

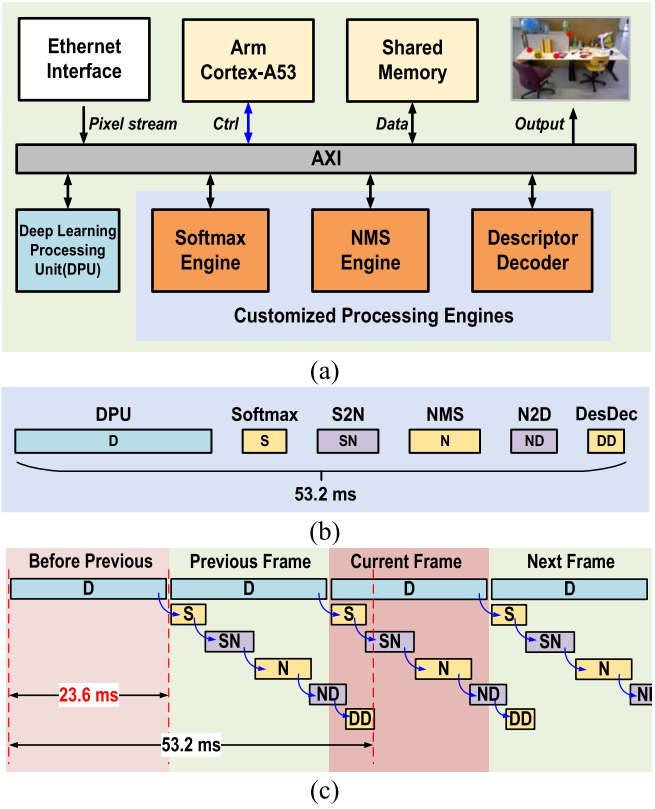

Fig. 5. (a) Hardware architecture (b) Non-pipelined operating flow (c) Proposed software-hardware hybrid pipeline computing.

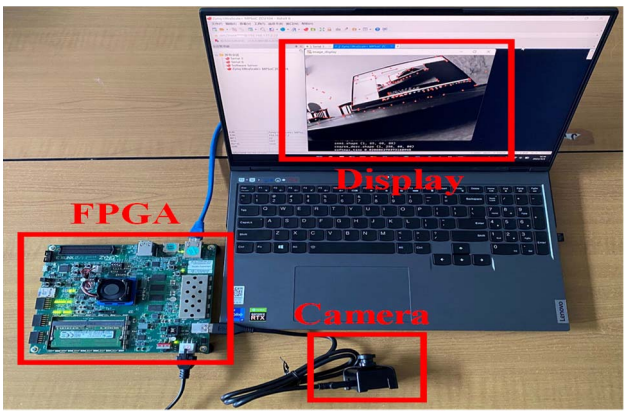

Fig. 6. The testing setup of the MobileSP.

part is implemented using the DPU which is a hard IP provided by Xilinx FPGA to accelerate the neural network. The other parts including the Softmax, NMS and Descriptor Decoder (DesDec) are implemented as customized processing engines to speed up the processing. The ARM Cortex-A53 core is used to coordinate the operation and data exchange of different modules.

Fig. 5(b) shows the operating flow of processing one image frame and the time consumption of each step. The operating flow includes the data processing of DPU, Softmax, NMS and DesDec. It also includes two data reshaping steps from the Softmax to the NMS and from the NMS to the DesDec (S2N and N2D), which are performed by the ARM Cortex-A53 core. By adding up the time consumption of each step, we can calculate the time consumption of the entire operating

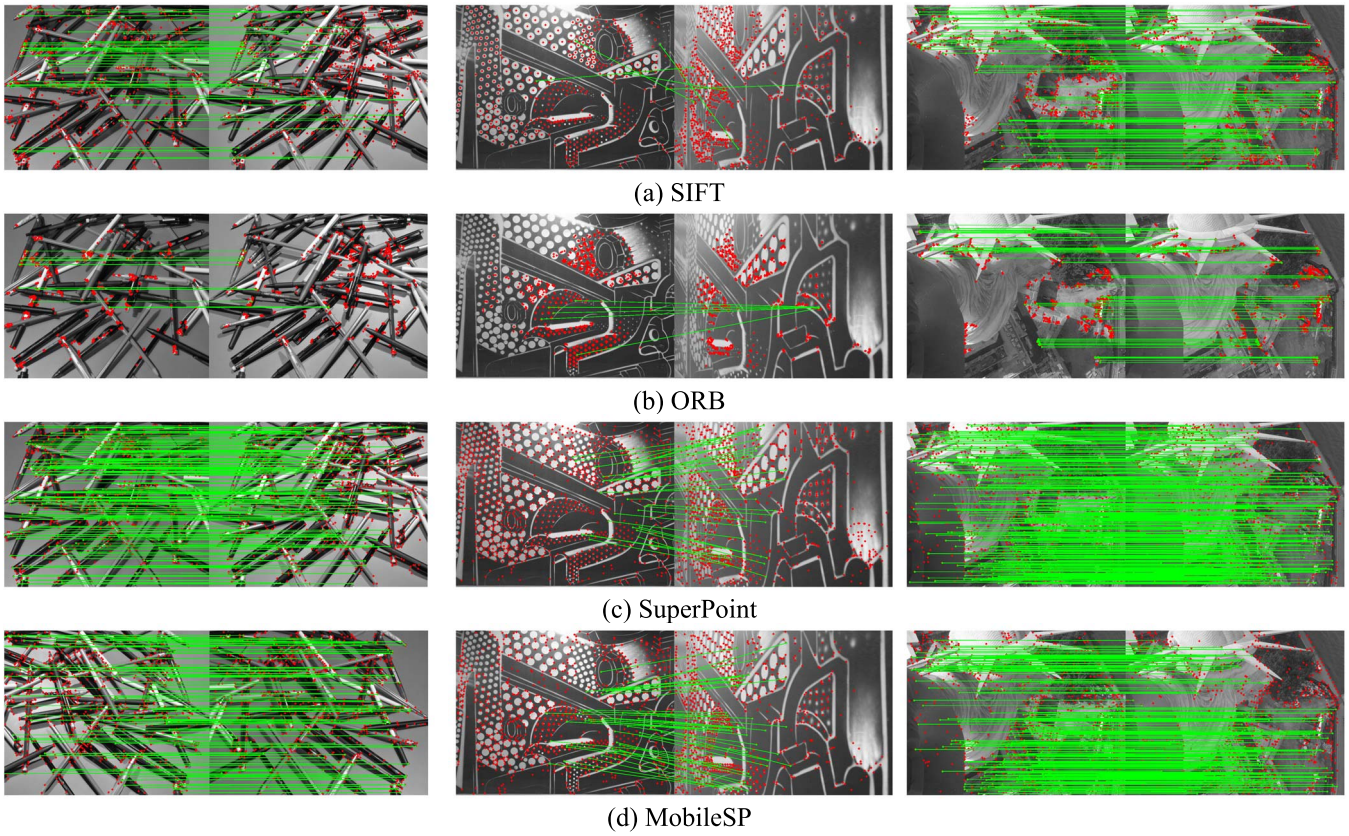

Fig. 7. Exemplary results of keypoints extraction & matching. The green lines indicate the matched points.

flow which is 53.2ms. This is still very large for real-time processing.

To improve the processing throughput, we have proposed a software-hardware hybrid pipeline computing technique, as shown in Fig. 5(c). To enable this, the hardware modules have been modified so that they can start processing the next frame once the current frame is completed. For example, the DPU can start computing the CNN part in the next frame once the CNN in the current frame is done. It is same for the Softmax, NMS and DesDec modules. The handshake interfaces among the modules have been carefully designed to ensure smooth operating flow. However, the S2N and N2D operations are performed as software in the ARM core. To pipeline them together with the hardware, we have created two software threads for each of them. As shown in Fig. 5(c), in the pipelined operating flow, while the DPU is processing the current frame, the Softmax, S2N and NMS modules are processing the previous frame, and the N2D and DesDec modules are processing the frame before the previous frame. In this way, the effective time consumption between two neighboring frames is largely shortened from 53.2ms to 23.6ms.

V. Experimental Results

A. Experiment Setup

The proposed MobileSP has been implemented and verified on a Xilinx ZCU104 FPGA board with operating frequency of 150MHz. Fig. 6 shows the testing setup. A camera is used to capture the real-time image data, which is then transmitted to the FPGA board for real-time processing. The processed results are transmitted to the laptop for displaying through the Ethernet interface.

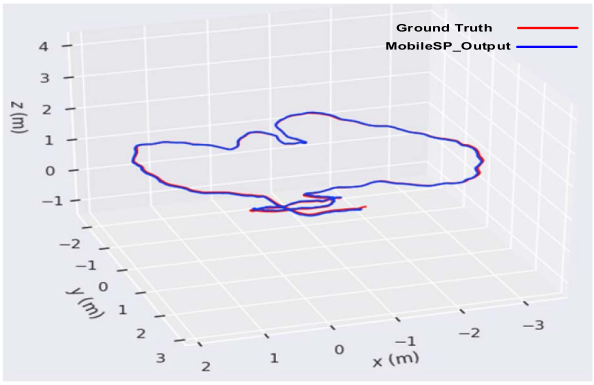

Fig. 8. The output example of the MobileSP with ground truth, running a sequence from TUM dataset.

In the design, the DPU uses 8-bit fixed-point data and the other modules such as the Softmax Engine and the NMS Engine use floating-point data. The accuracy of the MobileSP is evaluated with two commonly used datasets HPatches [36] and TUM [37]. The HPatches is mainly used to evaluate the accuracy of keypoint extraction, while the TUM is used to evaluate the accuracy of visual odometry (VO) which combines the keypoint extraction with the matching and the posture estimation. For the testing with dataset, the laptop is used to send the images from the datasets to the FPGA through the Ethernet interface and the extracted keypoints from

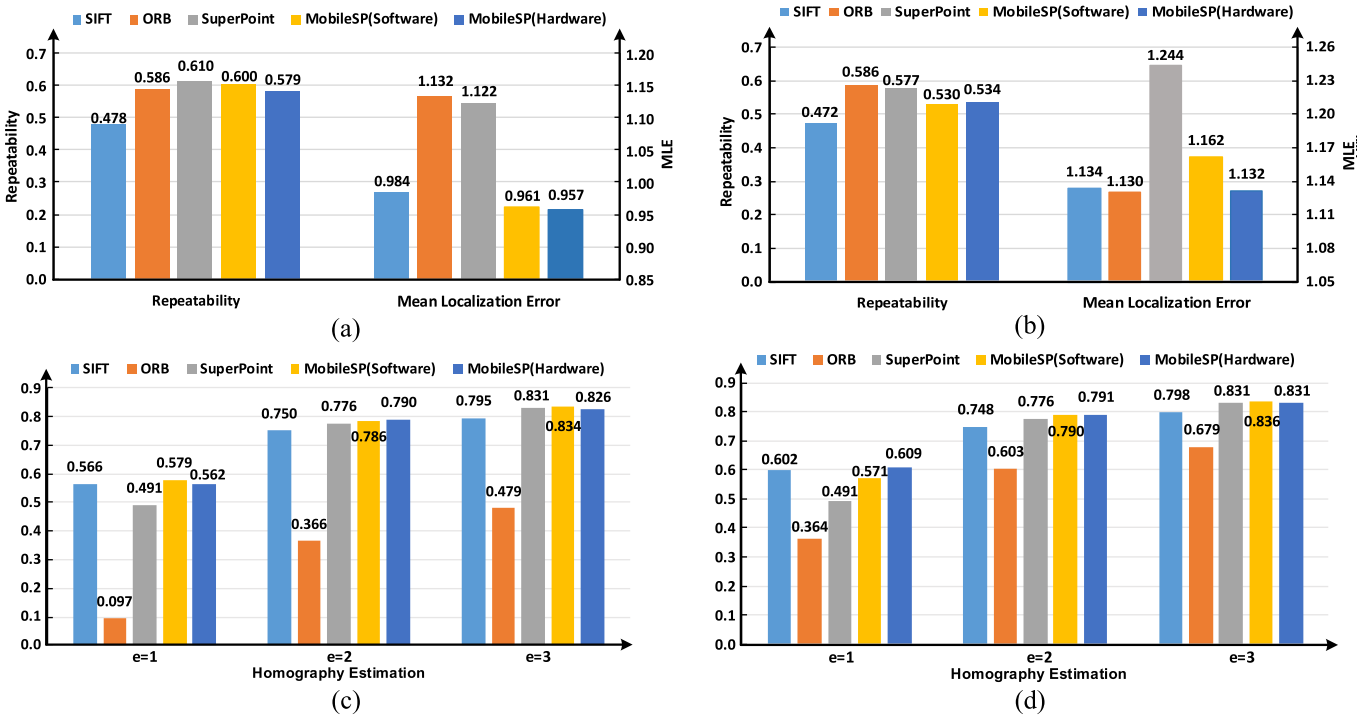

Fig. 9. Accuracy comparison on HPatches datasets. (a) reliability and MLE for the image resolution of 240 × 320 (b) reliability and MLE for the image resolution of 480 × 640 (c) homography estimation for the image resolution of 240 × 320 (d) homography estimation for the image resolution of 480 × 640.

the FPGA are sent back to the laptop for analysis and post-processing. On the laptop we have constructed a VO system using the common ORB-SLAM2 framework for the matching and the posture estimation.

B. Accuracy Results

Fig. 7 shows some qualitative results of the keypoints extraction and matching using the MobileSP as the keypoint extractor, compared with the well-known keypoint extraction algorithms including the SIFT [9], ORB [11] and SuperPoint [18]. It can be seen from the images that the results of the MobileSP are very similar to that of the SuperPoint and are much better than the SIFT and ORB which only extracted limited number of keypoints that can be successfully matched. Especially for the second column of image pairs, the SIFT and ORB almost failed to match due to low quality of keypoint detection and description. To demonstrate the robustness of the MobileSP, Fig. 8 shows the output trajectory of the MobileSP (blue line) compared with the ground truth (red line). The two trajectories are well overlapped as can be seen from the figure.

To obtain quantitative results, the accuracy of the MobileSP has been evaluated using the HPatches dataset. Three commonly used metrics including the repeatability, mean localization error (MLE) and homography estimation [18] are used for the evaluation. For the images in the dataset, as in the SuperPoint work [18], two resolutions including 480×640 and 240×320 are tested. For the homography estimation, various thresholds of correctness (e) are tested. As shown in Fig. 9, the MobileSP achieves relatively high reliability and low MLE, compared to other methods. It also achieves relatively high homography estimation for different thresholds of correctness, compared to other methods. In addition, we have also

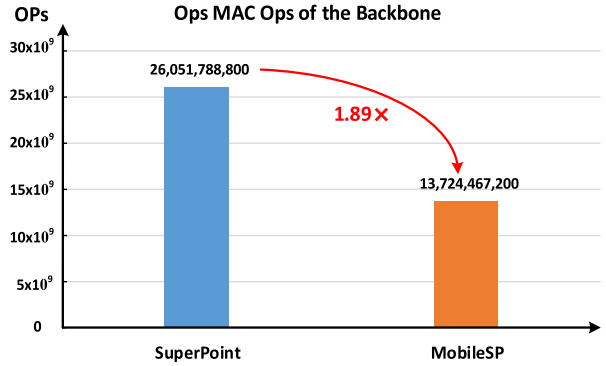

Fig. 10. The number of MAC operations of the backbone of the MobileSP compared with the SuperPoint.

compared the accuracy of hardware version of the MobileSP on FPGA with the software version of the MobileSP on PC. It can be seen from Fig. 9 that the hardware version has very similar accuracy to that of the software version.

C. Performance Analysis

Fig. 10 shows the number of MAC operations of the backbone of the MobileSP compared with SuperPoint. It can be seen that the number of MAC operations is reduced by 1.89× proposed with the proposed partially shared detection and description encoding architecture. Fig. 11 shows the computation reduction of the proposed pre-sorting based NMS engine. It can be seen that the number of comparison operations of the NMS is reduced by 12.25× with the proposed technique. Fig. 12 shows the comparison of the processing time per frame between the proposed hybrid pipeline architecture and the non-pipeline architecture. The processing time is reduced by 2.2× with the proposed architecture.

TABLE III
COMPARISON WITH THE STATE-OF-THE-ART DESIGNS

		DAC2019 [20]	DAC2020 [21]	FCCM2020 [24]	DAC2020 [38]	**Our Work**
FPGA		Zynq-7045	Zynq-7020	ZCU102	ZCU102	**ZCU104**
Utilization	LUT	56954	14472	80599	-	**100913**
	FF	67809	16686	175434	-	**195029**
	DSP	111	75	1307	-	**1318**
	BRAM	78	252	499.5	-	**205**
Frequency		100MHz	100MHz	200MHz	300MHz	**150MHz**
Resolution		480×640	480×640	480×640	480×640	**480×640**
Performance		31.45fps/55.87fps*	10fps**	20fps	20fps	**42fps**
Dataset		TUM	EuRoC	TUM	-	**TUM**
ATE***		4.3cm	-	39.76cm	-	**1.82cm**

*The key frame rate is 31.45 fps and the normal frame rate is 55.87 fps.
**Estimated from the reference [21].
***Absolute Trajectory Error.

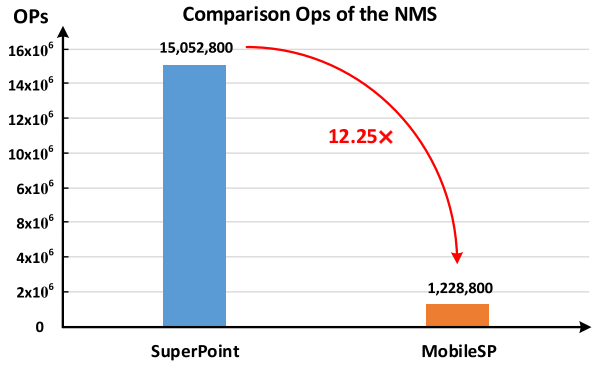

Fig. 11. The number of comparison operations of NMS of the MobileSP compared with the SuperPoint.

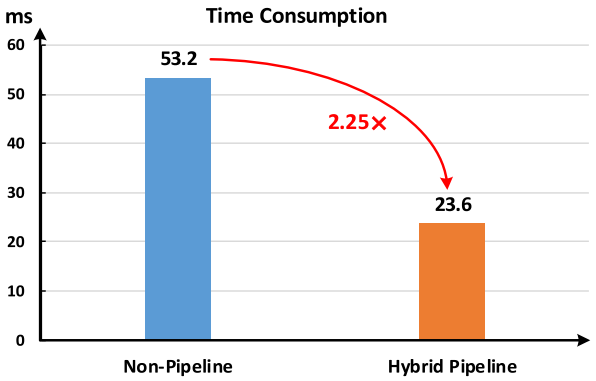

Fig. 12. Time consumption reduction of the proposed hybrid pipeline on FPGA.

We have also compared the utilization, performance and accuracy of the MobileSP with several state-of-the-art hardware designs [20], [21], [24], [38] as shown in Table III. Compared to other work, the proposed design consumes more hardware resource. This is mainly due to the use of deep convolutional neural network which help reduce the ATE and improve other performance such as repeatability, mean location error and homography estimation. Compared to [24] which also uses deep convolutional neural network, we have achieved more than 2 times higher frame rate. As most of these designs use the TUM dataset to evaluate the accuracy of keypoint extraction within the VO framework based on the ATE, we have followed their methods. With the proposed algorithm-hardware co-design techniques, the MobileSP achieves a high frame rate (42fps) with the lowest ATE, compared with other designs.

VI. CONCLUSION

In this work, we proposed an FPGA-based real-time keypoint extraction hardware accelerator named MobileSP for mobile VSLAM applications. This design combines several algorithm- and hardware-level design techniques including a partially shared detection & description encoding architecture, a pre-sorting based NMS engine and a software-hardware hybrid pipeline computing technique to reduce the computation and improve the processing speed while maintaining high accuracy. The proposed design has been implemented and verified on a ZCU104 FPGA board. It achieves real-time performance of 42 fps with low ATE of 1.82 cm simultaneously, outperforming several state-of-the-art designs.

REFERENCES

[1] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 1980.
[2] M. Calonder, "BRIEF: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, Sep. 2010, pp. 778–792.
[3] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007, doi: 10.1109/TPAMI.2007.1049.
[4] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.
[5] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.
[6] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Jun. 2016, doi: 10.1109/TIV.2016.2578706.

[7] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, Feb. 2002, doi: 10.1109/34.982903.

[8] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Comput. Graph. Appl.*, vol. 21, no. 6, pp. 34–47, Nov. 2001, doi: 10.1109/38.963459.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, Nov. 2004.

[10] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2006, pp. 404–417.

[11] E. Rublee, V. Rabaud, K. Konolidge, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. ICCV*, Nov. 2011, pp. 2564–2571.

[12] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989, doi: 10.1162/neco.1989.1.4.541.

[13] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and Cooperation in Neural Nets*. Berlin, Germany: Springer, 1982, pp. 267–285.

[14] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform," in *Proc. ECCV*, 2016, pp. 467–483.

[15] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "LF-Net: Learning local features from images," in *Proc. NIPS*, 2018, pp. 1–11.

[16] M. Dusmanu *et al.*, "D2-Net: A trainable CNN for joint detection and description of local features," in *Proc. CVPR*, 2019, pp. 8084–8093.

[17] X. Shen *et al.*, "RF-Net: An end-to-end image matching network based on receptive field," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8132–8140.

[18] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 224–236.

[19] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4938–4947.

[20] R. Liu, J. Yang, Y. Chen, and W. Zhao, "eSLAM: An energy-efficient accelerator for real-time ORB-SLAM on FPGA platform," in *Proc. 56th Annu. Design Automat. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[21] B. Asgari, R. Hadidi, N. Shoghi Ghaleshahi, and H. Kim, "PISCES: Power-aware implementation of SLAM by customizing efficient sparse algebra," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6, doi: 10.1109/DAC18072.2020.9218550.

[22] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Economic LSTM approach for recurrent neural networks," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 11, pp. 1885–1889, Nov. 2019, doi: 10.1109/TCSII.2019.2924663.

[23] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Designing novel AAD pooling in hardware for a convolutional neural network accelerator," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 3, pp. 303–314, Mar. 2022, doi: 10.1109/TVLSI.2021.3139904.

[24] Z. Xu, J. Yu, C. Yu, H. Shen, Y. Wang, and H. Yang, "CNN-based feature-point extraction for real-time visual SLAM on embedded FPGA," in *Proc. IEEE 28th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM)*, May 2020, pp. 33–37, doi: 10.1109/FCCM48280.2020.00014.

[25] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, Aug. 1988, vol. 15, no. 50, p. 5244.

[26] J. Shi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 1994, pp. 593–600, doi: 10.1109/CVPR.1994.323794.

[27] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, May 2010, doi: 10.1109/TPAMI.2009.77.

[28] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2010, pp. 183–196.

[29] Y. Verdie, K. Moo Yi, P. Fua, and V. Lepetit, "TILDE: A Temporally Invariant Learned DEtector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5279–5288, doi: 10.1109/CVPR.2015.7299165.

[30] M. Pollefeys, R. Koch, and L. Van Gool, "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters," *Int. J. Comput. Vis.*, vol. 32, no. 1, pp. 7–25, 1999.

[31] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *Int. J. Comput. Vis.*, vol. 80, no. 2, pp. 189–210, 2007.

[32] Y. Tian, B. Fan, and F. Wu, "L2-Net: Deep learning of discriminative patch descriptor in Euclidean space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 661–669.

[33] A. Mishchuk *et al.*, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.

[34] J. Revaud *et al.*, "R2D2: Repeatable and reliable detector and descriptor," 2019, *arXiv:1906.06195*.

[35] C. Hou, C. Fang, Y. Lin, Y. Li, and J. Zhang, "Implementation of a CNN identifing modulation signals on an embedded SoC," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 490–493, doi: 10.1109/MWSCAS48704.2020.9184608.

[36] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5173–5182.

[37] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580, doi: 10.1109/IROS.2012.6385773.

[38] J. Yu *et al.*, "INCA: INterruptible CNN accelerator for multi-tasking in embedded robots," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6, doi: 10.1109/DAC18072.2020.9218717.

Ye Liu (Student Member, IEEE) received the B.S. degree in electronic information science and technology from North China Electric Power University, Baoding, China, in 2014. He is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China.

He worked at the Huaneng Group of China as an Electronics Engineer from 2014 to 2018. His current research interests include domain-specific architecture for visual SLAM and reconfigurable and low power AI processor design.

Jingyuan Li received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2016. He is currently pursuing the master's degree with the IoT Smart ICs and System Group, School of Information and Communication Engineering, University of Electronic Science and Technology of China. His research interests include digital circuit and systems, hardware accelerator, and AI processor.

Kun Huang (Graduate Student Member, IEEE) received the B.Eng. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2021, where he is currently pursuing the master's degree with the School of Information and Communication Engineering. His research interests include visual simultaneous localization and mapping algorithms and hardware acceleration.

Xiangting Li is currently pursuing the B.S. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include digital circuit and systems and hardware accelerator.

Xiuyuan Qi received the B.S. degree in electronic information engineering from Harbin Engineering University, Harbin, China, in 2022. He is currently pursuing the master's degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interest includes the design of hardware acceleration architectures in the field of visual autonomous driving.

Liang Chang (Member, IEEE) received the Ph.D. degree from Beihang University, Beijing, China. Since 2020, he has been an Associate Professor at the School of Information and Communication Engineering, University of Electronic Science and Technology of China. He has coauthored more than 40 scientific papers, including ISSCC, CCIC, ASSCC, MICRO, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, IEEE TRANSACTIONS ON COMPUTERS, ICCAD, and DATE. His research interests include computing in emerging nonvolatile memory, advanced memory-centric computer architecture, and AI processor for intelligent detection. He is the Regular Reviewer of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.

Yu Long received the B.S. and master's degrees in communication engineering from the University of Electronic Science and Technology of China, Chengdu, China. His research interests include machine vision and VSLAM/SLAM algorithm application in industry.

Jun Zhou (Senior Member, IEEE) received the dual B.S. degree in communication engineering and microelectronics from the University of Electronic Science and Technology of China (UESTC) in 2004 and the Ph.D. degree in microelectronics system design from Newcastle University, U.K., in 2008. He joined IMEC Netherlands in 2008 as a Research Scientist and worked on the energy-efficient processor design for intelligent sensing in collaboration with companies, such as Philips and NXP. In 2011, he joined the Institute of Microelectronics (IME), Agency for Science, Technology, and Research (A∗STAR), Singapore, where he led projects and supervises Ph.D. students on energy-efficient processor design for intelligent sensing. In 2017, he joined UESTC as a Professor and is currently leading the Research Group of Smart ICs and Systems for IoT Applications. His major research interest is processor and algorithm co-design for intelligent sensing. He has published more than 80 papers in prestigious conferences and journals, including ISSCC, JSSC, DAC, CICC, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, and FCCM. He is serving as an A-SSCC TPC Sub-Committee Chair of Digital Circuits & Systems, an Associate Editor of IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, an Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and the Chair of the Embedded AI Committee of Sichuan Institute of Electronics. He has also served as a TPC/OC Member for a number of IEEE conferences, including SoCC, ICCD, and ISCAS.