CNN-based Feature-point Extraction for Real-time Visual SLAM on Embedded FPGA

Zhilin Xu*‡, Jincheng Yu*‡, Chao Yu*, Hao Shen†, Yu Wang*‡ and Huazhong Yang*‡
*Department of Electronic Engineering, Tsinghua University, Beijing, China
†Meituan-Dianping Group, Beijing, China
‡Beijing National Research Center for Information Science and Technology (BNRist), Beijing, China
Email: {xuzl18, yjc16, yc19}@mails.tsinghua.edu.cn, shenhao04@meituan.com, {yu-wang, yanghz}@tsinghua.edu.cn

***Abstract*—Feature-point extraction is a fundamental step in many applications, such as image matching and Simultaneous Localization and Mapping (SLAM). The CNN-based feature-point extraction methods have made significant signs of progress in both feature-point detection and descriptor generation compared with handcrafted processes. However, the computational and storage complexity makes it difficult for CNN to run on real-time embedded systems. In this paper, we aim to deploy the advanced CNN-based feature-point extraction methods onto real-time embedded FPGA systems. We optimize the softmax data flow so that the computation of softmax and NMS can be reduced by $64\times$. We generate the normalized descriptors after picking the feature-points with the highest confidence so that the computation cost of normalization is reduced by $1500\times$. We use fixed-point in both of the CNN backbone and the post-processing operations, and implement them on the ZCU102 FPGA platform. The experimental results show that our proposed hardware-software co-design CNN-based feature-point extraction method outperforms the handcrafted techniques. Our feature-point extraction on the embedded platform runs at the speed of $20\ fps$, meeting the real-time requirement.**

I. Introduction

Simultaneously Localization and Mapping (SLAM) is the essential task of many moving robot applications, such as terrain exploration and indoor navigation. The visual odometer calculates the trajectory or position of the robot by comparing the relative positions of the feature points of each frame, which is a crucial module in SLAM. The feature-point extraction method usually has two steps: 1) feature-point detection and 2) feature descriptors generation. The descriptors of the same feature-point in different input images should be similar. SIFT [1] and ORB [2] are two popular open-source handcrafted methods for feature-point extraction. Compared with the handcrafted methods, the CNN-based feature-point extraction methods, such as DeepDesc [3] and SuperPoint [4], have made significant progress in both feature-point detection and descriptor generation. SuperPoint is one of the state-of-the-art CNN based methods, which includes both feature-point detection step and descriptor generation step in a single forward pass, and surpasses the traditional handcrafted methods in accuracy. Figure 1 shows the structure of SuperPoint.

However, due to its high computational complexity and memory footprint, it is challenging to deploy the CNN-based feature-point extraction method to the real-time embedded system. Previous works [5] [6] designed CNN accelerators on FPGA. The DPU [7] is one of the state-of-the-art CNN accelerators and is known for its energy efficiency in running various CNN structures with the help of end-to-end compiler DNNVM [8]. We adopt the DPU in this paper and quantize the CNN backbone of SuperPoint to 8-bit fixed-point number with neglectable accuracy loss. In addition to the CNN backbone, there are many *post-processing operations* in CNN-based feature-point extraction networks, such as Non-Maximum Suppression (NMS) [9] and confidence ranking in the feature-point detection, as well as pixel-wise normalization in the descriptors generation. These post-processing operations become the bottleneck of embedded systems with CNN accelerators. To deploy the entire process of feature-point extraction on real-time embedded systems, we propose a hardware-software co-design CNN-based feature-point extraction structure and accelerate the entire process on the Xilinx ZCU102 platform [10] with the following contributions:

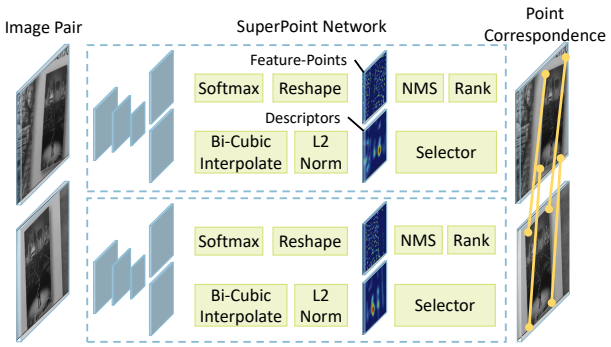

Fig. 1. Original SuperPoint for Geometric Correspondences

- We optimize the software data flow to make the post-processing operations, including the normalization, ranking, and NMS operations, consume less computational complexity and friendly to hardware.
- We quantize both of the CNN backbone and the post-processing operations to 8-bit fixed-point numbers and change the base of the power in the softmax operation from e to 2 with neglectable accuracy loss for one of the state-of-the-art CNN-based feature-point extraction methods, SuperPoint.
- We design the hardware architecture to accelerate the en-

tire process of SuperPoint, including the CNN backbone and the post-processing operations, making the feature-point extraction method run in real-time (20 fps) on embedded FPGA platform ZCU102 [10].

We evaluate our hardware-software co-design feature-point extraction method in a real-time (20 fps) SLAM system. The experimental results show that our method surpasses state-of-the-art SLAM methods.

The rest of the paper is organized as follows: We introduce the SuperPoint and the MPSOC in Section II. We present our hardware-software co-design for SuperPoint in Section III. The experimental results on the feature-point testbench ($HPatches$) [11] and the SLAM datasets (TUM) [12] are presented in Section IV. Finally, conclusions are discussed in Section V.

II. Preliminary

Figure 1 shows the overview of the feature-point extraction and matching method based on SuperPoint. The CNN backbone of SuperPoint maps the input image $I \in \mathbb{R}^{H\times W}$ to a tensor $\mathcal{X} \in \mathbb{R}^{H_c\times W_c\times 65}$ for feature-point detection and to a tensor $\mathcal{D} \in \mathbb{R}^{H_c\times W_c\times D}$ for descriptors generation, where $H_c = H/8$ and $W_c = W/8$.

The feature-point detector calculates $\mathcal{X} \in \mathbb{R}^{H_c\times W_c\times 65}$ and outputs coordinates of the k feature-points with the highest confidence. The 65 channels correspond to local, non-overlapping 8×8 pixel grid areas plus a background channel. After a channel-wise softmax, the points in different grid areas have equal confidences. Then the background channel is removed, and a $\mathbb{R}^{H_c\times W_c\times 64} \Rightarrow \mathbb{R}^{H\times W}$ reshaping is performed. The tensor of size $\mathbb{R}^{H\times W}$ corresponds to the confidence of each pixel of input image $I \in \mathbb{R}^{H\times W}$. The higher the confidence, the more likely the pixel is a feature-point. Non-Maximum Suppression (NMS) [9] is then applied to the detection to help ensure that the feature-points are evenly distributed throughout the image. The detector ranks the points by the confidence and selects k feature-points with the highest confidence. The output is the coordinate of the k feature-points with the highest confidence.

The descriptor generator first performs the bi-cubic interpolation of a semi-dense grid of descriptors $\mathcal{D} \in \mathbb{R}^{H_c\times W_c\times D}$ to obtain a dense grid of descriptors of size $\mathbb{R}^{H\times W\times D}$ and then L2-normalizes all the descriptors to unit length for further matching. A selector arranges the k descriptors corresponding to the k feature-points into the output vector according to the result of the detector.

III. Hardware-Software Co-Design

A. Softmax Optimization

The flow path of our SuperPoint-based feature-point extraction method is shown in Figure 2. Each pixel in the feature-map of the detector branch has 65 elements (noted as $\mathbf{z} = [z_1, z_2, ..., z_K], K = 65$), including the unnormalized confidence of an 8×8 area in the original input image and a background channel. The standard softmax function is defined as $\sigma(\mathbf{z})_i$ in Equation (1). $\sigma(\mathbf{z})_i$ is the normalized confidence

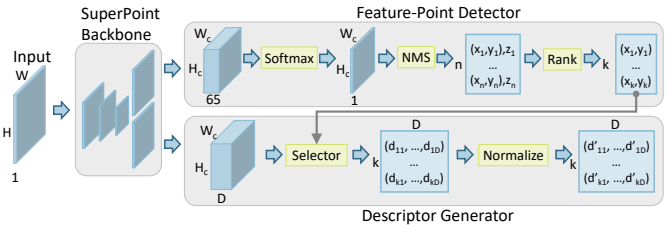

Fig. 2. Optimized Feature-point Extraction Method Based on SuperPoint

of the i^{th} point in the original 8×8 area. Since the results of the softmax function are all positive, we can calculate the reciprocal of the softmax function ($\frac{1}{\sigma(\mathbf{z})_i}$) as the normalized confidence, without affecting the results of the NMS and the ranking process. We change the base of the power from e to 2 to make it more hardware friendly. Since the divisor is a power of 2, we can also implement division by the shift operation.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, i = 1, \ldots, 64; \Rightarrow \frac{1}{\sigma'(\mathbf{z})_{\max}} = \frac{\sum_{j=1}^{K} 2^{z_j}}{2^{z_{\max}}} \tag{1}$$

The original softmax operation computes the softmax results of each element in $\mathbf{z}$. Thus, there are $\#SoftDiv_{ori}$ division operations in the original SuperPoint:

$$\#SoftDiv_{ori} = H_c \times W_c \times 64 \tag{2}$$

In most cases, only the point with the highest confidence in each grid region can be selected after ranking because softmax normalizes the confidence within each grid region. Performing the softmax operation only at the maximum point in the grid area can reduce the overhead of storage and computation and simplify the complexity of subsequent computations. Therefore, as shown in Equation (1), we only calculate the corresponding softmax result of the maximum element, which consumes only 1 division operation. There are total $\#SoftDiv_{opt}$ divisions after the softmax optimization:

$$\#SoftDiv_{opt} = H_c \times W_c \times 1 \tag{3}$$

Our method can significantly reduce the number of divisions by $64\times$, making it easy to accelerate softmax operation on FPGA.

We quantize the output feature-map of CNN (i.e., z_i) to 8-bit fixed-point numbers while still achieving comparable accuracy [5]. Figure 3(a) shows an overview of the softmax module. It consists of three parts: adder tree, comparer tree, and divider. Softmax reads 65 numbers from a grid region at once. Adder tree computes input to the power of 2 using shift operation and calculates their sum. The comparer tree reads the values of the first 64 channels and returns the maximum value, as well as its channel number which contains the position information. The divider uses the shift operation to calculate the reciprocal of confidence ($\frac{1}{\sigma'(\mathbf{z})_{\max}}$).

B. NMS Optimization

Non-Maximum Suppression (NMS) causes feature-points to be scattered throughout the whole input image. For each pixel

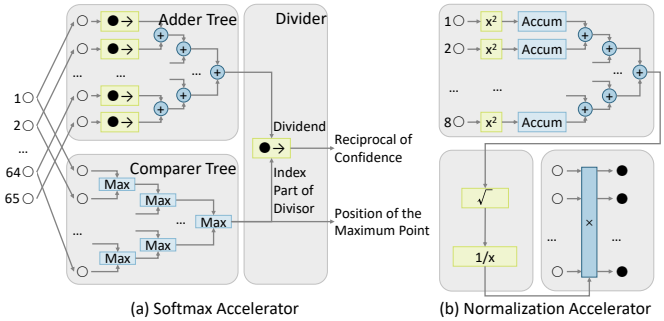

Fig. 3. Hardware Architecture of Accelerators

in the original input image, NMS compares the confidence of this pixel with that of the pixels in a square neighborhood whose edges include ϵ_{ori} pixels. If the confidence of the central target pixel is not the maximum in its neighbors, this point will be eliminated from the valid feature-points. The output of the NMS is a list of coordinates and confidences for each feature-point (z_n in Figure 2).

There are $H_c \times W_c \times 64$ points, and the NMS does $\epsilon_{ori}^2 - 1$ comparison operations for each point in the original NMS method. The ϵ_{ori} is set to 9 in the original SuperPoint. Thus, there are totally $\#NMSComp_{ori}$ comparison operations:

$$\begin{aligned} \#NMSComp_{ori} &= H_c \times W_c \times 64 \times (\epsilon_{ori}^2 - 1) \\ &= H_c \times W_c \times 5120 \end{aligned} \tag{4}$$

The softmax optimization introduced in Section III-A already gives the pixel with maximum confidence of each 8×8 block. Thus, we only need to compare each output pixel of softmax to its adjacent blocks. The comparison area is a square box with an edge of ϵ_{opt} pixels and $\epsilon_{opt} = 2 \times \lceil(\epsilon_{ori} - 1)/16\rceil + 1 = 3$. There are only $H_c \times W_c \times 1$ points. Thus, there are totally $\#NMSComp_{opt}$ comparison operations after NMS optimization:

$$\begin{aligned} \#NMSComp_{opt} &= H_c \times W_c \times 1 \times (\epsilon_{opt}^2 - 1) \\ &= H_c \times W_c \times 8 \end{aligned} \tag{5}$$

The total number of comparisons is reduced by $640\times$.

C. Ranking Optimization

The ranking operation is to find out the top k feature-points with maximum confidence. The output is a list of coordinates for the k feature-points. In the original implementation, the confidence of all valid feature-points after NMS is sorted, and only the first k feature-points are used in the applications like SLAM and image matching. There are N_{nms} valid points after NMS. The time complexity to sort all these N_{nms} points is $O(N_{nms} \cdot log(N_{nms}))$.

We create a heap of size k and then look for the k feature-points with the highest confidence [13]. We do not compute the order of these k points. The time complexity of the optimized ranking method is $O(N_{nms} \cdot log(k))$. In our experiments, $N_{nms} \approx 3000$ and $k = 200$. The running time is reduced by $8\times$, and detailed results are given in Section IV.

TABLE I
HARDWARE CONSUMPTION OF THE PROPOSED HARDWARE

	#DSP	#LUT	#FF	#BRAM
On-Board Resource	2520	418080	548160	912
DPU	1282	74496	171294	499
Softmax	0	4714	3205	0
Normalization	25	1389	935	0.5

TABLE II
RUNNING TIME COMPARISON OF EACH OPERATION

	CNN backbone*	Post-processing				
		Softmax	NMS	Rank	Norm	Total
CPU	24ms	31ms	27ms	0.97ms	42ms	100.97ms
Ours		1.97ms	0.7ms	0.12ms	1.44ms	4.23ms

* The CNN backbone runs on the accelerator.

D. Normalization Optimization

As mentioned in Section II, there are $H \times W$ descriptors that need to be normalized in the original SuperPoint. We L2-normalize the descriptors after ranking the feature-points, which means we only need to normalize k descriptors. So we put the selector before the normalization operation, as shown in Figure 2. In our experiments, we set $H = 480$, $W = 640$, and $k = 200$, then the computational complexity of the normalization process is reduced by $1500\times$.

The architecture of the normalization accelerator is illustrated in Figure 3(b). We also quantize the output feature-map of the descriptor branch to 8-bit fixed-point numbers [5]. The normalization module can read 8 numbers per clock cycle. The normalization process is divided into three stages and requires each descriptor to be read twice. In the first stage, we compute the sum of the squares of the descriptors, which takes 32 clocks cycles when $D = 256$. Then the reciprocal of the square root of the sum is computed as the normalization coefficient. In the final stage, the descriptor is read a second time and multiplied by the normalization coefficient. We quantize the descriptors to 8-bit fixed-point numbers, so the accelerator only uses very few hardware resources.

IV. EXPERIMENTS

We evaluate our method both on the feature-point testbench ($HPatches$ [11]) and the SLAM datasets (TUM [12]).

A. Hardware Resources Utilization and Optimization Effect

The proposed CNN-based feature-point extractor is implemented and evaluated on the ZCU102 evaluation board [10], which is provided by Xilinx. The CNN backbone is calculated by the Xilinx AI accelerator, DPU [7], which is a hardware IP implemented on the FPGA side of ZCU102 (Programmable logic, PL side). The softmax and the normalization steps run on our proposed accelerators, also on the PL side. The NMS and the ranking steps are operated on the CPU side (Processing System, PS side). The accelerators on the PL side, including the DPU and the proposed ones in Section III, are running at 200MHz. Table I shows the hardware resources utilization of DPU and our proposed accelerators. Our proposed accelerators only use very few hardware resources compared with the DPU.

TABLE III
ACCURACY RESULTS ON THE HPATCHES [11] DATASET

	Detector Repeatability		Homography Estimation	
	Illumination	Viewpoint	Illumination	Viewpoint
ORB	0.624	0.461	0.611	0.138
SIFT	0.597	0.486	0.807	0.264
Origin Superpoint	0.61	0.445	0.873	0.206
Ours	0.595	0.439	0.88	0.237

* Detector Repeatability is used to evaluate the accuracy of feature-point detection. Homography Estimation is used to evaluate the performance of descriptor generation. The higher, the better.

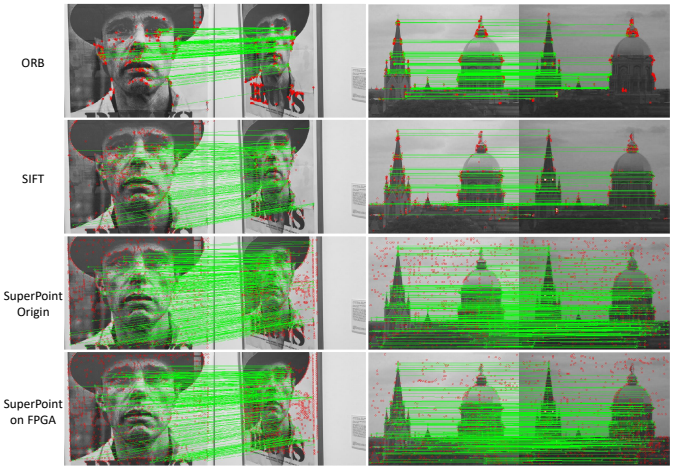

Fig. 4. Results on HPatches. The green lines show correct correspondences.

We compare the running time of each operation in SuperPoint before and after the optimization. The results are shown in Table II. The total running time of post-processing operations is reduced by more than $20\times$.

B. Result on the Feature-point Testbench

To evaluate the performance of the SuperPoint network after optimization, we compare detector repeatability and homography estimation on the $HPatches$ [11] dataset. We evaluate our system against the original SuperPoint system and well-known detector and descriptor systems ORB and SIFT.

Detector Repeatability is computed at 480×640 resolution, with 300 points detected in each image. Results are shown in Table III, the $DetectorRepeatability$ column indicates the performance of feature-point detection. There are mainly two different vision changes of the input pair: the illumination change and the viewpoint change. Our system performs on par with the original SuperPoint system, ORB, and SIFT on both of these vision changes. We compute a maximum of 1000 points for all systems at a 480×640 resolution for homography estimation. Results are shown in the $HomographyEstimation$ column of Table III. Our system outperforms ORB and performs on par with the original SuperPoint system and SIFT.

Figure 4 illustrates the visual results of ORB, SIFT, the original SuperPoint, and our design on HPatches. The feature-points extracted by ORB and SIFT are clustered together, and it is easy to obtain better results in Detector Repeatability tests. SuperPoint tends to produce more correct matches, and the feature-points are scattered throughout the whole input image, which is consistent with intuitive human feelings. Our design reaches a similar performance of the original SuperPoint.

TABLE IV
ACCURACY AND RUNNING TIME RESULTS ON THE TUM [12] SLAM DATASET

	RPE(m/s)	ATE(m)	Run time(ms)
SIFT	0.0319	0.4219	2397
ORB	0.0577	0.6105	229
Origin Superpoint	0.0280	0.3671	259
Ours	0.0283	0.3976	59

* RPE is the mean Relative Pose Error to indicate the translational drift per second. ATE is the root mean square Absolute Trajectory Error to indicate the translational drift of the entire trajectory. The less, the better.

C. Result on the SLAM Datasets

To evaluate the performance of the SuperPoint network in visual odometer, we experimented on the TUM [12] dataset. We evaluate SuperPoint against two well-known detector and descriptor systems: SIFT [1] and ORB [2]. We apply the three systems to the visual odometer. We also evaluate the performance after optimization. We compute a maximum of 200 points for all systems at a 480×640 resolution. We perform nearest neighbor matching from descriptors in adjacent frames. We use an OpenCV implementation (solvePnP()) [14] with all the matches to compute the transform matrix, and use Bundle Adjustment [15] to optimize results. All the computation of ORB and SIFT is done on the CPU. And all the computation of the original SuperPoint is done on the CPU except the CNN backbone.

The results are shown in Table IV. In terms of accuracy, SuperPoint outperforms ORB and SIFT. Our optimizations, including fixed-point quantization, and post-processing acceleration, do not introduce a significant loss of accuracy. In terms of calculation speed, SuperPoint takes less time than SIFT and is equivalent to ORB. After optimization, the running speed is increased by $4\times$, making real-time processing possible.

V. CONCLUSION

In this paper, we propose a hardware-software co-design feature-point extractor based on the state-of-the-art CNN based method, SuperPoint. The proposed feature-point extractor accelerates the entire process of SuperPoint, including the CNN backbone and the post-processing operations, making the feature-point extraction method run in real-time ($20fps$) on embedded FPGA platform ZCU102.

ACKNOWLEDGMENT

This work is supported by Meituan-Dianping Group and Beijing Sciences and Technology Project (No.Z181100008918018). This work is also supported by Independent Research Program of Electronic Engineering Department of Tsinghua University (No.20182001483, 20192001419) and Beijing Advanced Innovation Center for Future Chips.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," 2012.

[3] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 118–126.

[4] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236.

[5] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 26–35.

[6] 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 Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, p. 22, 2018.

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

[8] Y. Xing, S. Liang, L. Sui, X. Jia, J. Qiu, X. Liu, Y. Wang, Y. Shan, and Y. Wang, "Dnnvm: End-to-end compiler leveraging heterogeneous optimizations on fpga-based cnn accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[9] A. Neubeck and L. J. V. Gool, "Efficient non-maximum suppression," in *International Conference on Pattern Recognition*, ser. International Conference on Pattern Recognition, 2006.

[10] "Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit," 2019. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

[11] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[13] S. Niu, J. Guo, Y. Lan, and X. Cheng, "Top-k learning to rank: Labeling, ranking and evaluation," 2012.

[14] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[15] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment — a modern synthesis," in *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman, and R. Szeliski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372.