# HW/SW Co-design and FPGA Acceleration of a Feature-Based Visual Odometry

Chiang-Heng Chien
Department of Electrical Engineering
National Taiwan Normal University
Taipei City, Taiwan
e-mail: chiangheng.chien@gmail.com

Chiang-Ju Chien
Department of Electronic Engineering
Huafan University
New Taipei City, Taiwan
e-mail: cjc@cc.hfu.edu.tw

Chen-Chien Hsu
Department of Electrical Engineering
National Taiwan Normal University
Taipei City, Taiwan
e-mail: jhsu@ntnu.edu.tw

***Abstract*—In the field of visual odometry (VO) or SLAM, deriving camera poses from image features is the basic issue. Even though feature-based VO or SLAM are more efficient than non-feature-based methods, they are still unfortunately computationally demanding. This paper addresses the concerns of computational efficiency, computational resources and power-consumption problem of a VO algorithm by designing a hardware-software (HW/SW) co-design architecture for the implementation on a field-programmable gate array (FPGA) and a Nios II CPU. Given images from Nios II, features are extracted and matched by SIFT and linear exhausted search (LES) algorithms via hardware. The design of LES module is improved so that the speed is accelerated compared to our previous work. Subsequently, camera poses are estimated using an ICP algorithm, where the derivation of nearest orthogonal matrix is achieved by integrating Denman-Beavers (DB) approach and Taylor approximation method. As such, the required hardware resources are lesser. After hardware computations, the results are then transferred back to Nios II. To show the effectiveness of the proposed approach, experiments using KITTI dataset are conducted. The results show that, taking the advantages of efficient computation of hardware, the computational time is greatly reduced, compared to a full-software implementation. Moreover, usage of hardware resources are also lesser than existing methods.**

***Keywords-HW/SW Co-design; visual odometry; feature extraction and matching; FPGA, Nios.***

## I. INTRODUCTION

Concerning low cost, low power consumption and rich information, cameras are becoming the highly interested sensors for many autonomous systems. In the wake of increasing need for visual navigation, VO, which provides camera poses from two sequential images, is critical in any vision-based robotic systems. Because of its efficiency and reliability, feature-based VO is a preferable approach in the research communities. It derives camera poses geometrically from matched image features. Many feature-based VO use FAST [1] or SURF [2] algorithms for feature extraction, while KLT tracking [3] and FLANN [4] are often employed for feature matching. However, despite FAST and SURF can efficiently provide feature data, descriptions of those features can be unstable in scenarios where massive illumination changes or rotation changes appeared in the environments. As for KLT tracking and FLANN, they sacrifice robustness to reduce computational cost for matching, which could influence the accuracy of pose estimations. On the other hand, SIFT [5] and LES are considered the most robust feature extraction and matching algorithm, respectively. Yet SIFT features are encoded by 128 dimensions of descriptors, and LES is a brute-force matcher, they both suffer from heavy computational burden, limiting their applicability in real-time computer vision areas.

Based on matched features, algorithms such as PnP [6], essential matrix estimation [6], ICP [6], and so on, are responsible for providing 6-DOF camera poses. These approaches only requires linear algebra computation such as singular value decomposition (SVD), and the computational cost is relatively low. However, considering real-time processing, data delays from feature extraction and matching make a VO incapable of meeting up such a requirement. As a result, there are more and more approaches implementing a VO on hardware, or HW/SW co-design architecture. In fact, according to [7], using DSP, FPGA or GPU on computer vision is a desirable choice. A GPU+CPU heterogeneous design of a VO [8] is an example, which yields better performances than CPU-only approach. Nevertheless, extensive comparisons made in [7] show that FPGA provides outstanding advantages over DSP and GPU. Thus, HW/SW co-design using FPGA of a VO has been shown in the literatures. In [9], a visual-inertial odometry (VIO) was developed based on an embedded SoC using FPGA, aiming at optimizing the VIO algorithm to reach ultra-low latency and power. Another similar work [10] proposed an VIO algorithm-hardware co-design approach, trading off algorithm performance and hardware resource. It also incorporated software to obtain optimal parameters so that hardware resources could be minimized. Sadly, [9] and [10] require an inertial sensor. A HW/SW co-design architecture of a stereo VO was released in [11] for rover navigation on

978-1-7281-4740-6/19/$31.00 ©2019 IEEE

Mars. It built five distinct VO pipelines using different feature extractions and matching methods. The most reliable pipeline used Harris Corner detection and SIFT descriptors. Features are matched based both on similarities and image locations, and camera motion was estimated using absolute orientation. Though the use of hardware resources in [11] is not large, the speed and accuracy are yet to be improved.

One of the main advantages of using HW/SW Co-design architecture is that, since some operations are not easily to be implemented in a hardware, software can be prior to be involved in for performing those operations before full-hardware implementation is achieved. Moreover, it is beneficial to use software for verifying the performances of the proposed system. Potential applications of integrating the HW/SW design can also be carried off when real-time, low power-consumption are concerned.

In this paper, a HW/SW co-design of a VO using a FPGA and a Nios II is proposed. State-of-the-art FPGA-based SIFT [12] and LES [13] are employed, where the latter is improved by pipelining operations of data transfer and matching. As for camera motion estimation, ICP is involved in, and a novel hardware-implemented nearest orthogonal matrix search method is proposed to avoid using SVD. This is because FPGA-implemented SVD consumes lots of hardware resources and performs far from satisfactory if 50M Hz is used, as can be seen in [18]. The effectiveness of the proposed approach can be verified through KITTI dataset. Experimental results show that the acceleration speedup of the overall system is around 121 times faster than a general PC, while simultaneously providing accurate VO results. Also, the use of hardware resources is comparably lesser.

## II. Overview of the Proposed System

Fig. 1 shows the proposed HW/SW co-design architecture of the VO system. The red and blue dotted lines represent the software and hardware platforms, respectively. The hardware includes the on-chip Intel Cyclone and off-chip memories.

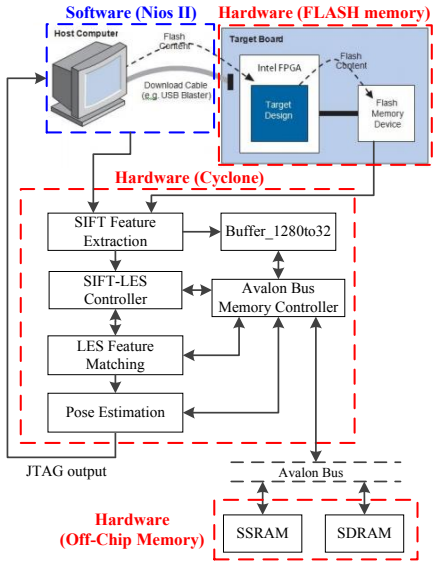


Figure 1. The architecture of the proposed HW/SW co-design VO.

In the beginning of the proposed system, images are written in a flash memory of a FPGA via the Nios II. Subsequently, Nios II reads data from the flash and transfer it to the FPGA so that SIFT can extract features from the images. In our previous work [13], FPGA-based LES algorithm was proposed, but 1280 bits of feature descriptors are stored in a SRAM, which is not a favorable design because it requires excessive memory capacity. Hence this paper stores feature data in off-chip SSRAM via Avalon Bus and its corresponding memory controller. Since SIFT feature descriptors are 128 dimensions with each dimension encoded by 10 bits, "Buffer_1280to32" module is therefore in charge of writing the 1280-bit descriptors 40 times sequentially to the 32-bit data width SSRAM. Image coordinates of features are provided by "SIFT-LES Controller", and the data are stored in the off-chip SDRAM. As matching starts, all feature descriptors of one image is prior to be stored in a SRAM such that as 32-bit descriptors of another image is transferred from SDRAM, matching can simultaneously be performed. With matched features, "Pose Estimation" module is responsible for deriving 6-DOF camera pose, which is achieved by ICP algorithm and a novel nearest orthogonal matrix search method. The VO estimations are then transferred back to the Nios II such that PC can be used to show the estimations or implement other applications.

## III. FPGA-Based Improved LES Algorithm

Different from our previous work [13], the proposed improved LES module is shown in Fig. 2.

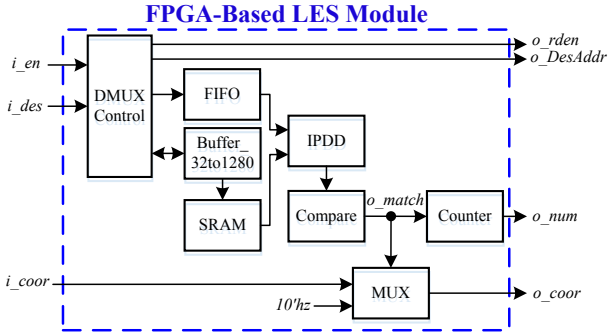


Figure 2. The architecture of the FPGA-based LES module.

As features are completely extracted by SIFT modules and stored in SSRAM, an enable signal *i_en* triggers the LES module. To match features, 32-bit feature descriptors *i_des* of one image are read from the SSRAM via Avalon bus according to the output signals *o_rden* and *o_DesAddr*. The former is the read enable of the SSRAM and the latter is the address of the descriptors in SSRAM, respectively. Subsequently, "Buffer_32to1280" module is responsible for collecting 32-bit data and forms a full 1280 bits feature descriptors. Then, the 1280-bit descriptor data is stored in a SRAM so that it can be constantly read for matching.

After a feature descriptor from one image is stored in the SRAM, feature descriptors from another images are sequentially read from the SSRAM through the same input. However, instead of using "Buffer_32to1280" module to form a full 1280-bit descriptor, the 32-bit data is stored in a

FIFO sequentially. The input and output data width of the FIFO is designed as 32-bit and 20-bit, respectively. Such a design is due to the fact that, since a SIFT has 128 dimension of descriptors with each dimension encoded by 10 bits, two dimensions of descriptors (20 bits) can be matched one at a time with the features stored in the SRAM, while the remaining 32-bit descriptors can be continuously read from the SSRAM and stored in the FIFO simultaneously. This pipeline operation not only accelerates the overall matching efficiency, but also prevents the massive use of hardware resources since 1280-bit data actually requires lots of logic elements. Moreover, because only two dimensions are required to be matched, Parallel Difference of Descriptors (PDD) module that employs reduction sum method in [13] is in fact not needed. The module is revised to Improved PDD (IPDD) module. Hence, compared to [13] where PDD takes 8 clocks to complete computing the difference of the two 1280-bits descriptors, IPDD only requires 3 clocks for the first pair of two dimensional descriptors, while the rest of the differences can be achieved one at a clock. As a result, as all descriptors of the second image is completely read from SSRAM, the matching process is also nearly completed. Consequently, the speed can be greatly improved.

Note that the remaining modules and the input/output signals are identical to [13]. Interested readers can refer to the reference for further investigations.

## IV. FPGA-Based Pose Estimation Module

### A. Architecture

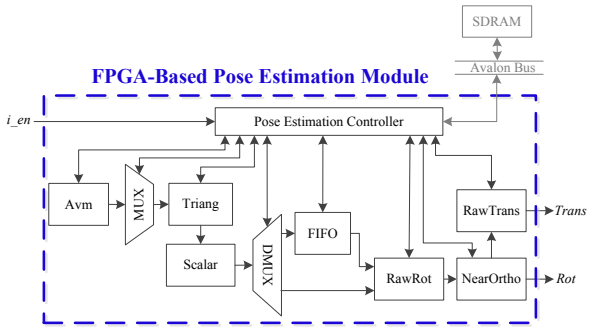


Figure 3. The architecture of the FPGA-based pose estimation module.

Fig. 3 shows the proposed architecture of pose estimation module, which is also designed in a FSM manner. As the module is enabled by the *i_en* signal, matched features are read from SDRAM via Avalon Bus Master "Avm", a multiplexer MUX is in charge of choosing either the matched feature data from the current image or the previous one. Since ICP algorithm is involved in the module for motion estimation, the 2D features are triangulated to 3D points using "Triang" module, and the "Scalar" module is used to scales 3D points so that matched 3D points of current and previous images are located in the same coordinate. Note that not until the current matched features be completely operated in the Scalar module do the previous matched features be read from the SDRAM and processed all the same way to the Scalar module. Therefore, the current scaled 3D points are required to be stored in a FIFO priorily so that matched features from both images can be operated in the subsequent modules synchronizely.

Based on the scaled 3D points, the RawRot module is responsible for deriving the initial rotation matrix from those points. This is simply achieved by the multiplication of the two 3D matched points formed by two matrices, i.e. $\mathbf{R}_{raw} = n^{-1}\mathbf{S}\mathbf{M}^T$; $\mathbf{R}_{raw}$ is the output of RawRot module, and matrices $\mathbf{S} \in \mathbb{R}^{3\times n}$ and $\mathbf{M} \in \mathbb{R}^{3\times n}$ are the 3D points; $n$ is the number of features. Because matrix multiplication has a property that the it can be accomplished by summing up several multiplications of submatrices of $\mathbf{S}$ and $\mathbf{M}$, i.e. $\mathbf{S} = \begin{bmatrix}\mathbf{S}_1 & \mathbf{S}_2 & \cdots & \mathbf{S}_p\end{bmatrix}$, $\mathbf{M} = \begin{bmatrix}\mathbf{M}_1 & \mathbf{M}_2 & \cdots & \mathbf{M}_p\end{bmatrix}$, $\mathbf{S}\mathbf{M}^T = \mathbf{S}_1\mathbf{M}_1^T + \mathbf{S}_2\mathbf{M}_2^T + \cdots + \mathbf{S}_p\mathbf{M}_p^T$, RawRot module is required to perform only $m$ points at a time. The $m$ points can be constructed respectively by $\mathbf{S}_q$ and $\mathbf{M}_q$, $q$ = 1,…,$p$. As such, when Scalar module outputs the previous matched 3D points, the RawRot module can concurrently multiply $3\times m$ submatrix one at a time priorily, where $m < n$. As a result, the Scalar and the RawRot modules are performed in a pipeline manner. Such a design accelerates the overall computational time.

After the RawRot module derives the initial rotation matrix, given that there are noises in the triangulations of 3D points, $\mathbf{R}_{raw}$ may not belong to special orthogonal group SO(3). Hence, NearOrtho module is in charge of finding the nearest orthogonal matrix of $\mathbf{R}_{raw}$. Having the property of orthogonality, the translation matrix can be derived based on $\mathbf{R}_{raw}$ and the gravity centers of the matched 3D points. The details of the ICP algorithm can be referred to [6].

### B. Novel Nearest Orthogonal Matrix Search Method and its Hardware Implementation

To meet the properties of special orthogonality for rotation matrix, almost all motion estimation algorithms use SVD to find the nearest orthogonal method. Even though there are numerous contributions implementing SVD on FPGA, the required hardware resources are unfortunately large. To avoid such a drawback, this paper proposes a novel algorithm for finding the nearest orthogonal matrix by first introducing the following derivation in [14],

$$\hat{\mathbf{A}} = \mathbf{A}\left(\sqrt{\mathbf{A}^T\mathbf{A}}\right)^{-1}, \quad (1)$$

where matrix $\hat{\mathbf{A}}$ is the nearest orthogonal matrix of $\mathbf{A}$. The tricky part of finding $\hat{\mathbf{A}}$ is the square root of $\mathbf{A}^T\mathbf{A}$. Fortunately there are various approaches that tackle such a problem. Among them, non-closed-form methods are found to be simpler to be implemented in hardware because only matrix addition, multiplication, and inverse are required. These methods include Newton's iterative algorithm, Meini's algorithm [15], Denman-Beavers (DB) algorithm [16], eigenvalue approach, and so on. Newton's method requires a good initial guess, and thus the solution is likely to diverge. Meini's and DB algorithms are preferable choices. They not only provide satisfactory convergence rate, the computations are also uncomplicated. To choose the

favorable solution for deriving the matrix square root of (3), investigating Meini's and DB algorithms can find that Meini's approach demands 4 matrix multiplications in each iteration, while DB approach does not need any multiplication. Instead, only matrix additions are required. Concerning that every signal is in integer form when implementing algorithms in FPGA, matrix multiplication is absolutely not preferable than matrix addition. This is mainly because integer arithmetic errors could be enlarged and accumulated during the multiplication operation, not mention that Meini's method is an iterative algorithm. Hence, we employ D-B method for the matrix square root, without the need of intricate floating point arithmetic in hardware.

However, according to numerical experiments reported in [15], Meini's method appears to be the fastest reliable iteration. It is true, that DB method does not converge as fast as Meini's. The more iterations it requires, the risks of accumulating integer arithmetic error rises. To fasten the convergence and prevents arithmetic errors, as soon as DB method derives the square root of $\mathbf{A}^T\mathbf{A}$ within a certain iterations and $\hat{\mathbf{A}}$ is provided such that $\hat{\mathbf{A}}$ is already nearly orthogonal, a Taylor series approach is employed. Let Y be the residual as shown below,

$$\mathbf{Y} = \hat{\mathbf{A}}^T\hat{\mathbf{A}} - \mathbf{I}. \tag{2}$$

Substitute (2) into (1), the approximation becomes,

$$\mathbf{A} = \hat{\mathbf{A}}(\mathbf{I}+\mathbf{Y})^{-1} = \hat{\mathbf{A}} - \hat{\mathbf{A}}\mathbf{Y}\left(\frac{1}{2}\mathbf{I} - \frac{3}{8}\mathbf{Y} + \frac{5}{16}\mathbf{Y}\mathbf{Y} - \ldots\right). \tag{3}$$

Usually the first two or three terms of (3) is sufficient to give a satisfactory result because the influences from the higher order terms are so small that they can be neglected. The above approach is similar to Levenberg-Marquardt optimization, which takes the advantages of gradient descent and Gauss-Newton method to fasten the optimization speed. That is, we first approach the nearest orthogonal approximation by (1) with DB method, followed by the Taylor approximation in (3). Moreover, since (3) only calls for a two multiplications along with a subtraction of matrices, the hardware resource and the computation efficiency can be reduced. The division by 2 in (3) can also be accomplished by a shift register, shifting only 1 bit right.

## V. EXPERIMENTS

### A. Experimental Setup

To verify the proposed hardware-software co-design of a VO system, Altera DE2i-150 Cyclone IV FPGA board is employed. Nios II is developed using the FPGA, and a general PC with an i7 CPU of 4.2 GHz as well as a 4.00 GB RAM is also involved in for performance comparisons. The PC performs not only the overall VO algorithm, but shows the results based on the data received from the Nios II. The performances of the improved LES module is compared with our previous work [13]. The hardware resources used by the proposed nearest orthogonal matrix module is compared with the FPGA-based SVD nearest orthogonal matrix search method. Images from the KITTI dataset [17] are used, and the accuracy of VO estimations are compared with the ground truths provided by the dataset in terms of RMSE. Investigating the speed, hardware resources usage, and required power consumption of the proposed system, comparisons are made with [10] and [11].

### B. Experimental Results

To show the speedup performances of the proposed approach, Table I and II respectively present the comparisons of the FPGA-based improved LES algorithm and the nearest orthogonal method, respectively. Both comparisons are provided with the implementation on a PC. To reach a fair comparison, Nios II is also involved in because its clock frequency is identical to FPGA, i.e. 50MHz.

TABLE I. PERFORMANCE COMPARISONS OF THE IMPROVED LES ALGORITHM IMPLEMENTING ON HARDWARE AND SOFTWARE PLATFORMS

| | | Platform | | |
|---|---|---|---|---|
| | | ***FPGA*** | ***Nios II*** | ***PC*** |
| Frequency | | 50 MHz | 50 MHz | 4.2 GHz |
| Features | | Image #1 and #2: 600 features | | |
| Time (s) | [15] | 0.00737 | 122.879 | 2.865 |
| | **Proposed** | **0.00622** | | |
| Multiplying Factor | [15] | × 1 | × 16677 | × 389 |
| | **Proposed** | × 1 | **× 19755** | **× 461** |
| Features | | Image #1 and #2: 700 features | | |
| Time (s) | [15] | 0.01 | 194.689 | 3.939 |
| | **Proposed** | **0.008** | | |
| Multiplying Factor | [15] | × 1 | × 19469 | × 394 |
| | **Proposed** | × 1 | **× 24330** | **× 492** |

TABLE II. COMPARISONS OF THE REQUIRED HARDWARE RESOURCE FOR NEAREST ORTHOGONAL MATRIX SEARCH METHODS.

| **Methods** | **Hardware Resources** | | |
|---|---|---|---|
| | ***LEs*** | ***Embedded Multiplier 9-bits elements*** | $f_{max}$ ***(Hz)*** |
| SVD-Based Method | 19581 | 166 | 85.2M |
| **Proposed** | 6954 | 49 | 225M |

As can be seen from Table I, the proposed improved LES module saves around 20% of the computational time of [15]. It accelerates approximately 500 times faster than PC when 700 features are used for matching, implicitly guaranteeing that the proposed technique reaches a real-time performance. In Table II, it is assertive to state that the proposed nearest orthogonal matrix search method demands around only 35.5% of the logic elements of SVD-based method. The frequency is also much higher, indicating that such a design and efficient use of hardware resource provides more potential for integrations with other modules.

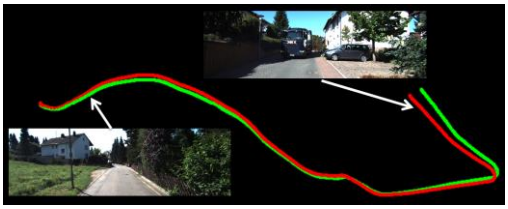

Figure 4. The estimated trajectory of the proposed HW/SW co-design VO.

Using the images from the KITTI dataset, the trajectory of the estimations given by the proposed HW/SW co-design

VO is presented in Fig. 4, where the red and green trajectories are the ground truth and the estimations, respectively. From the figure, it is clear that in large-scale environments, the proposed system is capable of providing accurate poses. The accuracy of the estimations in Fig. 4 in terms of RMSE are 3.462 (m) and 3.53°, respectively.

As for the overall HW/SW co-design architecture, the computational speed and usage of hardware resources are compared and presented respectively in Table III and IV. In the former table, we adopt computations of only 3 sequential KITTI images. The results show that the proposed approach speedup around 121 times faster than a PC, providing around 25 fps of the computational time. As for Table III, despite the power, the proposed system outperforms other methods. Nevertheless, it can also imply that using HW/SW co-design to perform a VO demands much lesser power than a PC, which generally requires roughly 70~80 W.

TABLE III. COMPARISONS OF COMPUTATIONAL SPPED

| | **Platform** | | |
|---|---|---|---|
| | ***FPGA (HW/SW)*** | ***Nios II*** | ***PC*** |
| Frequency | 50 MHz | 50 MHz | 4.2 GHz |
| Time (s) | 0.16 | 763.1 | 19.3 |
| Multiplying Factor | × 1 | ×4769 | ×121 |

TABLE IV. COMPARISONS OF REQUIRED HARDWARE RESOURCES AND POWER CONSUMPTION.

| | Methods | | |
|---|---|---|---|
| | [12] | [13] | **Proposed** |
| LEs / LUTs | 192K LUTs + 144K flip flops and 771 DSPs | 54K LUTs | 14K LEs |
| Memory | 2 MB | 1.46 MB | 1.32 MB |
| Power | 1.46 W | N/A | 2.25 W |

## VI. CONCLUSION

In this paper, a HW/SW co-design architecture of VO is proposed based on the state-of-the-art FPGA-based SIFT and LES methods. To improve our previous work, pipeline operation is employed so that the speed of the FPGA-based LES module is improved by around 20%. Moreover, taking advantages of pipeline processing and parallel computation, camera motion estimation is implemented in hardware, where a novel nearest orthogonal matrix search method is proposed to use hardware resources efficiently. The integration of DB method and Taylor expansion approach provides stable derivations of nearest orthogonal matrix. To verify the proposed approach, images from the KITTI dataset is used. Experimental results show that our system provides accurate pose estimation results, efficient performances, and lesser use of hardware resources.

## ACKNOWLEDGMENT

This paper is supported by Ministry of Science and Technology, Taiwan, under Grant MOST 108-2221-E-211-005, MOST 108-2634-F-003-002 and MOST 108-2634-F-003-003 through Pervasive Artificial Intelligence Research (PAIR) Labs..

## REFERENCES

[1] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 32, no. 1, pp. 105-119, 2008.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "SURF: speeded up robust features," Comput. Vis. Image Understand., vol. 110, no. 3, pp. 309–432, Jun. 2008.

[3] C. Tomasi, T. Kanade, "Detection and Tracking of Point Features," Tech. Rept. CMU-CS-91132, Carnegie Mellon University, 1991.

[4] M. Muja, and D.-G. Lowe, "Fast matching of binary features," IEEE Computer and Robot Vision, May, 2012, pp. 404-410.

[5] D.-G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, 2004.

[6] R. Hartley, and A. Zisserman, "Multiple view geometry in computer vision," Cambridge university press, 2003.

[7] A. HajiRassouliha, A.-H. Taberner, M.-P. Nash, and P.-N.-F. Nielsen, "Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms," Signal Processing: Image Communication, vol. 68, pp. 101-109, 2018.

[8] C.-H. Lin, W.-Y. Wang, S.-H. Liu, C.-C. Hsu, and C.-H. Chien, "Heterogeneous Implementation of a Novel Indirect Visual Odometry System," IEEE Access, vol. 7, pp. 34631-34644, 2019.

[9] D.-K. Mandal, S. Jandhyala, O.-J. Omer, G.-S. Kalsi, B. George, G. Neela, S.-K. Rethinagiri, S. Subramoney, L. Hacking, J. Radford, E. Jones, B. Kuttanna, and H. Wang, "Visual Inertial Odometry At the Edge: A Hardware-Software Co-design Approach for Ultra-low Latency and Power," IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), March, 2019, pp. 960-963.

[10] Z. Zhang, A. Suleiman, L. Carlone, V. Sze, and S. Karaman, "Visual-Inertial Odometry on Chip: An Algorithm-and-Hardware Co-design Approach," Robitcs: Science and Systems, 2017.

[11] G. Lentaris, I. Stamoulias, D. Soudris, and M. Lourakis, "HW/SW Codesign and FPGA Acceleration of Visual Odometry Algorithms for Rover Navigation on Mars," IEEE Trans. Circ. Syst. Video Technol., vol. 26, no. 8, pp. 1563-1577, 2016.

[12] S.-A. Li, W.-Y. Wang, W.-Z. Pan, and C.-C. Hsu, "FPGA-based hardware design for scale-invariant feature transform," IEEE Access, vol. 6, pp. 43850-43864, 2018.

[13] C.-H. Chien, C.-J. Chien, and C.-C. Hsu, "Hardware-Software Co-Design of an Image Feature Extraction and Matching Algorithm," IEEE International Conference on Intelligent Autonomous System, Singapore, 2019, pp. 37-41.

[14] N.-J. Higham, "Computing the polar decomposition—with applications," J. Sci. Stat. Comput., vol. 7, pp. 1160–1174, 1986.

[15] B. Meini, "The matrix square root from a new functional perspective: theoretical results and computational issues," Technical Report 1455., University of Pisa, 2003.

[16] E. Denman and N. Beavers, "The matrix sign function and computations in systems," Appl. Math. Comput., vol. 2, pp. 63-94, 1976.

[17] A. Geiger, P. Lenz, C.Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," Int. J. of Robotics Research, vol. 32, no. 11, pp. 1231-1237, 2013.

[18] M-.U. Torun, O. Yilmaz, and A.-N. Akansu, "FPGA, GPU, and CPU implementations of Jacobi algorithm for eigenanalysis," J. of Parallel and Distributed Computing, Vol. 96, pp. 172-180, 2016.