Fingerprint Local Invariant Feature Extraction on GPU with CUDA

Ali Ismail Awad
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology, Luleå, Sweden
E-mail: ali.awad@ltu.se
Faculty of Engineering, Al Azhar University, Qena, Egypt

Keywords: biometrics, fingerprint images, processing time, SIFT, SURF, GPU, CUDA

Received: January 24, 2013

Driven from its uniqueness, immutability, acceptability, and low cost, fingerprint is in a forefront between biometric traits. Recently, the GPU has been considered as a promising parallel processing technology due to its high performance computing, commodity, and availability. Fingerprint authentication is keep growing, and includes the deployment of many image processing and computer vision algorithms. This paper introduces the fingerprint local invariant feature extraction using two dominant detectors, namely SIFT and SURF, which are running on the CPU and the GPU. The paper focuses on the consumed time as an important factor for fingerprint identification. The experimental results show that the GPU implementations produce promising behaviors for both SIFT and SURF compared to the CPU one. Moreover, the SURF feature detector provides shorter processing time compared to the SIFT CPU and GPU implementations.

Povzetek: Predstavljen je nov algoritem prepoznavanja prstnih odtisov.

1 Introduction

Biometrics authentication is an emerging technology, and it is a crucial need for different applications such as physical access and logical data access. It compensates some weakness of the traditional knowledge-based and toke-based authentication methods [1]. Fingerprint image, shown in Figure 1 (a), is one of the dominant biometric identifiers that keeps populate for its uniqueness, immutability, and low cost. Due to the high demand on fingerprint deployments, it receives a great research attention in order to enhance the overall performance of the automated fingerprint identification system. The aimed enhancements include the reduction of the system's response time, and the improvement of the system's identification accuracy [2].

Unfortunately, the system's response time comes from the summation of the consumed times by a consequence of system operations, which are defined as fingerprint enrolment, pre-processing, feature extraction, and features matching. Therefore, enhancing the response time should be carried out by investigating each system phase [3]. The accuracy of the fingerprint identification system depends on reducing the inter-user similarity by accurately extracting distinctive and robust features to image shift and rotation. The local invariant features are robust to image scale, shift, and rotation. These qualified features can be extracted using some local invariant feature detectors [4].

A local feature of an image is usually associated with a change of an image property such as intensity, color, and texture [5]. The advantages of using the local features in fingerprint identification are that they are computed at multiple points in the image, and hence, they are invariant to image scale and rotation. In addition, they do not need further image pre-processing or segmentation operations [6]. The dominant local feature extractors are the Scale Invariant Feature Transform (SIFT) [7], and the Speeded-Up Robust Feature (SURF) [8]. Due to their robustness and their time efficiency, SIFT and SURF have been deployed in a broad scope of applications such as object recognition [9], texture recognition, and image retrieval [10], [11]. Thus, the two feature detectors have been selected for this research, and they have been applied on a standard fingerprint images database.

Graphic Processing Unit (GPU) is a 3D graphic accelerator that is available in most of the recent computers, and it runs equivalent to the CPU with better performance in parallel processing tasks. The GPU programming opens doors to image processing, computer vision, and pattern recognition algorithms to run efficiently compared to the CPU implementations [12]. GPU supports different computing languages with minimal code changes to port the previously developed algorithms to GPU. The Compute Unified Device Architecture (CUDA) [13] is one of the GPU computing languages that supports "C" and "Fortran" languages for efficient algorithms deployment [12].

The response time degradation becomes even worse in the system identification mode because the system needs to run (1:N) matching operations over a huge size fingerprint database. Some research contributions tried to reduce the identification time by classifying fingerprint database [14], [15], while some other researchers focused on the matching process, and proposed an efficient Matching Score Matrix algorithm (MSM) [10], [16] over a fingerprint database to

(a) (b) (c)

Figure 1: Fingerprint local feature extraction and matching: (a) Raw fingerprint image, (b) Extracted local features using the SURF detector, and (c) Fingerprint matching using the extracted local features. The number of matched features have been reduced for a clear representation purpose.

achieve better identification time reduction. This paper targets the feature extraction time and the matching time reductions as two factors for enhancing the overall system's response or identification time.

The contribution of this paper is a step forward toward reducing the fingerprint feature extraction and features matching time, and hence, enhancing the overall system's response time. The contribution includes the development of SIFT and SURF local feature detectors on both CPU and GPU for processing a whole fingerprint database. Moreover, the behaviour of SIFT and SURF on both CPU and GPU with focus on the number of extracted features, the feature extraction time, and the features matching time are considered. The reported results in this research can be used as a start point and ground truth for developing many GPU based fingerprint algorithms in the future.

The rest of this paper is organized as follows. Section 2 reviews the preliminaries backgrounds for the local feature extraction, and the graphic processing unit. Section 3 sheds lights on the implementation methodologies, and explains the exhaustive evaluations of both SIFT and SURF feature detectors deployed on the both CPU and GPU, in terms of processing time, number of features, and features matching. Conclusions and future work are reported in section 4.

2 Preliminaries

This section clarifies the local feature extraction, and highlights the different structures of the SIFT and the SURF feature detectors. Moreover, it covers the GPU architecture compared to the CPU one.

2.1 Local Feature Detectors

SIFT is one of the popular methods for image matching and object recognition. The SIFT feature detector has been used by some researchers in biometrics based authentication with applications on fingerprints [10] and palmprints [17]. Due to its reliability, SIFT features are used to overcome different fingerprint degradations such as noise, partiality, and rotations.

The SIFT feature detector works through sequential steps of operations. These steps can be summarized as: 1) Scale space extrema detection, 2) Keypoints localization, 3) Keypoint orientation assignment, and 4) Building the keypoints descriptor [18], [19]. The Difference-of-Gaussian (DOG) is used to detect the keypoints as the local extrema of DOG function. The DoG function is defined as [18]:

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma), \end{aligned} \quad (1)$$

where $I(x,y)$ is the input image at point (x,y) pixels, and $G(x,y,\sigma)$ is the variable-scale Gaussian function that is defined as:

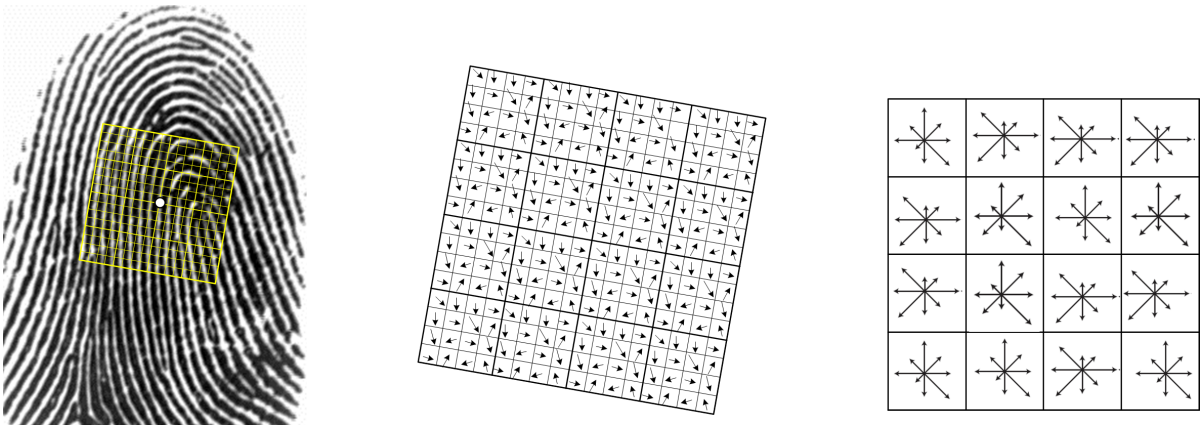
$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \quad (2)$$

and $L(x,y,\sigma)$ is the scale space of an image, and it is defined as:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y), \quad (3)$$

where $(*)$ represents a convolution operation.

The pixel is compared against 26 neighboring pixels (8 in the same scale, 9 in the above scale, and 9 in the below scale) to detect the local pixel extrema and minima. Following on, the detected keypoint is localized by setting its neighborhoods and examine them for contrast and edge parameters. The keypoints with low contrast and weak edge responses are then rejected. The keypoint neighborhoods region is used to build the histogram of the local gradient directions, and the keypoint orientation is calculated as the peak of the gradient histogram [11]. The default SIFT feature extraction produces keypoint associated with a descriptor of 128 element length. The descriptor is constructed from $(4 \times 4$ cells$) \times 8$ orientations [19]. The cascaded operations of building a single SIFT keypoint descriptor from fingerprint image, and the descriptor struc-



(a) (b) (c)

Figure 2: The process of building a single SIFT keypoint descriptor: (a) A single SIFT keypoint selected from fingerprint image, (b) 16×16 pixel gradients, and (c) A single 4×4 cells keypoint descriptor with 8 pixel orientations each. The default length of a single SIFT keypoint descriptor is $4 \times 4 \times 8 = 128$ element.

ture are shown in Figure 2. Applying SIFT feature extraction translates the fingerprint image into a set of keypoints according to the detected local maxima. The keypoint is associated with a descriptor related to the gradients of the enclosed pixels.

The SURF feature detector works in a different way from SIFT. The SURF keypoint detector uses Hessian matrix for keypoint extraction compound with the integral image to increase the computation efficiency. The Hessian matrix $H(x, \sigma)$ is calculated at a given point $p = (x, y)$ pixels of image I at scale σ as [8]:

$$H(x, \sigma) = \begin{bmatrix} S_{xx}(x, \sigma) & S_{xy}(x, \sigma) \\ S_{xy}(x, \sigma) & S_{yy}(x, \sigma) \end{bmatrix}, \tag{4}$$

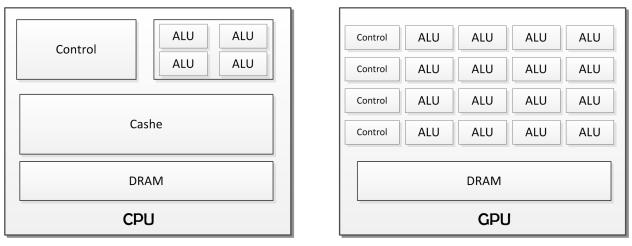
where $S_{xx}(x, \sigma)$ is the convolution $(*)$ of the Gaussian second order derivative with the image I in a point p.

The SURF descriptor is formed around the keypoints neighborhood by using Haar wavelet responses instead of gradient histogram applied in the SIFT. The standard length of the SURF descriptor can be 64 (4×4 subregions $\times$ 4 Wavelet components), 36, and 128 vector length. The SURF features are found to be an efficient compared to the SIFT one with preserved repeatability, robustness, and distinctiveness of the descriptors [8], [19]. Figure 1 (b) and (c) show the feature extracted using the SURF detector, and a sample of the local features matching process, respectively.

2.2 Graphic Processing Unit

The GPU is a multi-processor unit equipped with four types of memories that are defined as constant, texture, global, and shared memory for efficient data access. The GPU was initially designed for processing graphic functions, and it was required special skills for programming such functions via OpenGL [20]. The hardware architecture of the GPU is internally different from the CPU design. The two hardware design architectures are shown in Figure 3 (a) and (b) for the CPU and the GPU, respectively. The GPU architecture takes into its account the existence of many Arithmetic and Logic Units (ALUs) devoted for parallel processing and flow control [21].

The full power of the GPU architecture can be accessed via CUDA computing language as the threads are grouped into blocks, the blocks are grouped into grids, and the grid is executed as a single GPU kernel [20]. In real execution, the blocks are mapped to the GPU cores for efficient scalability. The CUDA computing language is designed to support general purpose computing on GPU. CUDA programming brings a development environment similar to "C" which is familiar to most of programmers and researchers. Additionally, minimal changes are required to port the CPU based code to the CUDA programming scheme, and be compiled using the provided NVCC compiler. Furthermore, CUDA introduces additional "C" libraries such as cuFFT for Fast Fourier Transform, and cuRAND for random number generation. Concisely, CUDA provides an integrated "C" similar environment for building the GPU based applications [13].



(a) (b)

Figure 3: The differences between the CPU and the GPU hardware design architectures: (a) The CPU hardware architecture, (b) The GPU hardware architecture with multiple Arithmetic and Logic Units.

3 Performance Evaluation

The open source computer vision library (OpenCV) [22] is a promising tool for developing most of computer vision and image processing algorithms. Recently, a new OpenCV module that provides a GPU acceleration, and covers the most significate part of the library was added to the OpenCV [20]. Although, the GPU module provides an implementation to the SURF feature detector, it does not provide the same implementation to the SIFT detector. Therefore, we have adopted, compiled, and worked with the SIFTS [23] as an open source library for evaluating the SIFT performance using the GPU implementation.

Prior to the evaluation process, OpenCV version 2.4.2 has been compiled with CUDA 4.2, the Threading Building Blocks (TBB) for the GPU, and the multi-core CPU supports, respectively. The local features matching was carried on the CPU using OpenCV `BruteForceMatcher` with `knnMatch` support. During the experimental work, the optimum `Knn` radius is set to 1 for the best matching performance.

3.1 Experimental Environment Setup

The experiments have been conducted using a PC with Intel® Core™ i3-2120 running at 3.30 GHz, and 8 GB of RAM. The PC is empowered by NVIDIA GeForce GT 430 GPU with 96 CUDA cores, and 1 GB of memory running on Windows® 64-bit. It is worth noticing that the quality of the extracted features is not considered in the present phase of this research.

The experimental work was applied on a standard fingerprint database, namely, the Fingerprint Verification Competition 2002 (FVC2002) [24] DB2_B subset. The FVC2002 DB2_B subset includes 80 fingerprint images come from 10 different persons. Therefore, the feature extraction using SIFT or SURF has been repeated 80 times. Whereas the matching process has been repeated over 80×80 images to produce a matching matrix with 6400 elements for evaluating the CPU based matching, and 80 elements for evaluating the GPU based matching with the average time.

3.2 Performance Evaluation of SIFT

According to the previous evaluation of the SIFT feature detector reported in [10], we set the optimum SIFT `threshold` to 0.01, and the other OpenCV SIFT parameters are set to the default values. The results of the SIFT evaluation on the CPU and on the GPU are shown in Figure 4. The plotted data are the average value of running the SIFT detector over the full database subset, and the matching time on the CPU is the average of 6400 matching processes.

The experimental results prove the significant reduction of the feature extraction time, and the features matching time when running the SIFT on the GPU. The time reduction comes from the powerful 96 CUDA cores supported by the NVIDIA GPU hardware compared to the two cores with two threads each supported by the CPU. The amount of the speed ups of running the SIFT detector on the GPU are 3.9X and 67.2X for the SIFT feature extraction, and the SIFT features matching, respectively.

The CPU and the GPU utilizations for a particular 14 seconds time period are drawn in Figure 5. From that figure, the CPU utilization starts low (around 25%) during the feature extraction phase, and it goes extremely high (around 98%) during the SIFT feature matching. On the other hand, the GPU load is fixed around 85% during the feature extraction and matching operations. Additionally, the figure gives an indication that is however, the matching time is extremely reduced, the full power of the GPU utilization still below the CPU one.

3.3 Performance Evaluation of SURF

The SURF feature detector has been evaluated on the CPU and the GPU using the same SIFT evaluation criteria, re-

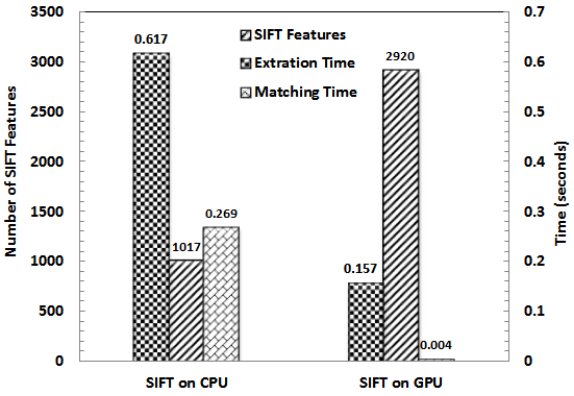


Figure 4: The evaluation of the SIFT detector on the CPU and the GPU with respect to the number of features, the extraction time, and the matching time.

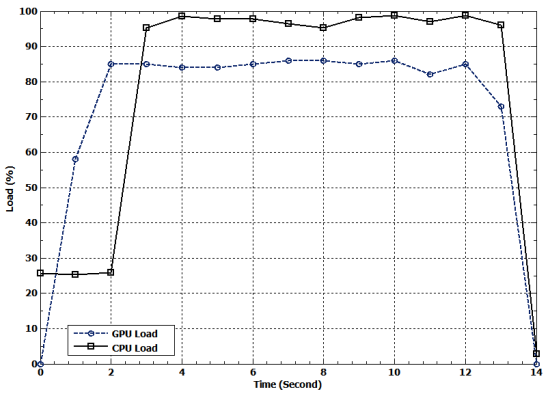


Figure 5: The CPU and the GPU utilizations of the SIFT detector measured in a 14 seconds long as a particular time slot.

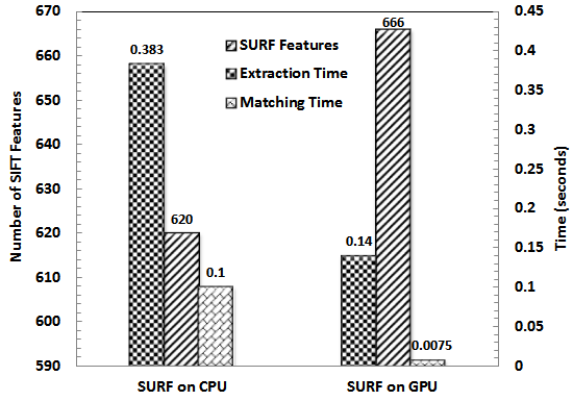


Figure 6: The evaluation of the SURF detector on the CPU and the GPU with respect to the number of features, the extraction time, and the matching time.

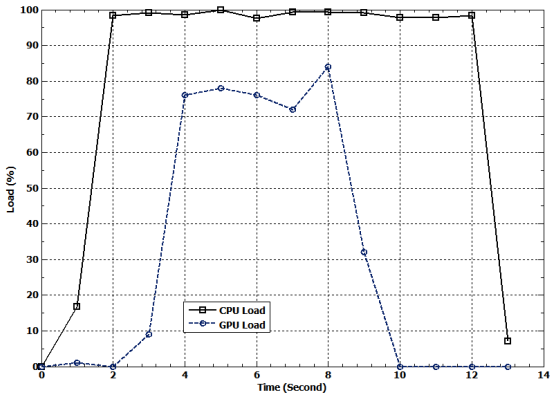


Figure 7: The CPU and the GPU utilizations of the SURF detector measured in a 14 seconds long as a particular time slot.

spectively. According to the evaluation results shown in Figure 6, the advantage of running the SURF detector on the GPU compared to the CPU is apparent. However, the number of extracted features on the CPU and the GPU are almost same. The feature extraction time on GPU is significantly reduced from 0.383 to 0.140 second. Moreover, the features matching time is reduced from 0.1 to 0.0075 second with almost the same number of features. The amount of the gained speeds up of using the GPU are 2.7X and 13.3X for the SURF feature extraction, and features matching.

Once again, the CPU and the GPU utilizations have been measured and reported in Figure 7. The SURF feature detector behaves very well on the GPU, and it consumes around 70 to 80% of the GPU power. On the other side, the SURF feature detector consumes around 98% from the total CPU power.

3.4 Discussion

The experimental work confirms the efficiency, and the superiority of the SURF feature detector compared to the SIFT feature detector. The evaluation factors are the feature extraction time, and the features matching time. From Figure 4 and Figure 6, the SURF feature detector runs twice faster than the SIFT one on the CPU due to the SURF enhancements such as using the integral images, Hessian matrix, and Haar Wavelet for keypoint detection and descriptor construction.

However, the SURF feature detector is optimized for a faster running on the CPU and the GPU; Figure 6 represents a higher matching time on the GPU (0.0075 second) compared to the SIFT (0.004 second). We do believe that the difference comes from the data transfer between the CPU and the GPU. The OpenCV implementation provides data upload and download between the CPU and the GPU for each matching process. This confirms the fact that the data transfer between the CPU and the GPU still a great challenge, and it should be avoided as much as possible for efficient GPU based implementations [20].

4 Conclusions and Future Work

This paper has presented a study for reducing the fingerprint identification time using two dominant local feature detectors, particularly SIFT and SURF implemented on CPU and GPU. The aim of the paper is to find the best implementation of a feature detector in terms of the number of features, the feature extraction time, and the features matching time. The experimental results proved the superiority of the SURF feature detector compared to the SIFT one. Moreover, the efficiency of using the GPU for both SURF and SIFT has been confirmed. However, SURF consumes longer matching time on the GPU, this time can be significantly reduced by avoiding the data transfer between the CPU and the GPU. Optimizing the SURF feature detector to work completely on the GPU, and avoiding the data transfer between the CPU and the GPU is considered as an appreciated future work. In addition, deploying the fingerprint related algorithms to work on the GPU is a common future direction.

5 Acknowledgments

We express sincere thanks to Professor Kensuke Baba, Kyushu University, and Ms Serina Egawa, Kyushu University, for providing the primary OpenCV source code for evaluating the SIFT detector on the CPU.

References

[1] Awad, A.I.: Machine learning techniques for fingerprint identification: A short review. In: Hassanien, A.E., Salem, A.B.M., Ramadan, R., Kim, T.h. (eds.) Advanced Machine Learning Technologies and Applications, Communications in Computer and Infor-

mation Science, Vol. 322, pp. 524–531. Springer Berlin Heidelberg (2012)

[2] Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S.: Handbook of Fingerprint Recognition, Second Edition. Springer-Verlag (2009)

[3] Egawa, S., Awad, A.I., Baba, K.: Evaluation of acceleration algorithm for biometric identification. In: Networked Digital Technologies. Part 2, Communications in Computer and Information Science, Vol. 294, pp. 231–242. Springer (2012)

[4] Jain, A.K., Ross, A.A., Nandakumar, K.: Introduction to Biometrics. Springer (2011)

[5] Mikolajczyk, K., Tuytelaars, T.: Local image features. In: Li, S., Jain, A. (eds.) Encyclopedia of Biometrics, pp. 939–943. Springer US (2009)

[6] Tuytelaars, T., Mikolajczyk, K.: Local invariant feature detectors: A survey. Foundations and Trends in Computer Graphics and Vision 3(3), 177–280 (Jul 2008)

[7] Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1150–1157 (1999)

[8] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding 110(3), 346–359 (Jun 2008)

[9] Baran, J., Gauch, J.: Motion tracking in video sequences using watershed regions and SURF features. In: Proceedings of the 50^{th} Annual Southeast Regional Conference. pp. 256–261. ACM-SE '12, ACM, NY, USA (2012)

[10] Awad, A.I., Baba, K.: Evaluation of a fingerprint identification algorithm with SIFT features. In: Proceedings of the 3^{rd} 2012 IIAI International Conference on Advanced Applied Informatics. pp. 129–132. IEEE, Fukuoka, Japan (September 2012)

[11] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(10), 1615–1630 (2005)

[12] Wynters, E.: Parallel processing on NVIDIA graphics processing units using CUDA. Journal of Computing Sciences in Colleges 26(3), 58–66 (January 2011)

[13] NVIDIA Compute Unified Device Architecture (CUDA), `http://www.nvidia.com/`

[14] Liu, M.: Fingerprint classification based on Adaboost learning from singularity features. Pattern Recognition 43(3), 1062–1070 (2010)

[15] Park, C.H., Park, H.: Fingerprint classification using fast Fourier transform and nonlinear discriminate analysis. Pattern Recognition 38(4), 495–503 (April 2005)

[16] Maeda, T., Matsushita, M., Sasakawa, K.: Identification algorithm using a matching score matrix. IEICE Transactions on Information and Systems E84-D(7), 819–824 (2001)

[17] Chen, J., Moon, Y.S.: Using SIFT features in palmprint authentication. In: Proceedings of the 19^{th} International Conference on Pattern Recognition. pp. 1–4. IEEE (2008)

[18] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)

[19] Saleem, S., Bais, A., Sablatnig, R.: A performance evaluation of SIFT and SURF for multispectral image matching. In: Campilho, A., Kamel, M. (eds.) Image Analysis and Recognition, Lecture Notes in Computer Science, Vol. 7324, pp. 166–173. Springer Berlin / Heidelberg (2012)

[20] Pulli, K., Baksheev, A., Kornyakov, K., Eruhimov, V.: Real-time computer vision with OpenCV. Communications of the ACM 55(6), 61–69 (June 2012)

[21] Poli, G., Saito, J.H.: Parallel face recognition processing using neocognitron neural network and GPU with CUDA high performance architecture. In: Oravec, M. (ed.) Face Recognition, pp. 381–404. In-Tech (2010)

[22] OpenCV: Open Source Computer Vision library, `http://opencv.willowgarage.com/wiki/`

[23] Wu, C.: SiftGPU: A GPU implementation of scale invariant feature transform (SIFT) (2012), `http://cs.unc.edu/~ccwu/siftgpu/`

[24] Maio, D., Maltoni, D., Cappelli, R., Wayman, J., Jain, A.K.: FVC2002: Second Fingerprint Verification Competition. In: Proceedings of the 16^{th} International Conference on Pattern Recognition (ICPR2002), Quebec City. pp. 811–814 (2002)