# An Embedded Solution for Multispectral Palmprint Recognition

Chao LI[1,2,*], Yannick BENEZETH[2], Keisuke NAKAMURA[3], Randy GOMEZ[3], Fan YANG[2]
[1]State Key Laboratory of Acoustics, Institute of Acoustics, Chinese Academy of Sciences, 100190 Beijing, Chine
[2]LE2I UMR6306 CNRS, Arts et Métiers, Univ. Bourgogne Franche-Comté, F-21000 Dijon, France
[3]Honda Research Institute Japan, 8-1 Honcho Wako-shi Saitamaken, 351-0188 Japan
Email: *chao.li.1986@ieee.org

***Abstract*—Palmprint based identification has attracted much attention in the past decades. In some real-life applications, portable personal authentication systems with high accuracy and speed efficiency are required. This paper presents an embedded palmprint recognition solution based on the multispectral image modality. We first develop an effective recognition algorithm by using partial least squares regression, then a FPGA prototype is implemented and optimized through high-level synthesis technique. The evaluation experiments demonstrate that the proposed system can achieve a higher recognition rate at a lower running cost comparing to the reference implementations.**

## I. Introduction

In the past decades, the personal authentication applications greatly benefited from the biometric technology for its advantages of convenience and accuracy. A series of biometric traits have been utilized and evaluated, such as face, fingerprint, iris, palmprint, finger-knuckle-print, hand geometry, etc. Palmprint is considered as a reliable biometric feature with high usability. It can greatly improve the performances of authentication systems, such as running speed, user friendliness, low cost, accuracy, etc [1]. Meanwhile, since the tissue of palm skin has different absorption capacity to different wavelengths of light and multispectral images capture more precisely these information, each band of multispectral images represents particular features of a plam. That allows to obtain multifarious information to improve the distinguishability of palmprint image features.

Recently, aimed at different processing cycles, including preprocessing, feature extraction, matching and decision cycles, many palmprint based multispectral biometric solutions have been developed. For example, within the feature extraction, Xu and Guo [2] represents the multispectral palmprint images as quaternion features extracted through the quaternion principal components analysis, and achieve better performance in recognition applications. Xu et al. [3] improve the multispectral palmprint recognition method in the feature extraction and matching cycles by using digital shearlet transform and multiclass projection extreme learning machine. Hong et al. [4] develop a hierarchical approach for multispectral palmprint recognition by fusing the block dominant orientation code and block-based histogram of oriented gradient features extracted from different light bands. According to the reported experiment data obtained within the laboratory environment, today's multispectral palmprint biometric algorithms have been able to provide high accuracy performance: average recognition rates of $99.9\%$ for the IITD palm database [5] and $99.56\%$ for the PolyU multispectral palmprint database [3].

In practical, finding a person of interest from a large candidate database is far from easy. It usually requires a high performance hardware platform to manage the candidate database and/or ensure the execution speed of the system. Furthermore, in some real-life applications, i.e. access control or e-banking, the authentication process needs to be made in a terminal equipment with real-time abilities due to the requirements about information security and application environment. For these issues, Zhang et al. [1] propose a low-cost multispectral palmprint system that can operate in real time and acquire high-quality images. It provides a high recognition accuracy by fusing the multispectral information at score level. However, the hardware device of this system is based on the CPUs without embeddability. Kumar and Shekhar [6] successfully prototype a multimodal palm biometric system into an Altera DE2-115 board from Terasic, but it achieves only an EER of $16.65\%$ and a verification speed of 0.8 seconds per image, which can hardly satisfy the accuracy performance or real-time requirement of the real-life applications. Consequently, finding an embedded solution for high accuracy multispectral palmprint biometric is still an open challenge and motivates the work of this paper as well.

Our work consists of three cycles: algorithm design, RTL (Register-Transfer Level) implementation and performance optimization. We first propose a new multispectral palmprint recognition algorithm. In the multispectral image based biometric technique, the information presented by multiple biometric measures need to be consolidated to perform a final decision by using the data fusion technique [7]. Usually, there are four levels of fusion in the multispectral biometric methods: image/pixel level, feature level, matching score level and decision level. We chose the score level fusion technique for its benefits shown in the research of Zhang et al. [1]. That is, the dissimilarities of the band maps of the input multispectral images are measured first within their own bands, and then the results are fused in the score level. Meanwhile, a partial least square regression based classifier is used for matching calculation. This regression method can effectively improve the test accuracy by modeling the relations between training and desired response matrices, as well as handle the classification problems. Further more, the regression model of this method is intrinsically a matrix multiplication operation, which is very fit for embedded systems.

Since conventional FPGA design flow cannot provide a

ISBN 978-0-9928626-7-1 © EURASIP 2017

software-friendly environment for the algorithm development and verification, we implement the design through a High-Level Synthesis (HLS) SW/HW co-design flow [8]. This techinique can effectively accelerate the design cycles by automate the C-to-RTL synthesis, even for the users with fewer register-transfer language programming experiences, as well as improve the maintainability of the design by facilitating the algorithm description. Furthermore, a series of optimizations are made in the code level to improve the design performances.

Finally, the proposed solution is evaluated using the Region Of Interest (ROI) version of the PolyU multispectral palmprint database [9]. The experiment results demonstrate a very high recognition rate, nearly 100%. Meanwhile, the proposed hardware implementation and optimization can effectively accelerate the processing efficiency. The final implementation achieves a recognition speed of 1.37 frames per millisecond (with a multispectral image resized into 13-by-13 and including normalization, feature matching, score fusion and decision), which allows to perform more complex pre- and post- processing in real-time, such as hand image segmentation, gesture analysis, hand language interpretation, etc..

The remainder of this paper is organized as follows: Section 2 describes the proposed palmprint recognition algorithm; Section 3 presents the implementation and optimization cycles; Section 4 analyzes the experiment results; finally, a conclusion is given in Section 5.

## II. Algorithm description

Fig. 1 shows the overall flowchart of the implemented palmprint recognition algorithm. Usually, the multispectral images are organised as multispectral cube arrays denoted by $C_i \in \mathbb{R}^{H\times W\times B}$ with $1 \leq i \leq N$, where $H$ and $W$ refer to the height and width of the images respectively, $B$ is the band number and $N$ is the number of multispectral palmprint images (known also as observed samples). Therefore the first step of the design is to reshape the set of multispectral image arrays into the form of $X \in \mathbb{R}^{N\times D\times B}$ with $D = H \times W$. Let $X_b \in \mathbb{R}^{N\times D}$ be the $b$-th frame of $X$ in the $B$ direction, so the rows and columns of $X_b$ respectively correspond to the samples and vectorized reflectance map (feature variable vector) within the $b$-th band.

PLS based classifications necessitate either a testing or a training data set. In the second step, every sub matrix $X_b$ is treated as a testing matrix of PLS and assigned to the concerned matching channel. Since the proposed algorithm consists of multiple independent regression models, we define the combination of each regression model and its pre-process cycle as a matching channel. Within each channel, the testing matrix $X_b$ is first normalised: $\bar{X}_b = (X_b - \mu_b)/\sigma_b$, where $\bar{X}_b \in \mathbb{R}^{N\times D}$ is the normalized testing matrix in the $b$-th band, $\mu_b$ and $\sigma_b$ are the mean and standard deviation of the training matrix $X_{train.b}$. Then, the matching score matrix of the $b$-th band, $\hat{Y}_b \in \mathbb{R}^{N\times N_r}$, can be obtained by multiplying $\bar{X}_b$ by its PLS regression coefficients $\theta_b \in \mathbb{R}^{D\times N_r}$: $\hat{Y}_b = \bar{X}_b \times \theta_b$. $\hat{Y}_b$ is a $N$-by-$N_r$ matrix, in which $N_r$ is the number of classes (palm candidates). In our case, the element of the matching score matrix, $y_b(i,j)$, is expected to be 1 when the $i$-th input sample matches with the $j$-th candidate palm, otherwise 0.

**Algorithm 1** Pseudocode of PLS regression algorithm

**Input:** training matrix $\bar{X}_{train.b}$, target outputs $\bar{Y}_r$, principal component number $k$

**Output:** regression coefficients $\theta_b$

1: initialization
2: **for** $i = 1, 2, \ldots, k$ **do**
3: $u_i \leftarrow$ first column of $\bar{X}^T_{train.b}\bar{Y}_r$
4: $u_i \leftarrow u_i/||u_i||$
5: **repeat**
6: $u_i \leftarrow \bar{X}^T_{train.b\_i}\bar{Y}_r\bar{Y}_r^T\bar{X}_{train.b\_i}u_i$
7: $u_i \leftarrow u_i/||u_i||$
8: **until** convergence
9: $p_i \leftarrow \bar{X}^T_{train.b\_i}\bar{X}_{train.b\_i}u_i/(u_i^T\bar{X}^T_{train.b\_i}\bar{X}_{train.b\_i}u_i)$
10: $c_i \leftarrow \bar{Y}_r^T\bar{X}_{train.b\_i}u_i/(u_i^T\bar{X}^T_{train.b\_i}\bar{X}_{train.b\_i}u_i)$
11: $\bar{X}_{train.b\_i+1} \leftarrow \bar{X}_{train.b\_i}(I - u_ip_i^T)$
12: **end for**
13: $\theta_b = u(p^Tu)^{-1}c^T$

The regression coefficient $\theta_b$ is obtained by using the Partial Least Squares (PLS) approach [10]. PLS regression is a machine learning algorithm that uses the covariance to guide the selection of features before performing least-squares regression. To maximize the covariance between the training input $X_{train.b}$ and desired matching score matrix $Y_r$, we first compute their normalization matrices $\bar{X}_{train.b}$ and $\bar{Y}_r$, then the largest singular value of $\bar{X}^T_{train.b}\bar{Y}_r$ is computed by using an iterative method proposed in [10], which repeats a sequence of steps shown in Algorithm 1. In this algorithm, $k$ is the number of principal component. This iterative process results in $u_i$ converging to the first right singular vector of $\bar{X}^T_{train.b}\bar{Y}_r$ and returns the regression coefficients.

After dissimilarity measurement, the matching score matrices of each channel $\hat{Y}_b$ are fused: $\hat{Y} = \sum_{b=1}^{B}\hat{Y}_b$, and finally the maximum element of each row of the fused matching score matrix, $\hat{Y}$, is used to perform the decision vector.

## III. Implementation and optimization

This work implements a 1-to-1000 ($N = 1$ and $N_r = 1000$) palmprint identification design for the PolyU multispectral palmprint database, which covers red, green, blue and Near-Infrared (NIR) bands. Algorithm 2 shows the pseudocode of the implemented algorithm. In this original version, PLS regression model and decision function (Line 14-17 and 21) are packaged into the sub functions to add to its readability. The algorithm behavior is first described by using C language, then synthesized into register-transfer level automatically with the help of Vivado_HLS (formerly AutoPilot from AutoESL), which is a leading HLS tool [11].

Despite of many benefits in terms of complexity, maintainability, development productivity, etc., it exists still a significant performance gap between HLS-based and manual register-transfer level implementations for complex applications in terms of time control, execution speed, consumption, etc. [12][13]. Consequently, we made a series of code-level optimizations to improve the performance of the original implementation. The optimization forms used include function inline, loop manipulation, pipeline and symbol expression manipulation.

ISBN 978-0-9928626-7-1 © EURASIP 2017

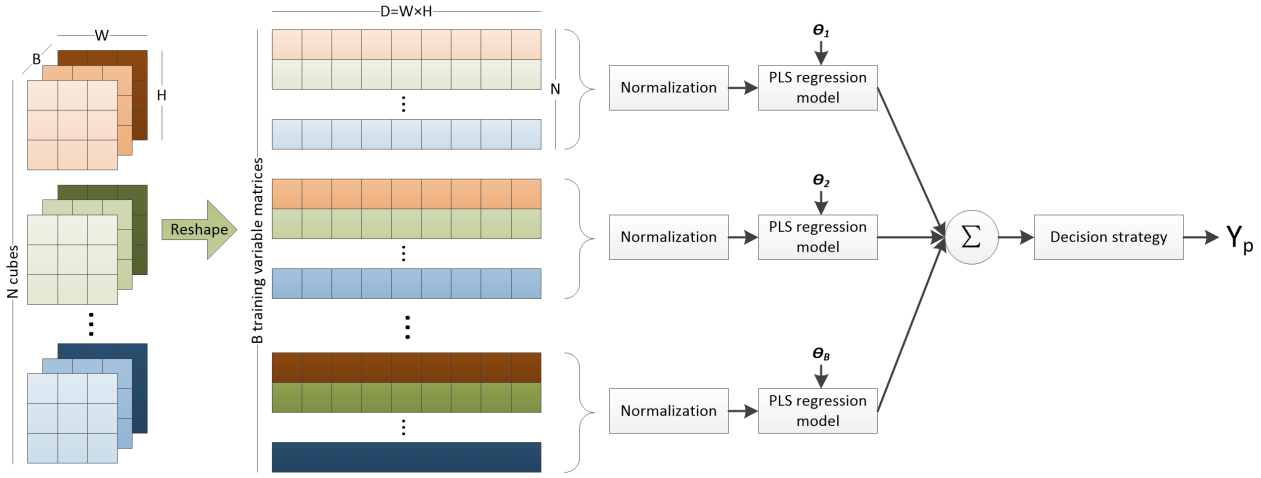


Fig. 1. Overall flowchart of the proposed algorithm.

**Algorithm 2** Pseudocode of the original algorithm behavior

**Input:** multispectral cube $X$, regression coefficients $\theta$
**Output:** decision $Y_p$

1: initialization
2: **for all** the feature values of the red band **do**
3: $\bar{X}_{red}(k) \leftarrow (X_{red}(k) - \mu_b)/\sigma_b$
4: **end for**
5: **for all** the feature values of the green band **do**
6: $\bar{X}_{green}(k) \leftarrow (X_{green}(k) - \mu_b)/\sigma_b$
7: **end for**
8: **for all** the feature values of the blue band **do**
9: $\bar{X}_{blue} \leftarrow (k) \leftarrow (X_{blue}(k) - \mu_b)/\sigma_b$
10: **end for**
11: **for all** the feature values of the NIR band **do**
12: $\bar{X}_{NIR} \leftarrow (k) \leftarrow (X_{NIR}(k) - \mu_b)/\sigma_b$
13: **end for**
14: $\hat{Y}_{red} \leftarrow plsreg\_model(\bar{X}_{red}, \theta_{red})$
15: $\hat{Y}_{green} \leftarrow plsreg\_model(\bar{X}_{green}, \theta_{green})$
16: $\hat{Y}_{blue} \leftarrow plsreg\_model(\bar{X}_{blue}, \theta_{blue})$
17: $\hat{Y}_{NIR} \leftarrow plsreg\_model(\bar{X}_{NIR}, \theta_{NIR})$
18: **for all** the elements of $\hat{Y}$ **do**
19: $\hat{Y}(i) \leftarrow \sum_{b=1}^{B} \hat{Y}_b(i)$
20: **end for**
21: $Y_p \leftarrow decision\_func(\hat{Y})$

**Algorithm 3** Pseudocode of the optimized algorithm behavior

**Input:** multispectral cube $X$, regression coefficients $\theta$
**Output:** decision $Y_p$

1: initialization
2: **for all** the feature values of all the bands **do**
3: *#pragma AP pipeline II=5*
4: $\bar{X}_{red}(k) \leftarrow (X_{red}(k) - \mu_b)/\sigma_b$
5: $\bar{X}_{green}(k) \leftarrow (X_{green}(k) - \mu_b)/\sigma_b$
6: $\bar{X}_{blue}(k) \leftarrow (X_{blue}(k) - \mu_b)/\sigma_b$
7: $\bar{X}_{NIR}(k) \leftarrow (X_{NIR}(k) - \mu_b)/\sigma_b$
8: **end for**
9: **for** the $i$-th element of $\hat{Y}$ **do**
10: *#pragma AP pipeline II=86*
11: $\hat{Y}_{red}(i) \leftarrow \bar{X}_{red} * \theta_{red}(i)$
12: $\hat{Y}_{green}(i) \leftarrow \bar{X}_{green} * \theta_{green}(i)$
13: $\hat{Y}_{blue}(i) \leftarrow \bar{X}_{blue} * \theta_{blue}(i)$
14: $\hat{Y}_{NIR}(i) \leftarrow \bar{X}_{NIR} * \theta_{NIR}(i)$
15: $tmp1 \leftarrow \hat{Y}_{red} + \hat{Y}_{green}$, $tmp2 \leftarrow \hat{Y}_{blue} + \hat{Y}_{NIR}$
16: $\hat{Y}(i) \leftarrow tmp1 + tmp2$
17: **if** $\hat{Y}(i-1) < \hat{Y}(i)$ **then**
18: $Y_p \leftarrow i$
19: **end if**
20: **end for**

In order to manipulate the loops in different function levels, the function hierarchy is first flattened. This transformation enables logic optimization across function boundaries and improve latency/interval by reducing function call overhead. Next, we manipulate the loops of the source code by using loop fusion and unrolling. HLS abstracts the input source code as a control and datapath flow graph, in which a sequence of successive operations is processed as a control step. Fig. 2-(a) shows the diagram of the control flow extracted from the function inline version of the proposed algorithm. In order to reduce the state and transit number, the loops of normalization computations (Lines 2, 5, 8 and 11) and loops containing in the PLS regression models (Line 14,15,16 and 17) are fused into a single one respectively due to the same loop boundary and independent bodies (Fig. 2-(b)). Furthermore, the initialization operations in $S_{11}$ is moved to the beginning of the input code and fused into the initialization step $S_0$. Finally, the loop of $S_2$ in Fig. 2-(b) is unrolled completely to parallelize its iterations (see Fig. 2-(c)). In our case, the transformation of loop manipulation can reduce the hardware consumption of logical control and add to the instruction level parallelism by centering the operations into a single control step.

The pseudocode of the optimized implementation is shown in Algorithm 3, in which two optimization directives (*#pragma AP pipeline*) are placed under the loops to perform iteration pipeline optimization. The factor $II$ is used to specify the desired initiation interval for the pipeline. Additionally, the expression of score level fusion (Line 19 in Algorithm 2) is segmented into short expressions (Line 15 and 16 in Algorithm 3). This transformation can enhance the detection ability of HLS tools in terms of Instruction-Level Parallelism.

ISBN 978-0-9928626-7-1 © EURASIP 2017

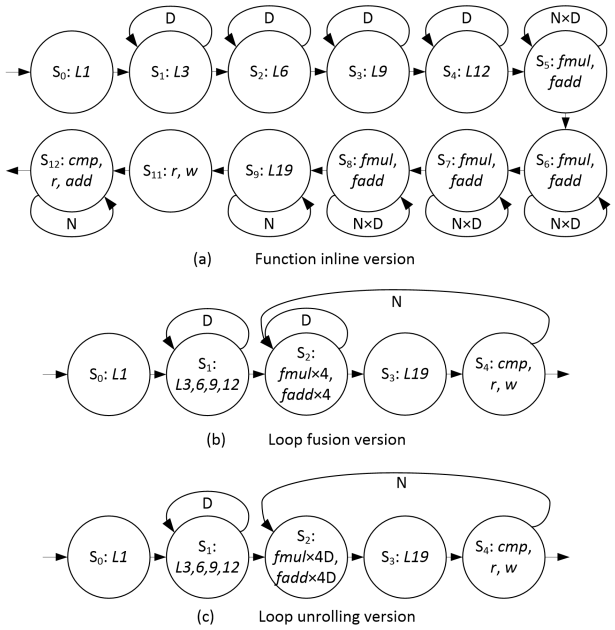


Fig. 2. Diagram of the finite state machine extracted from the proposed algorithm: $S_*$ is the state identification, $L*$ is the line number of the operations in Algorithm 2, and $fmul$, $fadd$, $cmp$, $r$ and $w$ are the multiplication, addition, comparison, reading and writing operators.

TABLE I. COMPARISON OF RECOGNITION RATE.

| Methods | Bands | Sample number (train vs. test) | Average recognition rate |
|---|---|---|---|
| MPELM [3] | NIR+red | 1 vs. 11 | 97.33% |
| | NIR+red | 3 vs. 9 | 99.56% |
| QPCA [2] | All | 6 vs. 6 | 98.13% |
| Proposed | NIR+green | 1 vs. 6 | 98.23% |
| | All | 1 vs. 6 | 99.96% |

## IV. EXPERIMENTS

This section presents the evaluation experiments of the proposed algorithm and its FPGA implementations. The ROI version of the PolyU multispectral palmprint database provided by Hong Kong Polytechnic University [1] is used for tests. This database is collected from 250 volunteers ($250 \times 2 = 500$ palms), including 195 males and 55 females, and captured with NIR and visible light (red, green and blue color). All the images are divided into two sessions, whose average time interval was about 9 days. For each session, 6 samples are acquired, so we have $6 \times 2 \times 500 = 6,000$ samples of $128 \times 128$ pixels in total.

Our experiments consist of accuracy and efficiency performance evaluations. For the accuracy evaluation, the recognition rate is used as metric, and the test is made with the smallest training set. The training set is created by choosing a single sample from the six samples of one session at random, then the other session for testing. The experiment is repeated 10 times in order to obtain an unbias result, in which the two sessions are used for training and testing alternately.

Table I compares the recognition accuracy of the proposed method with other two reference implementations, which are based on the Multiclass Projection Extreme Learning Machine (MPELM) [3] and Quaternion Principal Component Analysis (QPCA) [2]. We can see that our method can provide a similar accuracy performance compared to the other methods by using only two bands and a single sample for training. When all the four bands are applied, a very high recognition rate, nearly 100%, is achieved. It can be seen also that the case of MPELM [3] uses a larger test set than ours by mixing the two sessions. Considering that using the samples from the same session can facilitate the identity authentication problem, we conclude that our algorithm provides the best accuracy performance within the most challenging conditions.

TABLE II. HARDWARE CONSUMPTION COMPARISON.

| Components | BRAM | DSP | FF | LUT |
|---|---|---|---|---|
| Expression | - | - | 0 | 2877 |
| Instance | - | 40 | 3132 | 3638 |
| Memory | 16 | - | 0 | 0 |
| Multiplexer | - | - | - | 13300 |
| Register | - | - | 26197 | - |
| ShiftMemory | - | - | 0 | 16846 |
| Total | 16 | 40 | 29329 | 36661 |
| Utilization(%) | 5 | 31 | 65 | 81 |

The running speed performance of the design is evaluated using the device *xc5vfx70tff1136-1* of Xilinx with a clock cycle period of 8.34 ns, which is estimated by Vivado_HLS. Our experiment demonstrates that the original version of the proposed implementation achieves a running speed around $1.58 \times 10^{-2}$ seconds, and the running time of the optimized version is $7.3 \times 10^{-4}$ seconds, accelerating the design by $21.67\times$.

Finally, Tab. II presents the consumption of different components of the optimized implementation. Comparing to the original version, its average hardware utilization rate increases by around $5.88\times$, which is much lower than the acceleration ratio. That demonstrates that the applied optimization approach can effectively improve the implementation performance by using the additional area of the target device, and provide a high efficiency-area ratio. In additional, our implementation requires an external memory to save the test data and regression coefficients. For a $N$-to-$N_r$ system, its size can be estimated as follow: $(N + N_r) \times D \times S$, where $D$ is the number of the feature variables, and $S$ is the size of data type.

## V. CONCLUSION

This paper presents an embedded solution for real-time palmprint recognition applications. We first developed a new PLS regression based palmprint authentication algorithm, then implement it in FPGA with a series of optimizations. Experiments demonstrate that this embedded solution can provide a very high recognition accuracy as well as a high efficiency performance. Further more, comparing to the solutions based on the platforms of other types, i.e. the palmprint verification system specially designed for real-time applications by Zhang et al. [1] using C++ language in a PC with T6400CPU (2.13 GHz) and 2-GB RAM, our approach achieves a much higher running speed (1.5 vs. $7.3 \times 10^{-4}$ seconds). In the future work, we will further improve our solution by transplanting the training cycle of the proposed algorithm into the FPGAs and making more simulations and evaluations in deep in order to realize an autonomous, adaptative and portable biometric system.

ISBN 978-0-9928626-7-1 © EURASIP 2017

## REFERENCES


[1] D. Zhang, Z. Guo, G. Lu, L. Zhang, and W. Zuo, "An online system of multispectral palmprint verification," *IEEE Trans. Instrumentation and Measurement*, vol. 59, pp. 480–490, 2010.

[2] X. Xu and Z. Guo, "Multispectral palmprint recognition using quaternion principal component analysis," in *2010 International Workshop on Emerging Techniques and Challenges for Hand-Based Biometrics (ETCHB)*, Aug 2010, pp. 1–5.

[3] X. Xu, L. Lu, X. Zhang, H. Lu, and W. Deng, "Multispectral palmprint recognition using multiclass projection extreme learning machine and digital shearlet transform," *Neural Computing and Applications*, vol. 27, no. 1, pp. 143–153, 2016.

[4] D. Hong, W. Liu, J. Su, Z. Pan, and G. Wang, "A novel hierarchical approach for multispectral palmprint recognition," *Neurocomputing*, vol. 151, Part 1, pp. 511 – 521, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231214011692

[5] A. Kumar and S. Shekhar, "Personal identification using multibiometrics rank-level fusion," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 5, pp. 743–752, Sept 2011.

[6] M. Pudzs, R. Fuksis, R. Ruskuls, T. Eglitis, A. Kadikis, and M. Greitans, "Fpga based palmprint and palm vein biometric system," in *2013 International Conference of the BIOSIG Special Interest Group (BIOSIG)*, Sept 2013, pp. 1–4.

[7] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of Biometrics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[8] P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digital Circuit*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[9] Z. Guo, D. Zhang, L. Zhang, and W. Liu, "Feature band selection for online multispectral palmprint recognition," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1094–1099, June 2012.

[10] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[11] W. Meeus, K. Van Beeck, T. Goedemé, J. Meel, and D. Stroobandt, "An overview of today's high-level synthesis tools," *Design Automation for Embedded Systems*, vol. 16, no. 3, pp. 31–51, 2012.

[12] K. Rupnow, Y. Liang, Y. Li, D. Min, M. Do, and D. Chen, "High level synthesis of stereo matching: Productivity, performance, and software constraints," in *2011 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2011.

[13] Y. Liang, K. Rupnow, Y. Li, D. Min, M. N. Do, and D. Chen, "High-level synthesis: Productivity, performance, and software constraints," *Journal of Electrical and Computer Engineering*, vol. 2012, p. 14, 2012.


ISBN 978-0-9928626-7-1  