J Real-Time Image Proc (2018) 15:407–420
https://doi.org/10.1007/s11554-015-0494-6

ORIGINAL RESEARCH PAPER

Design and evaluation of a parallel and optimized light–tissue interaction-based method for fast skin lesion assessment

Chao Li · Vincent Brost · Yannick Benezeth · Franck Marzani · Fan Yang

Received: 16 June 2014 / Accepted: 24 February 2015 / Published online: 10 March 2015

Abstract In recent years, image processing technics have attracted much attention as powerful tools in the assessment of skin lesions from multispectral images. The Kubelka–Munk Genetic Algorithm (KMGA) is a novel method which has been developed for this diagnostic purpose. It combines the Kubelka–Munk light–tissue interaction model with the Genetic Algorithm optimization process, and allows quantitative measure of cutaneous tissue by computing skin parameter maps such as melanin concentration, volume blood fraction, oxygen saturation or epidermis/dermis thickness. However, its efficiency is seriously reduced by the mass floating-point operations for each pixel of the multispectral image, and this prevents the algorithm from reaching industrial standards related to cost, power and speed for clinical applications. In this paper, our work focuses on the improvement of this theoretical achievement. Therefore, we repropose a new C-based Parallel and Optimized KMGA (PO-KMGA) technique designed and optimized using multiple ways: KM model optimized re-writing, operation massively parallelized using POSIX threads, memory use optimization and routine pipelining with Intel C++ Compiler, etc. Intensive experiments demonstrate that our introduced PO-KMGA framework spends less than 10 min to finish a job that the conventional KMGA spends around two days to do in the same hardware environment with a similar algorithm performance.

Keywords Multispectral image processing · Kubelka–Munk model · Genetic algorithm · Function parallelization · High-performance computer · POSIX threads

C. Li (✉) · V. Brost · Y. Benezeth · F. Marzani · F. Yang
LE2I CNRS 6306 Laboratory, University of Burgundy, 21078 Dijon, France
e-mail: chao_li02@etu.u-bourgogne.fr

V. Brost
e-mail: vincent.brost@u-bourgogne.fr

Y. Benezeth
e-mail: yannick.benezeth@u-bourgogne.fr

F. Marzani
e-mail: franck.marzani@u-bourgogne.fr

F. Yang
e-mail: fanyang@u-bourgogne.fr

1 Introduction

The incidence of skin diseases has drastically stepped up in recent years. According to the World Health Organization, more than 2 million non-melanoma skin cancers and about 132,000 melanoma skin cancers are recorded every year, of which 48,000 related to deaths [3, 22]. It is also estimated that around 0.5–1 % of people is affected by vitiligo in the world. Although certain diseases are not medically harmful and will not cause much physical pain, they have always a strong psychological effect to the patients. Hence, the assessment of cutaneous lesions is a subject that increasingly attracts the medical researchers.

The diagnostic of cutaneous lesions is usually based on the analysis of the ambient light reflected by the skin surface. This re-emitted light carries important information about the physical and optical tissue parameters. Well-trained dermatologists analyze the skin color and interpret the clinical pathologies with the help of their knowledge and experience. In order to avoid the mistake due to the subjective judgment, some imaging systems could be used to assist clinicians. Meanwhile, the acquisition devices installed in these earlier systems are color cameras or dermatoscope which gives a color information of the skin with different scales thanks to the use of lens in dermatoscopy.

In order to produce an enhanced information for diagnostic, some novel sophisticated multispectral imaging processing methods emerged. To our knowledge, two approaches are usually used to analyze human skin reflectance spectrum. The first is based on statistical analysis of the reflectance spectrum, such as partial least squares regressions [37], Support Vector Machine (SVM) [24], Blind Source Separation (BSS) [23] or Independent Component Analysis (ICA) and Principal Component analysis (PCA) [34]. These techniques are based on composition assumption that assumes that skin reflectance is a combination of different source components' spectra weighted by their mixing quantities. The second is the analysis of the reflectance spectra by means of physical models of light transportation that are used on the optical properties of skin (scattering and absorption). Thanks to the efforts of other researchers, different light propagation models have been developed based on the modified Beer–Lambert law or Monte Carlo simulations. Kubelka–Munk is one of these models. It assimilates the parameter estimation problem as an optimization problem. That is, the properties of the model can be obtained from available apriori knowledge of the skin absorption spectra and scattering properties. Thus, comparing to the statistical analysis of the reflectance spectrum, this approach is not affected when the skin composition is different from the composition assumption.

However, Light–Tissue Interaction-based skin lesion assessment methods are very time consuming, because (a) the multispectral images are usually composed of more than three bands, which results in a large number of high-dimension vectors and (b) a long time is required to optimize the optical model function. Up to our knowledge, concerned researches achieved hardly any concrete results in terms of detection efficiency in the past few years. The medical image processing specialists could hardly seek out a more efficient optical model or function optimization algorithm for it without degrading the accuracy performance. For example, Shimada et al. [27] employed a modified Beer–Lambert law as a model for light propagation in skin, which requires only a short calculation time. However, this model is limited in terms of parameters and shows errors in visible wavelength extremity for the estimation of melanin. Meanwhile, parallel computing has made great progress, and many highly effective devices, such as CPU, GPU or MPoPC (Multi-core Processors on Programmable Chips), have been made available to engineers at a very convenient price. These achievements offer nice opportunities to increase the efficiency of complex designs. Moreover, hardware optimizations would not affect the accuracy of the targeted implementation because the basic algorithm is only a little bit modified during the development procedure. Therefore, the motivation of our work is to accelerate the Light–Tissue Interaction-based skin lesion assessment system by transplanting it onto a parallel computing platform.

Our introduced framework is based on the work of Jolivot et al. [18], in which a Kubelka–Munk Genetic Algorithm (KMGA) method is developed and proposed. KM is the name of the light–tissue interaction model combined with Genetic Algorithm (GA) for the optimization process. This innovative skin lesion assessment method is designed to retrieve five skin parameter maps from a multispectral image. In our previous efforts, the algorithm has been well implemented and verified using Matlab [17]. As expected, a common personal computer usually has to spend several days to retrieve a set of data from a standard multispectral lesion image with it. This shortcoming seriously hampers the practical application of this technology as an help for diagnostic which could be done by dermatologists or even general practitioners. Fortunately, previous researches have shown that GA-based designs can be optimized via different parallel computing environments [2, 13, 14, 36]. For KMGA, the huge quantity of data from multispectral images processing and GA's population information results in a bottleneck of memory consumption on GPUs and FPGAs, while multi-core CPUs have better overall properties, specially for efficiency and robustness performances. Hence, multi-core CPUs is selected as the hardware devices for high-performance KMGA implementation in this work.

This paper presents the experience of the Parallel and Optimized KMGA (PO-KMGA) development. After a detailed code source analysis, we have rewritten KM skin optical model in order to reduce the instructions number and speed up system execution. Our implementation also takes advantages of the POSIX (Portable Operating System Interface) to share the measured data and parallelize its computation at each pixel. We optimized as well the routine using Intel C++ Compiler and other program acceleration techniques. According to our experiments, PO-KMGA consumes only 5-6 minutes to finish the job that takes around 2 days with the previous module.

The remainder of the paper is organized as follows: Sect. 2 describes the fundamental principles of KMGA and analyses further its prototype's architecture, Sect. 3 introduces PO-KMGA method with multiple optimizations, Sect. 4 provides an experimental analysis and evaluation of the proposed implementation using a standard multispectral image, and finally, Sect. 5 concludes this paper and proposes some perspectives.

2 KMGA algorithm description

2.1 Kubelka–Munk model for light skin interaction

In order to retrieve the different skin physical or biological properties, several skin models have been developed [4, 25, 29]. Kubelka–Munk model [19] is one of the most popular and simplest approaches for computing light transport in a highly scattering medium and has been widely used to model the light–skin interaction. In our case, the skin is seen as a 2-layer medium (epidermis and dermis) and five principal parameters that affect the reflectance and the transmittance are included in the model: melanin concentration, epidermis thickness, blood concentration, blood oxygen saturation and dermis thickness. The total reflectance R_{tot} and transmittance T_{tot} are expressed as:

$$R_{\text{tot}} = R_{1,2} = R_1 + \frac{T_1^2 R_2}{1 - R_1 R_2} \tag{1}$$

$$T_{\text{tot}} = T_{1,2} = \frac{T_1 T_2}{1 - R_1 R_2} \tag{2}$$

The reflectance R_n and transmittance T_n for a single layer n can be expressed as a function of the thickness of the layer d_n, the absorption coefficient $\mu_{a,n}$ and the scattering coefficient $\mu_{s,n}$:

$$R_n = \frac{(1-\beta_n^2) \times (e^{K_n d_n} - e^{-K_n d_n})}{(1+\beta_n)^2 e^{K_n d_n} - (1-\beta_n)^2 e^{-K_n d_n}} \tag{3}$$

$$T_n = \frac{4\beta_n}{(1+\beta_n)^2 e^{K_n d_n} - (1-\beta_n)^2 e^{-K_n d_n}} \tag{4}$$

where:

$$K_n = \sqrt{\mu_{a,n}(\mu_{a,n} + 2\mu_{s,n})} \tag{5}$$

$$\beta_n = \sqrt{\frac{\mu_{a,n}}{\mu_{a,n} + 2\mu_{s,n}}} \tag{6}$$

In our case, the suffix n equals 1 or 2 for epidermis or dermis. The optical absorption coefficient of epidermis layer $\mu_{a.\text{epidermis}}$ is known as a function of concentration of melanin f_{mel}, the melanin spectral absorption coefficient $\mu_{a.\text{melanosome}}$ and the baseline skin absorption coefficient $\mu_{a.\text{baseline}}$:

$$\mu_{a.\text{epidermis}} = f_{\text{mel}}\mu_{a.\text{melanosome}} + (1 - f_{\text{mel}})\mu_{a.\text{baseline}} \tag{7}$$

where:

$$\mu_{a.\text{melanosome}} = 6.6 \times 10^{11} \lambda^{-3.33} \tag{8}$$

$$\mu_{a.\text{baseline}} = 0.244 + 85.3 \times \exp\frac{-(\lambda - 164)}{66.2} \tag{9}$$

On the other hand, the dermal absorption coefficient $\mu_{a.\text{dermis}}$ is expressed as follows:

$$\mu_{a.\text{dermis}} = f_{\text{blood}}(C_{\text{oxy}}\mu_{a.\text{oxy}}) + f_{\text{blood}}(1 - C_{\text{oxy}})\mu_{a.\text{deoxy}} + (1 - f_{\text{blood}})\mu_{a.\text{baseline}} \tag{10}$$

where f_{blood} is the blood concentration in %, C_{oxy} is the oxygen saturation in blood, $\mu_{a.\text{oxy}}$ and $\mu_{a.\text{deoxy}}$ are the absorption coefficients of the oxy-hemoglobin and deoxy-hemoglobin in cm^{-1}. The values of $\mu_{a.\text{oxy}}$ and $\mu_{a.\text{deoxy}}$ with the different wavelengths can be calculated from the Takatani–Graham table [16] using the following equations:

$$\mu_{a.\text{oxy}} = ln10 \times HbO_2(\lambda) \times G/M \tag{11}$$

$$\mu_{a.\text{deoxy}} = ln10 \times Hb(\lambda) \times G/M \tag{12}$$

where HbO_2 and Hb are the oxy-hemoglobin and deoxy-hemoglobin content in cm^{-1} l/mol, G is the hemoglobin's weight in gram per liter and M is the gram modecular weight of hemoglobin. The values of G and M can be calculated from the data presented in [31]. In our experiments, we selected 150 g/l and 64,500 g/mol as G and M. The scattering coefficients of the both layers $\mu_{s.\text{epidermis}}$ and $\mu_{s.\text{dermis}}$ are the sum of the Mie scattering coefficient $\mu_{s.\text{Mie}}$ and Rayleigh coefficient $\mu_{s.\text{Rayleigh}}$:

$$\mu_{s.\text{epidermis}} = \mu_{s.\text{dermis}} = \mu_{s.\text{Mie}} + \mu_{s.\text{Rayleigh}} \tag{13}$$

where:

$$\mu_{s.\text{Mie}} = 2 \times 10^5 \times \lambda^{-1.5} \tag{14}$$

$$\mu_{s.\text{Rayleigh}} = 2 \times 10^{12} \times \lambda^{-4} \tag{15}$$

2.2 Model inversion procedure with Genetic Algorithm

In the previous section, we discussed the principle of the KM skin model. According to Eqs.1–15, it is possible to express $R(\lambda)$, the total reflectance of the incident light with a certain wavelength, as a function of the skin parameters:

$$R(\lambda) = f_{\text{KM}}(f_{\text{mel}}, D_{\text{epi}}, f_{\text{blood}}, C_{\text{oxy}}, D_{\text{dermis}}) \tag{16}$$

where D_{epi} and D_{dermis} are the thickness of the epidermis layer and the dermis layer.

The reflectance values are measured with an acquisition system. In our experiments, we use the ASCLEPIOS system described in [18]. After a pre-processing step, the output of this system is a reflectance cube with two spatial dimensions and one spectral dimension. This cube can be seen as an image where each pixel is described by a vector, representing the reflectance spectrum of the skin as a function of the wavelength (see Fig. 1). From this measure, the objective is to find the five skin parameters for each pixel that are the most suitable for this measured reflectance spectrum. It is obvious that the KM model is a

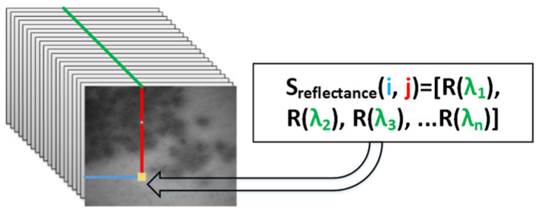

$S_{reflectance}(i, j)=[R(\lambda_1), R(\lambda_2), R(\lambda_3), ...R(\lambda_n)]$

Fig. 1 Reflectance spectrum $S_{\mathrm{reflectance}}$ at a single pixel formed from the reflectance cube measured: *blue* and *red* represent pixel's position and *green* represents the different wavelength values

non-linear function with five arguments. Several studies have been done about how to solve the similar complex non-linear function. The achievements of Viator et al. [35] and Choi [26] have proved that Genetic Algorithm (GA) is an effective approach.

In KMGA, a candidate solution (called individual) is composed of five skin parameters (f_{mel}, D_{epi}, f_{blood}, C_{oxy} and D_{dermis}), the spectrum generated by these parameters using Eq.16 and a fitness value as shown in Fig. 2. This last one depends on the similarity between the spectrum generated by the parameters and the measured spectrum. It can be expressed by a metric scale such as Root Mean Squared Error (RMSE). The set of individuals is called a population.

Figure 3 illustrates GA's overall implementation for the KM inversion. Firstly, N_P individuals are randomly generated within the range shown in Table 1 to form an initial population. Then, in an iterative framework, population evolves by techniques inspired by natural evolution. The evolution of the population is composed of three steps: best-individual selection, crossover mutation and random selection. During the selection process, only the best individuals are kept for the next iteration. Then, crossover process selects multiple couples of individuals (parents) from the population and one crossover parameter from the five skin parameters to create two new individuals (offsprings) by swapping the parents' selected parameter values. Finally, mutation process randomly generates some new skin parameter values to replace certain individuals' old ones. The aim of mutation is to keep exploring the search space and avoid being trapped into a local minimum. These processes are repeated until a predefined number of iterations. Finally, the best candidate is selected.

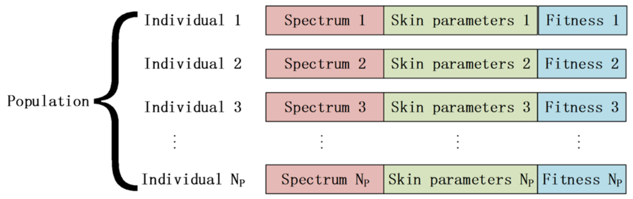


Fig. 2 Population data structure: the skin parameters are f_{mel}, D_{epi}, f_{blood}, C_{oxy} and D_{dermis}

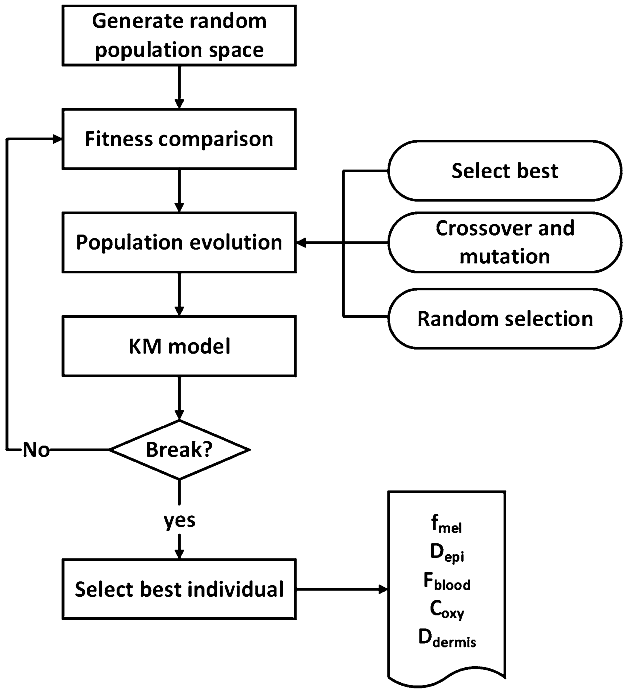


Fig. 3 Overall genetic algorithm procedure for KM model inversion

Table 1 Size of search spaces for skin parameters (cf. [18]).

Skin parameter	Symbol	Range
Melanin concentration	f_{mel}	1.3–43 %
Epidermis thickness	D_{epi}	0.01–0.15 mm
Volume blood fraction	f_{blood}	0.2–7 %
Oxygen saturation	C_{oxy}	25–90 %
Dermis thickness	D_{derm}	0.6–3 mm

3 Parallel and optimized KMGA design

The initial version of KMGA's prototype was built with the help of Matlab environment and is very time consuming. For example, a dual-core computer has to spend about two days in the skin parameter maps retrieving from a 328 $\times$ 270 image with 34 values per pixel for spectrum dimension. In this section, we design a novel Parallel and Optimized KMGA (PO-KMGA) with the following advantages:

- PO-KMGA's algorithm is more computationally efficient than before. We re-specify the KM algorithm and replace some of the operations by a look-up table in order to reduce the redundant operations down to minimum.
- PO-KMGA is implemented through a multi-threads shared-memory architecture that enables our design to

multiply the execution speed by parallelizing its computations.
- The random number generator of GA is improved in order to make the simulated results more reasonable.
- The data size is reduced, which improves the memory complexity.
- During the compilation, the operations of the source code are re-scheduled according to the pipelining strategy.

3.1 KM model analysis and optimization for computation complexity reduction

According to Fig. 3, KMGA method consists mainly in population initialization, population generation, and population evolution. Experimental results show that the population initialization and generation take up to 96 % of the total execution time, in which population evolution takes 3 % and other operations only 1 %. KM is the key technique used during the time consuming process of population initialization and generation. Thus, in order to speed up the KMGA method, we definitely have to improve its efficiency. We therefore proposed an optimized KM model using Matlab-to-C source translation.

KM function, the set of Eqs. 3–6, shows that the original model is complex and most parts of its instructions are costly and redundant. For example, the $e^{K_n d_n}$ term appears four times in Eq. 3. These redundant instructions are insignificant and costly. So, in our case it is absolutely possible to avoid them by arithmetical reducing. Thus we reform the KM function:

$$R_n = F_R(f_{\mathrm{mel}}, D_{\mathrm{epi}}, f_{\mathrm{blood}}, C_{\mathrm{oxy}}, D_{\mathrm{dermis}}) \tag{17}$$

$$T_n = F_T(f_{\mathrm{mel}}, D_{\mathrm{epi}}, f_{\mathrm{blood}}, C_{\mathrm{oxy}}, D_{\mathrm{dermis}}) \tag{18}$$

where F_R and F_T are optimized KM functions. These symbols are expressed as:

$$F_R = \frac{\mu_{s.n} \times (E-1)}{(\mu_{a+s} + K_n) \times E - (\mu_{a+s} - K_n)} \tag{19}$$

$$F_T = \frac{2K_n \epsilon}{(\mu_{a+s} + K_n) \times E - (\mu_{a+s} - K_n)} \tag{20}$$

where:

$$\mu_{a+s} = \mu_{a,n} + \mu_{s,n} \tag{21}$$

$$E = \epsilon^2 = e^{2K_n d_n} \tag{22}$$

The above rewritten KM functions are simple and effective to keep processors' obligations down to minimum, because reading the calculated data from local register, instead to repeat the computations, can provide a great gain of executive time. In our case, exp() is the most costly function, therefore we cut its repetitions down to only once per iteration by combining like terms and defining two interim storing registers E and ϵ. The other instructions are as well reduced more or less by similar approaches. Table 2 compares the necessary instructions number between the initial prototype and our implementation: the optimized function requires fewer instructions than before.

On the other hand, Sect. 2.1 presents that the optical absorption and scattering coefficients of epidermis and dermal layers are ultimately functions of wavelength and skin parameters. We can reformulate Eqs. (7, 10) and (13) using vector equations:

$$U_{a.\mathrm{epidermis}} = f_{\mathrm{mel}} U_{a.\mathrm{melanosome}} + (1 - f_{\mathrm{mel}}) U_{a.\mathrm{baseline}} \tag{23}$$

$$\begin{aligned} U_{a.\mathrm{dermis}} = & f_{\mathrm{blood}}(C_{\mathrm{oxy}} U_{a.\mathrm{oxy}}) \\ & + f_{\mathrm{blood}}(1 - C_{\mathrm{oxy}}) U_{a.\mathrm{deoxy}} \\ & + (1 - f_{\mathrm{blood}}) U_{a.\mathrm{baseline}} \end{aligned} \tag{24}$$

$$U_{s.\mathrm{epidermis}} = U_{s.\mathrm{dermis}} = U_{s.\mathrm{Mie}} + U_{s.\mathrm{Raylight}} \tag{25}$$

where:

$$U_{a.\mathrm{melanosome}} = 6.6 \times 10^{11} \times \begin{bmatrix} \lambda_1^{-3.33} \\ \lambda_2^{-3.33} \\ \lambda_3^{-3.33} \\ \vdots \\ \lambda_n^{-3.33} \end{bmatrix} \tag{26}$$

$$U_{a.\mathrm{melanosome}} = 0.244 + 85.3 \times \exp\left\{ - \left(\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_n \end{bmatrix} - 164 \right) / 66.2 \right\} \tag{27}$$

$$U_{a.\mathrm{oxy}} = ln10 \times \begin{bmatrix} HbO_2(\lambda_1) \\ HbO_2(\lambda_2) \\ HbO_2(\lambda_3) \\ \vdots \\ HbO_2(\lambda_n) \end{bmatrix} \times G/M \tag{28}$$

Table 2 Necessary instructions number comparison between original and optimized KM functions.

Model	$+/-$	$\times/\div$	x^y
Original KM	13	17	13
Optimized KM	8	9	3

$$U_{a.\mathrm{deoxy}} = ln10 \times \begin{bmatrix} Hb(\lambda_1) \\ Hb(\lambda_2) \\ Hb(\lambda_3) \\ \vdots \\ Hb(\lambda_n) \end{bmatrix} \times G/M \quad (29)$$

$$U_{a.\mathrm{Mie}} = 2 \times 10^5 \times \begin{bmatrix} \lambda_1^{-1.5} \\ \lambda_2^{-1.5} \\ \lambda_3^{-1.5} \\ \vdots \\ \lambda_n^{-1.5} \end{bmatrix} \quad (30)$$

$$U_{a.\mathrm{Raylight}} = 2 \times 10^{12} \times \begin{bmatrix} \lambda_1^{-4} \\ \lambda_2^{-4} \\ \lambda_3^{-4} \\ \vdots \\ \lambda_n^{-4} \end{bmatrix} \quad (31)$$

In Eqs. (23–31), $U_{a.\mathrm{melanosome}}$, $U_{a.\mathrm{baseline}}$, $U_{a.\mathrm{oxy}}$, $U_{a.\mathrm{deoxy}}$, $U_{a.\mathrm{epidermis}}$ and $U_{a.\mathrm{dermis}}$ are six coefficient vectors with different wavelengths. In our case, the waveband is fixed from 450 to 780 nm with the step of 10 nm. These vectors can be pre-calculated and stored in the memory. Thus, instead of calculating the values wanted at each iteration, the processor is able to read them directly from the memory. This approach avoids all the repeated computations due to Eqs. 8, 9, 11–15 in the routine.

3.2 Multi-threads KMGA parallelization

In order to achieve more effective use of hardware resources in a multiple cores environment, many parallel programming methods have been developed [8, 20, 28]. A lot of studies have demonstrated that the parallelization architecture can greatly improve the image processing prototypes' efficiency [5, 11, 12, 15, 33]. POSIX Threads (Pthreads) is one of the most popular parallel programming technologies for the UNIX-like operation systems. The POSIX standard defines an API (Application Programming Interface) for creating and manipulating threads [1]. Comparing to the other methods, Pthreads' advantages consist in the rapid thread creation [10], the shared global memory and the private data zone for each thread, so the Pthreads implementations may gain more potential improvement of the running-time and hardware cost performances.

In parallel programming, the first step is usually to find out the independent data flows in original algorithm. Fig. 1 illustrates that KMGA method analyses each single pixel's reflectance spectrum. That is, the algorithm sweeps all over the lesion zone pixel by pixel, and their data flows are absolutely independent with each other. This intrinsic nature of KMGA provides a nice parallelization potential.

Using the Pthreads technology, we designed a SPMD (Single Program Multi Data) parallelization as shown in Fig. 4: the multispectral image and retrieved skin parameter maps are stored in two allocated shared memory regions in floating-point format; PO-KMGA divides the multispectral image into N work areas and each of them is distributed to a single thread. Because all the data flows are independent, there are no interactions between two threads. Once the processing of a distributed reflectance spectrum is finished, the thread accesses to SM (Shared Memory) again with store address to write data into the right position and starts another operation. During the processing, parallel threads need only to traverse their own data segment using KMGA method.

In additional, given that a multi-core superscalar processor is selected as hardware platform, PO-KMGA scheduled the algorithm via a MIMD (Multiple Instruction Stream Multiple Stream Data Stream) strategy, which allows a superscalar CPU architecture to execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different functional units on the processor such as an arithmetic logic unit, a bit shifter, or a multiplier. For example, the computations of Eq.23 consist in two independent terms $f_{mel}U_{a.\mathrm{melanosome}}$ and $(1-f_{\mathrm{mel}})U_{a.\mathrm{baseline}}$, MIMD can therefore accelerate computations by scheduling their instructions in parallel rather than make the algorithm to run them one by one via a superscalar CPU architecture.

Moreover, POSIX threads can be automatically scheduled depending on the cores number in a out-of-order execution way by the compiler, therefore the developers have no need to consider the compatibility between the thread number implemented and core number to make processor run at full load all the time. This allows an effective use of processors resources and an important acceleration of PO-KMGA processing.

3.3 Parameters configuration of GA

As there is no golden rule to define the value of the GA different parameters, PO-KMGA's parameters configuration was performed empirically through series of tests. This subsection presents how the parameters of PO-KMGA are configured according to our previous work [17].

Population size

In Genetic Algorithm, the convergence condition changes with the population size. A large population ensures the diversity over the search space and limits the risk of convergence towards local minima, but the execution time for each generation is increased. In opposition, a small

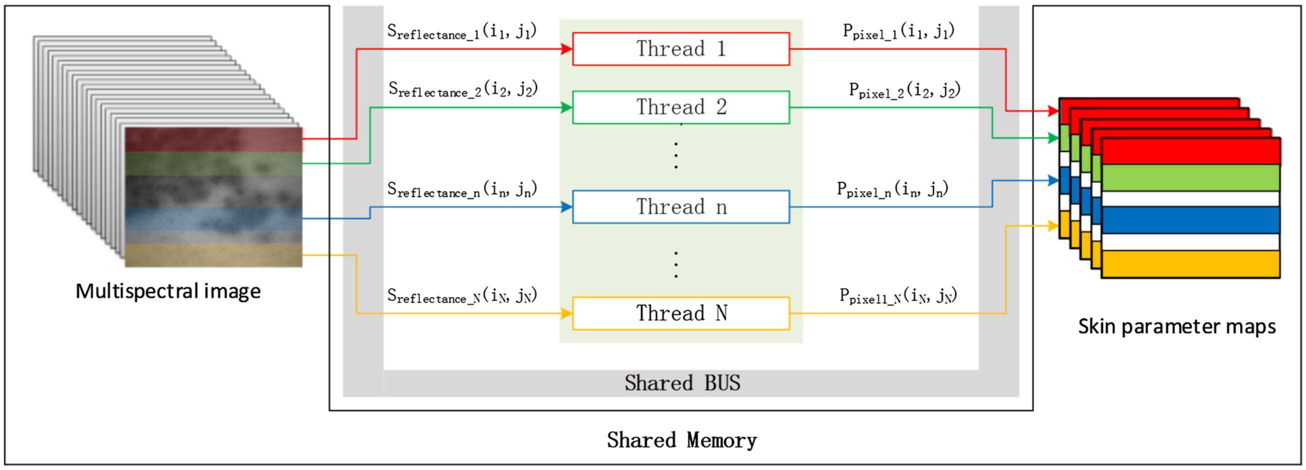


Fig. 4 N-threads PO-KMGA architecture using POSIX threads: $S_{\text{reflectance}\_n}$ is the reflectance spectrum at (i_n, j_n) in the work area of the nth thread and P_{pixel_n} is its skin parameters retrieved

population increases the risk to converge towards a local minimum but offers a higher execution speed.

Table 3 compares the experimental results of KMGA and PO-KMGA with five population sizes: 500, 300, 100, 50 and 25 for 25 iterations. According to the comparative result, the populations with 100, 300 and 500 individuals offer the similar fitness values both for the two implementations. Considering that a large population requires a huge computational time with only a slightly better precision (10.41E-4 for KMGA and 4.36E-4 for PO-KMGA when the population varies from 100 to 300 individuals), we select a population size of 100 individuals. The precision gained was not judged sufficient in comparison to lost in computational time.

Iteration number

Genetic Algorithm ends after a number of iterations, by comparing the fitness value with a satisfactory fitness level. Indeed, a higher fitness level (lower satisfactory fitness value) may improve the accuracy performance of the design and avoid falling into an local optimum (known as premature convergence), but it might lead to a huge computational time, the algorithm can even be trapped into an infinite loop.

In our case, instead of using a satisfactory fitness level, we set an evolution termination iteration number for PO-KMGA according to a series of tests. Fig. 5.20 of [17] illustrates the experimental results related to the convergence of a population composed of 100 individuals with 10, 25, 50 and 100 iterations. We noted that after around 20 to 25 iterations, the fitness value does not change and the algorithm is considered to have converged. Thus, KMGA selects 25 as the total number of iterations. Meanwhile, the experimental results shown in Fig. 5 compare the convergence rate of KMGA and PO-KMGA. It is found that a steady fitness value is obtained after 40 iterations. That is, PO-KMGA converges more slowly with the increase of the iteration number than the former. This is because the data precision falls from 64 bit down to 32 bit (*double* versus *float*) due to the memory use optimization effected in the C implementation (see Sect. 3.4), therefore we raise our implementation's iteration number up to 40 to ensure its accuracy.

Crossing and mutation rate

We select both crossing and mutation probability depending on the literature [30] ($P_c = 0.6$ and $P_m = 0.02$). As in our experiments, it is noticed that the convergence is not smooth and often temporarily increased with a mutation probability of 0.02, we select 0.03 as PO-KMGA's P_m.

3.4 Supplementary optimizations in PO-KMGA

Random sequence generator optimization

KMGA creates the random sequence according to Linear Congruential Generator (LCG) algorithm that yields a

Table 3 Average fitness value for different population sizes with an iteration number of 25.

Population size	KMGA average fitness values ($\times 10^{-4}$)	PO-KMGA average fitness values ($\times 10^{-4}$)
25	120.31	150.77
50	96.57	111.45
100	26.79	36.72
300	16.38	32.36
500	12.64	13.21

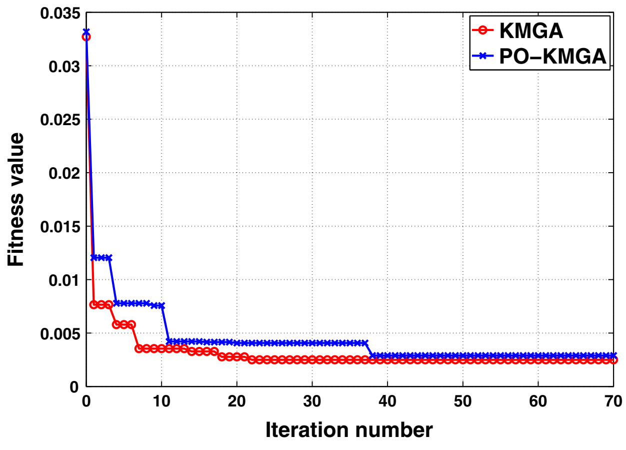


Fig. 5 Convergence of a population composed of 100 individuals with KMGA and PO-KMGA

sequence of pseudo-randomized numbers calculated with a discontinuous piecewise linear equation. This generator is defined by the recurrence relation:

$$X_{n+1} = (aX_n + c) \bmod m \tag{32}$$

where X is the sequence of random values, $m(m > 0)$ is the modulus, $a(0 < a < m)$ is the multiplier, $c(0 \le c < m)$ is the increment and $X_0(0 \le X_0 < m)$ is the start value (also called seed). This method does not consumes many hardware resources and executes fast. But as all the other pseudo-random generators, it works via a fixed algorithm, which results in changeless skin parameters every time the routine is run. Thus, LCG alone cannot meet completely the randomicity requirement of genetic algorithm.

A popular solution to this problem is to initialize the random generator with different seed values. In our implementation, the system's start time is used as seed to initialize LCG. Thus comparing to KMGA, PO-KMGA can provide better randomicity performance relative to population evolution and its simulation results are more close to actualities.

Memory use optimization

PO-KMGA's memory use is optimized by changing the data type and saving the memory allocations. KMGA defines the individual as a $N \times 3$ array in double-precision floating-point format. The three columns of this array refer separately to the reflectance spectrum, skin parameter and fitness vectors shown in Fig. 2. Thus, although the skin parameters and fitness vectors' sizes are only 5×64 bits and 1×64 bits, the system still allocates an $N \times 64$-bit memory region to each. In order to make an effective use of memory space, the population space is defined as a $(34 + 5 + 1) \times 32$-bit floating-point vector in PO-KMGA. The allocated memory's size is perfectly able to meet a single individual's memory capacity requirement. Thus, our approach provides a better data storage performance.

Furthermore, this optimization accelerates the designs by reducing the memory access conflicts and making more effective use of memories. Given that the memory architecture of multi-core CPUs is hierarchical, and higher hierarchy has faster access speed, processors preferentially accede to local cache to fetch the wanted data. But due to the production cost, their storage space is limited. PO-KMGA stores all the spectral cube information in a shared memory and each core accedes to it to read the wanted data, and then these data are temporarily saved in the local cache. Obviously, the reduced data size makes local cache of each core to load more information than before; some of the information may be reused by different operations, therefore larger local data quantity accelerates the running time of the system by reducing the access times to the slower shared memory. With the reduction of access times, the access conflicts are gotten down as well. Thus, the use of shared bus is also optimized.

Optimizations with ICC

In order to ease the developers' workload, certain Compiler-based optimization methods are developed to help the users to parallelize or pipeline their routines automatically [6, 7]. For example, the Intel C++ Compiler (ICC) can analyze and vectorize the data flow in order to benefit from SSE (Streaming SIMD Extensions) instructions. It optimizes also the loops by helping OpenMP (Open Multi-Processing) or auto-vectorization to make effective use of caches and memory access [9]. During PO-KMGA's compilation by ICC, "Maximize speed" mode is used to schedule the algorithm within a multi-threads framework and improve our implementation's running-time performance from the low level of the whole system.

4 Parallel and optimized KMGA evaluation experiments

In this section, we firstly analyze the implementation results and perform an unbiased comparison between PO-KMGA and other two different KMGA versions: Matlab-KMGA and C-KMGA. Matlab-KMGA is the earliest version of KMGA implemented in Matlab by Jolivot et al. [18], while C-KMGA is its Matlab-to-C transforming version without any optimizations. Then we transplant PO-KMGA from CPU onto FPGA to evaluate the performance differences with different hardware platforms. Our experiments have been realized using a Dual-Core CPU P6200 (Intel Pentium(R) CPU P6200 @ 2.13 GHz × 2) and a Quad-Core CPU Q6600 (Intel Core$^{\text{TM}}$ Quad CPU Q6600 @ 2.40 GHz

$\times$ 4) with ICC 13.1.1 in 32-bit in Ubuntu 13.04 system. The FPGA implementation is simulated with Zynq-7000 development board of Xilinx in Vivado_HLS 13.1. The experimental specifications are displayed by Table 4. In order to ensure the accuracy of PO-KMGA, its iteration number is set to 40 rather than 25 relative to its prototype of Matlab (see Subsection Iteration number in Sect. 3.3).

4.1 Individual optimization methods evaluations

In this subsection, we evaluate the effects of all the running-time optimizing methods including model re-writing (RW), memory optimization (MO) and multi-threads optimization (MT). We set the running-time of C-KMGA with P6200 as reference, then individually optimize it via these methods one by one and test them with both of the CPUs. For the MT implementations, the thread numbers are equal to the core number of the used platform. The acceleration ratio comparison results shown in Fig. 6 demonstrate that the proposed approaches make more effective use of hardware resources.

The first pair of column in Fig. 6 indicates that there is a difference between the two test results of C-KMGA. This is caused by the performance difference between P6200 and Q6600. Next, model re-writing provides about a 3.5$\times$ speedup on average. KM model takes up approximately 96 % of the total running time according to our tests, so simplifying its computation provides a quite high gain in speed comparing with the other optimizations. Thirdly, for the memory use optimizing, the acceleration ratios of memory optimizing by data type transferring are limited to the data size ratio between before and after optimizations in theory. In our case, only part of the data is transferred from double to float, so the acceleration from memory optimizing is about 1.5 times and lower than 2 (64/32 bits) times. Finally, POSIX multiples the running speed depending on the cores and threads numbers (see the next subsection for a further discussion).

Table 4 Experimental parameter configuration

Parameters	Matlab/C-KMGA	PO-KMGA
Population size	100	100
GA iteration number	25	40
Best individuals	10	10
Random individuals	30	30
Crossing individuals	30 pairs	30 pairs
Mutation individuals	3	3
Multispectral sample number	34/pixel	34/pixel
Resolution	328 $\times$ 270	328 $\times$ 270
Thread number	1	1, 2 and 4

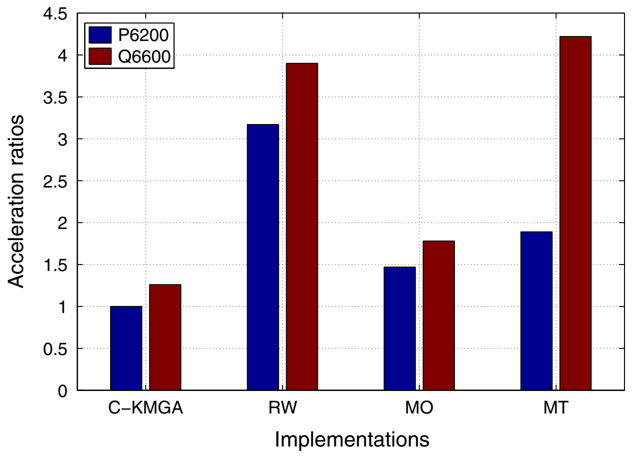


Fig. 6 Individual evaluation of the used optimization methods

4.2 Performance comparisons of final implementations

4.2.1 Running-time comparison

We firstly test the PO-KMGA implementation with different thread numbers using the two CPUs. The running-time test results shown in Fig. 7 indicate that Q6600 provides a much better performance in term of speed than P6200 for the single thread implementation.

Theoretically, the 2N-thread application should have doubled the execution speed comparing to the N-thread one. But due to the thread creation and management, extra clock periods are consumed in the multi-threads systems. For example, according to the test results, our implementation with 4 threads in parallel achieves only an acceleration of around 1.49$\times$ comparing with double threads in the same Q6600 CPU. Meanwhile, it is also found that there are no more gains in speed while the thread number exceeds the core number for P6200. This is because the instruction number that can be parallelized is limited by the processor's cores number.

In order to finish the job as soon as possible, the threads have to contend for the system resources against each other. That is, the cores are rotated among the different threads. Comparing to run the threads one by one in a single core, the advantage of this strategy is that the time wastes due to waiting state are avoided by the out-of-order execution among the threads. However, the system has to consume extra resources in control flow to manage the threads. As shown in Fig. 7, the running-time of four-thread implementation with P6200 is a little higher than its two-thread implementation. And finally, it is concluded that the best efficiency is achieved only while the threads number equals to the core numbers.

We compare as well the running-time performances of PO-KMGA with C-KMGA using the same multispectral

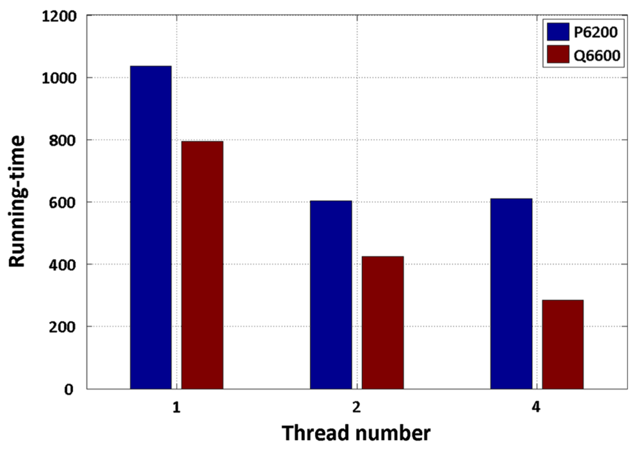


Fig. 7 Running-time test results with different thread numbers (in seconds)

image with different resolutions. The comparative results illustrated in Table 5 demonstrate that with the proposed methods, PO-KMGA further improves C-KMGA and achieves an acceleration of 17.11× for C-KMGA with a standard multispectral image. It is also found that the acceleration ratios increase with the image resolutions, i.e. the acceleration ratio of PO-KMGA/Matlab-KMGA with a 328 × 270 image using Q6600 is around two times higher than the one obtained with of a 18 × 15 image, though they should be same in theory. This is because the memory using optimization leads to a more effective use of CPU cache for high-resolution images. Hence, relative to original Matlab or C implementations, the more the data need to be computed, the more is the speed improvement achieved by PO-KMGA.

4.2.2 *Accuracy comparison*

The five skin parameter maps retrieved by PO-KMGA are shown in Fig. 8c. Comparing with Fig.8b, the two simulation results subjectively lead to a similar visual effect. In order to get an unbiased conclusion, this difference is quantified using RMSE (Root Mean Square Error) formulated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{W \times L}\sum_{i=1}^{i=W}\sum_{j=1}^{j=L}(I_{i,j} - K_{i,j})^2} \tag{33}$$

Table 5 Acceleration ratios of PO-KMGA/C-KMGA with different image resolutions.

	18 × 15	36 × 30	65 × 54	109 × 90	328 × 270
P6200	3.51	3.92	4.76	6.23	8.46
Q6600	8.72	9.11	11.12	13.17	17.11

where W and L are the widths and lengths of the images, and I and K refer to the two comparative maps. The computing results are shown in Table 6.

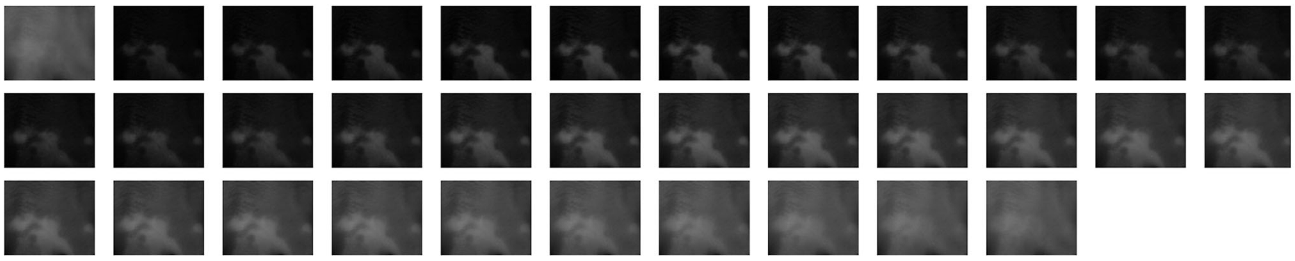
We should note that RMSE measures only the similarity between the two simulation results, but could not point out which one is the best, because the skin parameter maps of Matlab-KMGA are simulated results as well as PO-KMGA, and could not be seen as a reference-like measured map. Thus, the average fitness of the PO-KMGA implementations is compared with the one of KMGA's prototype in Fig. 9. We can find that the KMGA prototype has lower fitness values than PO-KMGA. That is, the skin parameters retrieved by KMGA are more close to the lesion zone's real physical and chemical properties than PO-KMGA. This is because PO-KMGA's data precision is lowered in order to optimize the memory use performance (see Sect. 3.4). However, the values of PSNR (Peak Single-to-Noise Ratio) for the retrieved maps vary from 20.29 to 31.43 dB, which are all within the typical range for PSNR values (between 20 and 25 dB, where higher is better [21, 32]). Therefore, we conclude that this accuracy loss is acceptable for practical applications.

4.3 CPU vs FPGA

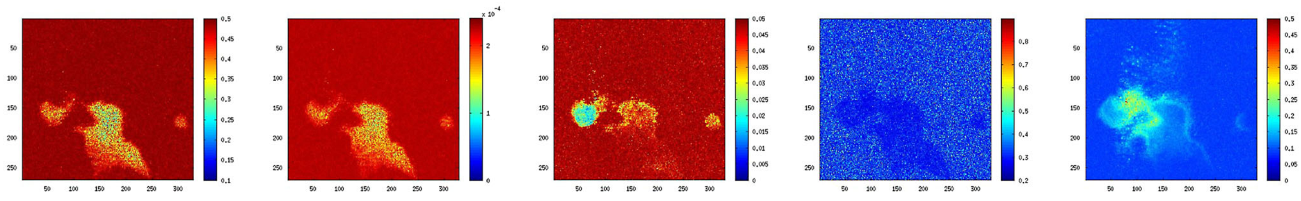
The last experiments compare PO-KMGA with its FPGA implementation. For the purpose of an unbias comparison, we synthesize directly the C code of PO-KMGA into RTL (Register Transfer Language) via HLS (High-Level Synthesis) procedure with the same parameter configurations. Nevertheless, both these two implementations are farthest parallelized with their hardware constrains respectively.

Table 7 displays the results of this experiment. Obviously, CPU is about 9.13 times more efficient than FPGA. This is because: (a) Q6600 has a much higher clock frequency than Zynq 7000, and (b) Q6600 can execute 4 threads simultaneously while only a single thread can be implemented in Zynq 7000 because of the hardware constrains. On the other hand, thanks to the HLS technic, the C-to-RTL translation does not affect the algorithm specification. That is, these two implementations have exactly the same functions, which leads to the same fitness value as shown in Table 7.

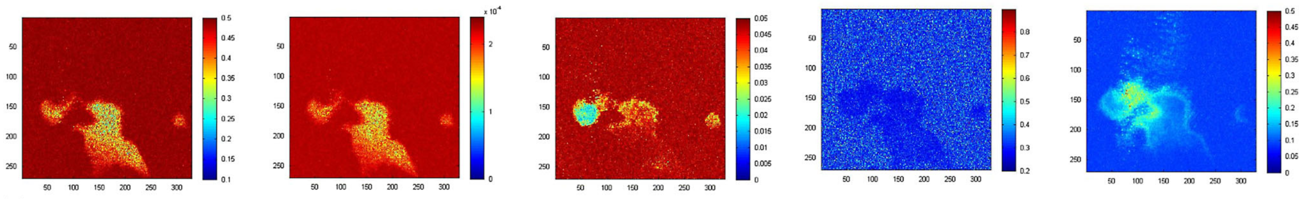
Nevertheless, we should note that, for the reasons of memory size constrain, we simulate the algorithm with only a single pixel using FPGA, while the efficiency of CPU implementation is estimated from the test results with an entirely 328 × 270 multispectral image. Thus, the former is not affected by time consuming while the latter does. It could therefore infer that the real efficiency ratio of CPU vs FPGA is higher than 9.13× in our case.



(a) Multispectral image measured by ASCLEPIOS from 450 to 780 nm with the step of 10 nm.



(b) Skin parameter maps retrieved by KMGA : melanin concentration map, epidermis thickness map, volume blood fraction map, oxygen saturatio map and dermis thickness map



(c) Skin parameter maps retrieved by PO-KMGA : melanin concentration map, epidermis thickness map, volume blood fraction map, oxyge saturation map and dermis thickness map

Fig. 8 Multispectral image measured by ASCLEPIOS and simulation results of KMGA and PO-KMGA

Table 6 Similarity evaluation between Matlab-KMGA and PO-KMGA.

Skin parameter maps	RMSE
Melanin concentration	0.013
Epidermis thickness	0.00503
Volume blood fraction	0.0019
Oxygen saturation	0.0306
Dermis thickness	0.0113

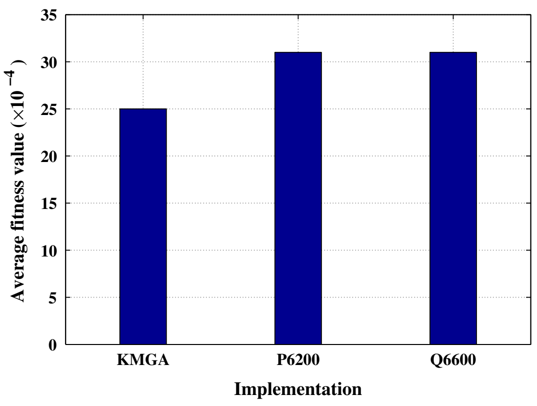


Fig. 9 Comparison between the fitness values of KMGA and PO-KMGA with a 328×270 multispectral image: a 2-threads implementation is tested for P6200 and a 4-threads implementation is tested for Q6600

5 Conclusions and perspectives

In this paper, we design and evaluate a novel framework for fast Light–Tissue Interaction-based skin lesion medical assessment using multispectral image processing and its optimization process' parameters configuration. We introduced multiple acceleration and optimization techniques such as functions re-writing for calculation complexity reduction, operation massively parallelization using POSIX threads, memory optimization, and routine pipelining with Intel C++ Compiler.

We have evaluated our Parallel and Optimized KMGA (PO-KMGA) method with a standard multispectral image using a dual-core and a quad-core CPUs. Experiments show that, when the number of threads and physical cores are four for both of them, PO-KMGA is 604.8 times faster than the conventional KMGA prototype with almost the same order of accuracy. The optimizations made in PO-KMGA accelerate the original version of its C implementation 8.46× for P6200, and 17.11× for Q6600. These achievements demonstrate that our approaches are able to

Table 7 CPU and FPGA simulation results for PO-KMGA: Q6600 for CPU and Zynq 7000 for FPGA.

	CPU	FPGA
Execution time (ms/pixel)	3.23	29.48
Fitness value	0.0031	0.0031

greatly improve the assessment efficiency of KMGA-based medical device. Moreover, according to the comparison between its CPU and FPGA implementations, it is found that CPU offers a much better efficiency performance. With PO-KMGA, the patients will not have to wait for several days for the diagnosis.

In the comparison between the two methods' accuracy performances, it is found that KMGA has a better average fitness value than PO-KMGA. But this difference is so tiny ($< 10^{-4}$) that it is able to be ignored. Considering PO-KMGA's enormous advantage in running speed, two methods' distinction in accuracy is insignificant in practical application.

In additional, our evaluation results demonstrate also that the optimization approaches used are very effective to accelerate the complex image processing prototype's speed. The achievements of our work further make possible real-time applications of the conventional KMGA method. We hope that these achievements can bring help to the other relative performance optimizations researches.

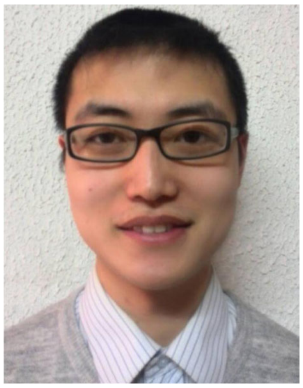
In future works, we will try to improve the performance of KMGA by developing data level optimization strategies. Since FPGA has different memory optimization strategies, a comparative study related to the optimization process or the memory hierarchy is required to find out new ways to map this problem. Meanwhile, a SW/HW co-design development framework for the complex real-time image processing algorithm will be studied to accelerate the development procedure.

Acknowledgments The authors would like to thank the China Scholarship Council, the Conseil Régional de Bourgogne-France and the Fond Européen de Développement Régional-France for their funding to our studies.

References

1. Iso/iec 9945–1. ansi/ieee std 1003.1. information technology—portable operating system interface (posix): part 1: system application interface (api) [c language] (1996)
2. Optimization of high speed pipelining in FPGA-based FIR filter design using genetic algorithm, vol. 8401 (2012). doi:10.1117/12.918934
3. W.H.O.: How common is skin cancer? Online (2014)
4. Kim, A.D., Moscoso, M.: Light transport in two-layer tissues. J. Biomed. Opt. 10(3), 034015 (2005). doi:10.1117/1.1925227
5. Akgun, D.: A practical parallel implementation for tdlms image filter on multi-core processor. J. Real Time Image Process. pp. 1–12 (2014). doi:10.1007/s11554-014-0397-y
6. Andin, J.M., Arenaz, M., Rodrguez, G., Tourio, J.: A novel compiler support for automatic parallelization on multicore systems. Parallel Comput. **39**(9), 442–460 (2013)
7. Armstrong, B., Kim, S.W., Park, I., Voss, M., Eigenmann, R.: Compiler-based tools for analyzing parallel programs. Parallel Comput. **24**, 401–420 (1998)
8. Ayguade, E., Copty, N., Duran, A., Hoeflinger, J., Lin, Y., Massaioli, F., Teruel, X., Unnikrishnan, P., Zhang, G.: The design of openmp tasks. Parallel Distrib. Syst. IEEE Trans. **20**(3), 404–418 (2009)
9. Barney, B.: Introduction to parallel computing. https://computing.llnl.gov/tutorials/parallel comp/
10. Barney, B.: Posix threads programming. https://computing.llnl.gov/tutorials/pthreads/
11. Bartovsk, J., Dokldal, P., Dokldalov, E., Georgiev, V.: Parallel implementation of sequential morphological filters. J. Real Time Image Proc. **9**(2), 315–327 (2014)
12. Biswal, P., Mondal, P., Banerjee, S.: Parallel architecture for accelerating affine transform in high-speed imaging systems. J. Real Time Image Proc. **8**(1), 69–79 (2013)
13. Gaetano, R., Chierchia, G., Pesquet-Popescu, B.: Parallel implementations of a disparity estimation algorithm based on a proximal splitting method. In: VCIP, pp. 1–6. IEEE (2012)
14. Garcia, C., Botella, G., Ayuso, F., Prieto, M., Tirado, F.: Multi-gpu based on multicriteria optimization for motion estimation system. EURASIP J. Adv. Signal Process. **2013**(1), 23 (2013). doi:10.1186/1687-6180-2013-23
15. Hosseini, F., Fijany, A., Safari, S., Fontaine, J.G.: Fast implementation of dense stereo vision algorithms on a highly parallel simd architecture. J. Real Time Image Proc. **8**(4), 421–435 (2013)
16. Jacques, S.L.: Spectroscopic determination of tissue optical properties using optical fiber spectrometer. Tech. rep. (2008). http://omlc.ogi.edu/news/apr08/skinspectra/index.html
17. Jolivot, R.: Development of an imaging system dedicated to the acquisition, analysis and multispectral characterisation of skin lesions. Ph.D. thesis, University of Burgundy (2011). http://tel.archives-ouvertes.fr/tel-00695305
18. Jolivot, R., Benezeth, Y., Marzani, F.: Skin parameter map retrieval from a dedicated multispectral imaging system applied to dermatology/cosmetology. Int. J. Biomed. Imaging **2013**, 15 (2013)
19. Kubelka, P., Munk, F.: Ein beitrag zur optik der farbanstriche. Zeitschrift fr tech. Physik **9**, 593601 (1931)
20. Kyrkou, C., Theocharides, T.: A parallel hardware architecture for real-time object detection with support vector machines. Comput. IEEE Trans. **61**(6), 831–842 (2012)
21. Li, X., Cai, J.: Robust transmission of jpeg2000 encoded images over packet loss channels. In: Multimedia and Expo, 2007 IEEE International Conference on, pp. 947–950 (2007)
22. Lucas, R., McMichael, T., Smith, W., Armstrong, B.: Solar ultraviolet radiation: global burden of disease from solar ultraviolet radiation. Tech. rep., Environmental Burden of Disease Series—WHO, No. 13 (2006)
23. Prigent, S., Descombes, X., Zugaj, D., Martel, P., Zerubia, J.: Multi-spectral image analysis for skin pigmentation classification pp. 3641–3644 (2010). doi:10.1109/ICIP.2010.5652072
24. Prigent, S., Descombes, X., Zugaj, D., Zerubia, J.: Spectral analysis and unsupervised svm classification for skin hyper-

pigmentation classification pp. 1–4 (2010). doi:10.1109/WHISPERS.2010.5594917
25. Schmitt, J.M., Zhou, G.X., Walker, E.C., Wall, R.T.: Multilayer model of photon diffusion in skin. J. Opt. Soc. Am. A **7**(11), 2141–2153 (1990)
26. Choi, S.H.: Fast and robust extraction of optical and morphological properties of human skin using a hybrid stochastic-deterministic algorithm: Monte-carlo simulation study. Lasers Med Sci. pp. 733–741 (2010)
27. Shimada, M., Yamada, Y., Itoh, M., Yatagai, T.: Melanin and blood concentration in human skin studied by multiple regression analysis: experiments. Phys Med Biol **46**(9), 2385 (2001)
28. Stratton, J.A., Stone, S.S., Wen-Mei, W.H.: Mcuda: An efficient implementation of cuda kernels for multi-core cpus. In: Languages and Compilers for Parallel Computing, pp. 16–30. Springer, Berlin (2008)
29. Svaasand, L., Norvang, L., Fiskerstrand, E., Stopps, E., Berns, M., Nelson, J.: Tissue parameters determining the visual appearance of normal skin and port-wine stains. Lasers Med. Sci. **10**(1), 55–65 (1995)
30. Syswerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 2–9. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989). http://dl.acm.org/citation.cfm?id=645512.657265
31. Takatani, S., Graham, M.D.: Theoretical analysis of diffuse reflectance from a two-layer tissue model. Biomed. Eng. IEEE Trans. BME, 26(12), 656–664 (1979)
32. Thomos, N., Boulgouris, N., Strintzis, M.: Optimized transmission of jpeg2000 streams over wireless channels. Image Process IEEE Trans **15**(1), 54–67 (2006)
33. van Tol, M., Jesshope, C., Lankamp, M., Polstra, S.: An implementation of the SANE virtual processor using POSIX threads. J. Syst. Architect. **55**(3), 162–169 (2009)
34. Tsumura, N., Ojima, N., Sato, K., Shiraishi, M., Shimizu, H., Nabeshima, H., Akazaki, S., Hori, K., Miyake, Y.: Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin. ACM Trans. Graph. **22**(3), 770–779 (2003). doi:10.1145/882262.882344
35. Viator, J.A., Jung, B., Svaasand, L.O., Aguilar, G., Nelson, J.S., Zhang, R., Verkruysse, W., Choi, B.: Determination of human skin optical properties from spectrophotometric measurements based on optimization by genetic algorithms. J. Biomed. Opt. 10(2), 024,030–024,030–11 (2005)
36. Xu, T., Wang, Y., Zhang, Z.: Pixel-wise skin colour detection based on flexible neural tree. Image Process. IET **7**(8), 751–761 (2013). doi:10.1049/iet-ipr.2012.0657
37. Zonios, G., Bykowski, J., Kollias, N.: Skin melanin, hemoglobin, and light scattering properties can be quantitatively assessed in vivo using diffuse reflectance spectroscopy. J. Invest. Dermatol. 117(6), 1452–1457 (2001). doi:10.1046/j.0022-202x.2001.01577.x



Chao Li received the M.S. in Electronic Engineering from the University of Burgundy in 2012 and the M.S. in Computer Science from the University of Paris Sud in 2013, respectively. He is currently a Ph.D. candidate of LE2I CNRS-UMR, Laboratory of Electronic, Computing and Imaging Sciences at the University of Burgundy. His research interests are in the areas of SW/HW Co-design, High Performance Computer and Image Processing.

Vincent Brost received the Master Degree and Ph.D. degrees from University of Burgundy, in 2000 and 2006, respectively. He is currently an associate professor at the University of Burgundy and member of LE2I (Laboratory of Electronic, Computing, and Imaging), France. His research interests are in the areas of rapid prototyping system, VLIW architecture, and Electronic Devices.

Yannick Benezeth is associate professor at the University of Burgundy (France). He obtained his Ph.D. in computer science from the University of Orléans in 2009. He also received an engineer degree from the ENSI de Bourges and the MS degree from the University of Versailles-Saint-Quentin-en-Yvelines in 2006. His research interests include multispectral image analysis applied to health and video content analysis (people detection, object tracking and scene understanding).

Franck Marzani received his M.Sc. in computer science from the University of Rennes, France in 1989. He obtained his Ph.D. in computer vision and image processing from the University of Burgundy, Dijon, France in 1998. His PhD dissertation dealt with 3D acquisition and processing of human motion. Since then he has worked in the LE2I laboratory (Laboratoire d'Electronique, d'Informatique et Image), which is a CNRS affiliated institute of research (UMR CNRS 6306) at the University of Burgundy. He received his "Habilitation Diriger les Recherches" in 2007 and he is a full professor since 2009. His research interests include both acquisition and analysis of multispectral images and 3D imaging. He has been developing an activity on feature extraction from multispectral images based on light–matter interaction models and texture mapping of such non-conventional image modalities on 3D meshes. These methodologies have been applied to cultural heritage and health with a focus on dermatology.

Fan Yang received the B.S. degree in electrical engineering from the University of Lanzhou, China, in 1982 and the M.S. (D.E.A.) (computer science) and Ph.D. degrees (image processing) from the University of Burgundy, France, in 1994 and 1998, respectively. She is currently a full professor and member of LE2I CNRS-UMR, Laboratory of Electronic, Computing, and Imaging Sciences at the University of Burgundy, France. Her research interests are in the areas of patterns recognition, neural network, motion estimation based on spatio-temporal Gabor filters, parallelism and real-time implementation, and, more specifically, automatic face image processing algorithms and architectures. Pr. Yang is member of the french research group ISIS (Information, Signal, Images and Vision), she livens up the theme C: Algorithm Architecture Adequation.