electronics

MDPI

Article

Parallel Classification Pipelines for Skin Cancer Detection Exploiting Hyperspectral Imaging on Hybrid Systems

Emanuele Torti [1,*], Raquel Leon [2], Marco La Salvia [1], Giordana Florimbi [1], Beatriz Martinez-Vega [2], Himar Fabelo [2], Samuel Ortega [2], Gustavo M. Callicó [2,*] and Francesco Leporati [1]

[1] Department of Electrical, Computer and Biomedical Engineering, University of Pavia, 27100 Pavia, Italy; marco.lasalvia01@universitadipavia.it (M.L.S.); giordana.florimbi01@universitadipavia.it (G.F.); leporati@unipv.it (F.L.)

[2] Institute for Applied Microelectronics (IUMA), University of Las Palmas de Gran Canaria (ULPGC), 35017 Las Palmas de Gran Canaria, Spain; slmartin@iuma.ulpgc.es (R.L.); bmartinez@iuma.ulpgc.es (B.M.-V.); hfabelo@iuma.ulpgc.es (H.F.); sortega@iuma.ulpgc.es (S.O.)

* Correspondence: emanuele.torti@unipv.it (E.T.); gustavo@iuma.ulpgc.es (G.M.C.); Tel.: +39-0382-985-678 (E.T.)

Received: 2 September 2020; Accepted: 10 September 2020; Published: 13 September 2020

Abstract: The early detection of skin cancer is of crucial importance to plan an effective therapy to treat the lesion. In routine medical practice, the diagnosis is based on the visual inspection of the lesion and it relies on the dermatologists' expertise. After a first examination, the dermatologist may require a biopsy to confirm if the lesion is malignant or not. This methodology suffers from false positives and negatives issues, leading to unnecessary surgical procedures. Hyperspectral imaging is gaining relevance in this medical field since it is a non-invasive and non-ionizing technique, capable of providing higher accuracy than traditional imaging methods. Therefore, the development of an automatic classification system based on hyperspectral images could improve the medical practice to distinguish pigmented skin lesions from malignant, benign, and atypical lesions. Additionally, the system can assist general practitioners in first aid care to prevent noncritical lesions from reaching dermatologists, thereby alleviating the workload of medical specialists. In this paper is presented a parallel pipeline for skin cancer detection that exploits hyperspectral imaging. The computational times of the serial processing have been reduced by adopting multicore and many-core technologies, such as OpenMP and CUDA paradigms. Different parallel approaches have been combined, leading to the development of fifteen classification pipeline versions. Experimental results using in-vivo hyperspectral images show that a hybrid parallel approach is capable of classifying an image of 50×50 pixels with 125 bands in less than 1 s.

Keywords: real-time systems; graphic processing units; multicore CPU; cancer detection; hyperspectral imaging

1. Introduction

Hyperspectral imaging (HSI) is a form of imaging spectroscopy that produces three-dimensional images whose pixels are characterized by the spectral information of the acquired scene. This cube contains the reflectance values of the acquired image, i.e., the fraction of incident electromagnetic radiation that is reflected upon a surface. Each material presents a specific variation of reflectance values with respect to wavelengths. This variation is called spectral signature and it is unique for each type of material, allowing precise discrimination [1]. Hyperspectral image classification systems

aim at recognizing the material contained in every pixel in the scene. The classification is performed adopting several supervised and unsupervised algorithms, whose elaboration could be computationally intensive. This is a critical issue since most of the applications require a response in a short processing time or even in real-time. Therefore, the scientific community have proposed several works exploiting parallel technologies for HSI classification [2].

HS images were created mainly for military purposes in the remote-sensing field [3]. Since then, they have become very useful in different areas, such as in geology [4], archaeology [5], or food inspection [6], among others [7]. HSI were used only by few companies and research institutes since the HS cameras and the computational systems were very expensive. The recent technological advances have facilitated the use of the HS images in other fields, such as medicine [8,9] and, in particular, cancer detection [8–11]. The reason why this technique is exploited, especially in cancer detection, is that HSI is a non-invasive, non-contact and non-ionizing method, capable of obtaining both spatial and spectral information from the scene. Moreover, the biochemical and morphological changes associated with lesions modify the optical characteristics of a tissue, such as absorption, scattering or fluorescence. These variations provide valuable diagnostic information useful in the diagnosis and detection phases [12–16]. In such a context, it is of crucial importance to achieve a fast or near real-time response. In order to reduce the processing time, modern systems adopt a simultaneous elaboration of the pixels of the image, when allowed by the algorithm. Therefore, the use of parallel technologies is suitable for a pixel-wise classification, where each processing core is in charge of elaborating a single pixel or a group of pixels. The state of the art relies on hybrid systems including multicore and many-core devices. The choice of hybrid systems is justified by the features of the processing chains that typically include algorithms with different levels of complexity [17–20]. The key idea is that each algorithm is managed by the device that best meets the processing constraints. The output of these classification systems is a thematic map, where a color is assigned to each pixel representing a specific tissue type or condition.

Skin cancer is one of the most common forms of cancer in the world and can be categorized as non-melanoma skin cancer (NMSC) and melanoma. NMSC was the 5th most common form of cancer worldwide in 2018, while melanoma was the 21st [21]. Pigmented skin lesions (PSLs) are caused by an extreme progression of melanocytes and can be classified as benign or malignant [22]. Atypical moles, also known as dysplastic nevi, are benign PSLs and are associated with an increased risk of evolving to melanoma [23]. Previous works in skin cancer detection have applied machine learning approaches to identify PSL using HS in-vivo skin cancer data [24]. In [25], the authors used a genetic algorithm to optimize the supervised machine learning algorithms for the identification of four types of PSLs: nevus, BCC (basal cell carcinoma), melanoma, and others (other types of PSLs not included in the remaining classes). A HS dermatologic classification system based on a combination of unsupervised and supervised algorithms was proposed to discriminate between malignant and benign PSLs [26]. Other commercial systems, such as SIAscope/SIAscopy [27] or MelaFind [28], use multispectral images to detect only melanoma lesion, limiting the identification of other malignant PSL that should be analyzed.

This work presents a parallel implementation of a HS dermatologic classification framework based on K-means and SVM (support vector machines) algorithms and snapshot HS cameras with the goal of achieving an automatic in-situ PSL identification. In our previous works, we described the HS in-situ acquisition system and performed an initial study to validate the hypothesis that HSI can be used to differentiate between PSLs through pixel-wise supervised classification [25]. However, in [26], we developed a complete classification framework for processing the HS data to differentiate between malignant and benign PSLs, but without considering the implementation and parallelization of the proposed algorithm. This manuscript presents a variation of the classification framework aiming to differentiate between malignant, benign, and atypical PSLs as well as an exhaustive study of the implementation and parallelization of such algorithm to obtain real-time performance. This real-time diagnosis could assist dermatologists in the discrimination of PSLs during clinical routing practice,

providing more diagnostic information regarding non-melanoma skin cancer than other commercial systems that only discriminate between melanoma and non-melanoma [27,28].

The paper is organized as follows. In Section 2, the hyperspectral skin cancer database and the classification framework are described. Section 3 presents the parallel implementations based on multicore CPU and many-core graphical processing unit (GPU) technologies. Then, the experimental results are reported and discussed in Section 4. Finally, Section 5 discusses some conclusions regarding the research presented.

2. Materials and Methods

2.1. Hyperspectral Skin Cancer Database

The HS dermatologic acquisition system developed in [25] was used to create the HS in-vivo skin cancer dataset employed in this work. The system is based on a snapshot HS camera (Cubert UHD 185, Cubert GmbH, Ulm, Germany) coupled to a Cinegon 1.9/10 (Schneider Optics Inc., Hauppauge, NY, USA) lens with a F-number of 1.9 and a focal length of 10.4 nm. The illumination system (Dolan-Jenner, Boxborough, MA, USA) employs a 150 W QTH (Quartz-Tungsten Halogen) lamp coupled to a fiber optic ring light guide to obtain cold light emission. This ring light is attached to the HS camera through a customized 3D printed dermoscopic contact structure. The resulting HS image contains 125 spectral bands covering the visual and near-infrared (VNIR) spectral range from 450 to 950 nm, having a spatial resolution of 50×50 pixels (pixel size of 240×240 μm).

The HS skin cancer database is composed of 76 HS images obtained from 61 subjects with different PSLs located in different parts of the body. The HS images were labeled using a labeling tool [29] based on the SAM (spectral angle mapper) algorithm to create a labeled dataset. This tool allows to manually select a reference pixel and label the most similar pixels according to the spectral angle metric. The data were labeled in three different classes: benign, malignant, and atypical. The HS labeled dataset was partitioned into training, validation, and test set. Validation and test set were composed by 9 HS images.

2.2. Hyperspectral Dermatologic Classification Framework

The HS dermatologic classification framework is composed of three main steps: HS data pre-processing, automatic PSL segmentation, and supervised classification. Figure 1 shows a block diagram of this framework. The first step consists in performing the pre-processing chain to homogenize the incoming raw HS image captured by the HS dermatologic acquisition system. After performing the pre-processing, the resulting image is automatically segmented, where the normal skin and PSL pixels are discriminated. This discrimination is performed using a spectral signature reference library, composed of three spectral signatures of benign, malignant and atypical PSL (in blue, red and black colors respectively in Figure 1) and three skin spectral signatures (in green color in Figure 1). To obtain the spectral reference from the PSL class, the average of the labeled spectral signatures was computed using the training set and from the skin class, the normal skin data were divided into three groups using the K-means clustering algorithm, where the number of clusters employed was selected after evaluating the results using the Silhouette [30], Calinski Harabasz [31], and Davies Bouldin [32] methods. These normal skin spectral signatures correspond with the centroids obtained with the K-means algorithm. Split the normal class into three groups allows having a variety of different skin types. These skin differences are particularly highlighted in the NIR region. Finally, the pixels previously identified as lesion are classified by a supervised classifier, providing the class results, i.e., benign, malignant, and atypical.

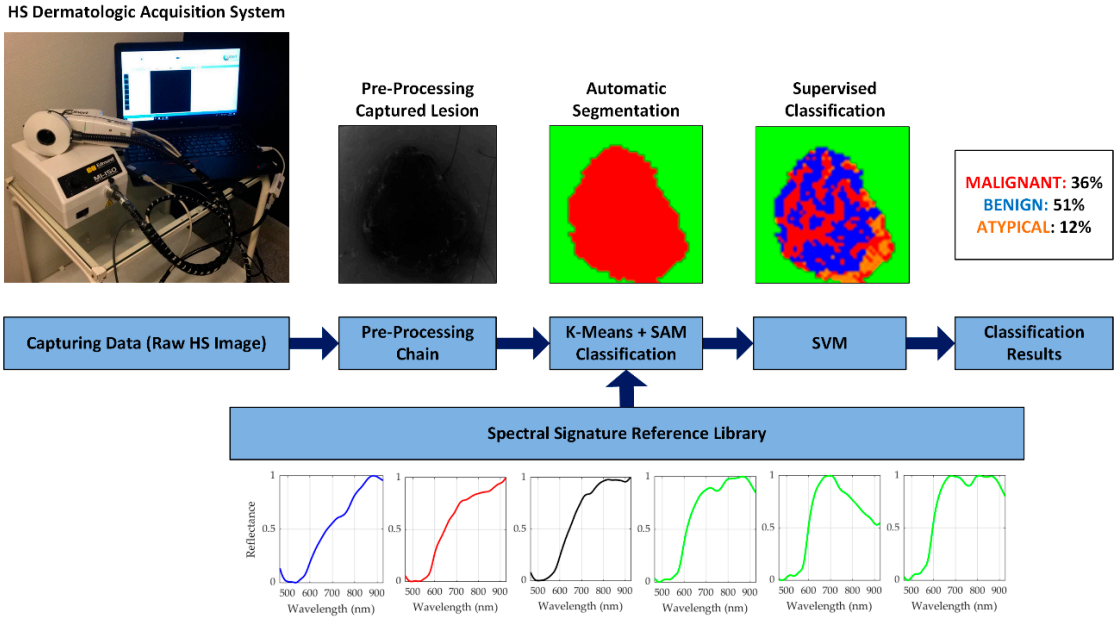

Figure 1. Block diagram of the HS dermatologic classification framework (pre-processing, automatic segmentation, and supervised classification) and HS dermatologic acquisition system. HS dermatologic acquisition system is composed by: HS snapshot camera; QTH (Quartz-Tungsten Halogen) source light; Fiber optic ring light guide; Skin contact part attached to the ring light; Laptop with the acquisition software installed. Spectral signature reference library is composed of six spectral signatures: benign, malignant, and atypical pigmented skin lesion spectral signatures in blue, red, and black colors respectively, and three different skin spectral signatures in green color.

2.2.1. Pre-Processing Chain

The HS data captured by the HS dermatologic acquisition system were pre-processed to homogenize the spectral signatures among the different patients. The pre-processing chain consists of four stages: calibration, extreme bands removal, noise filtering and normalization. The pseudo-code of the pre-processing chain is shown in Algorithm 1, where Y indicates a HS image with n pixels and b bands.

In Algorithm 1, lines from 3 to 7 perform the calibration stage. The raw HS image (Y) is calibrated employing a white reference image (W_{ref}), captured using a white reference tile able to reflect the 99% of the incident light, and a dark reference image (D_{ref}) obtained by having the light turned off and the camera shutter closed. Both images, W_{ref} and D_{ref} were obtained before the data acquisition using the same illumination conditions. The resulting calibrated image ($Y_{calibrated}$) is obtained following the equation shown in line 5.

After the calibration stage, the first 4 bands and the last 5 bands are removed due to the HS sensor low response. This extreme bands removal is performed in line 9 and the final spectral signature consists of 116 bands. Moreover, the HS data is filtered using a smooth filter to reduce the spectral noise in the remaining spectral bands. For each i-th iteration of the for loop in line 11, the smooth filter is applied to a pixel of the HS image with its respective 116 spectral bands. This filter is based on a moving average filter and requires that the first and last elements of the vector must not change. Line 12 contains the loop declaration where N is the value of the neighbors previously chosen. For this application, N is set to 5.

Finally, a normalization process between 0 and 1 is applied to each pixel with the goal of homogenizing its amplitude. Lines from 21 to 26 perform the normalization process where the resulting normalized image is obtained following equation shown in line 24.

Algorithm 1. Pre-processing chain.

Input: $Y \rightarrow$ Hyperspectral image with n pixels and b bands
$D_{ref} \rightarrow$Dark reference
$W_{ref} \rightarrow$White reference
$N \rightarrow$number of neighbours

1. Hyperspectral image Y acquisition
2. *Stage 1.1: Image calibration*
3. **for** i = 0 to *n-1* **do**:
4. **for** j = 0 to *b-1* **do**:
5. $Y_{calibrated}(i,j) = \frac{Y(i,j)-D_{ref}}{W_{ref}-D_{ref}}$;
6. **end**
7. **end**
8. *Stage 1.2: Extreme bands removal*
9. Remove the first 4 and last 5 noisy bands
10. *Stage 1.3: Smooth filtering*
11. **for** i = 0 to *n-1* **do**:
12. **for** j = 1 to *b-N-1* **do**:
13. $sum = 0$;
14. **for** x = 0 to *b-1* **do**:
15. $sum+ = Y_{calibrated}(i, j+x)$;
16. **end**
17. $Y_{calibrated}(i,j) = sum/b$;
18. **end**
19. **end**
20. *Stage 1.4: Normalization*
21. **for** i = 0 to *n-1* **do**:
22. Find the *max* and *min* values over the bands
23. **for** j = 0 to *b-1* **do**:
24. $Y_{calibrated}(i,j) = \frac{Y_{calibrated}(i,j)-min}{max-min}$;
25. **end**
26. **end**

Output: $Y_{calibrated}(i,j)$

2.2.2. Automatic PSL Segmentation

The pre-processed HS image is divided into normal skin and PSL using an automatic segmentation method based on K-means and SAM algorithms. The first step consists of performing the segmentation using the K-means algorithm to divide the HS images into k different clusters. The k value was previously selected after employing a clustering evaluation method [26]. The optimal k value for this application is three. After performing the segmentation into three clusters, a two-class segmentation map is generated to identify normal skin and PSL. This map is generated using the SAM algorithm, which compares the centroid from each cluster with a spectral signature reference library. The most similar spectral signature to each centroid is assigned to a certain class (PSL or normal skin). The library is obtained by computing the average of the labeled spectral signatures using the training set and the normal skin data were divided into three groups, allowing to have a variety of different skin types. Finally, the library contains six different spectral signatures: three from PSL (malignant, benign, and atypical lesions in Figure 1, represented in red, blue, and black colors, respectively) and three from normal skin (in green color in Figure 1).

The pseudo-code of the automatic segmentation is shown in Algorithm 2, where the number of clusters is determined by *k*, the threshold error by *threshold* and the maximum number of iterations by *MAX_ITER*. Line 2 performs the initialization of the *actual_centroids* variable to select the centroids used by the K-means algorithm. This variable is initialized with *k* different HS pixels randomly chosen from the input image *Y*. The *error* variable in line 3 is calculated as the average of the absolute values of the difference between centroids. This parameter is used as a constraint for the convergence of the algorithm. The main loop of the algorithm from line 6 to 13 computes the distances between a certain pixel and the centroids with an iterative procedure. The distance is computed using the Euclidean metric where each pixel of the HS image is assigned to a certain cluster when the minimum distance is reached. The use of *actual_centroids* and *previous_centroids* variables allows to analyze the variation from the previous iteration. This loop finishes when the error becomes lower than the established threshold or after a maximum number of iterations.

Algorithm 2. Automatic segmentation.

Input: *Y, k, threshold*, *MAX_ITER*, *HUGE_VAL*

1. *Stage 2.1: K-means initialization*
2. Randomly choose *k* pixels as *actual_centroids*
3. $error = HUGE\_VAL$;
4. $iter = 0$;
5. *Stage 2.2: K-means clustering*
6. **while** $error < threshold$ && $iter < MAX\_ITER$ **do**:
7. Compute the distance between pixels and centroids
8. Clusters update
9. $previous\_centroids = actual\_centroids$;
10. Update *actual_centroids*
11. Compute error between *actual_centroids* and *previous_centroids*
12. $iter++$;
13. **end**
14. **for** $i = 0$ to *k-1* **do**:
15. **for** $j = 0$ to n_{ref}*-1* **do**:
16. *dist(j)* = compute SAM between *actual_centroids(i)* and *ref(j)*
17. **end**
18. *h* = find the index of the minimum value of *dist*
19. Assign to the i-th cluster the same class as *ref(h)*
20. **end**

Output: PSL pixels

When the *while* loop finishes, the segmented output image is made of different clusters. To identify which cluster belongs to each class (normal skin or PSL), the spectral signature reference library is compared with each cluster using the SAM algorithm. Lines from 14 to 20 correspond to this procedure (similarity evaluation) where the six reference spectral signatures are compared with each cluster and assigned when the SAM result reached the minimum value.

2.2.3. Supervised Classification

The pixels identified as PSL by the automatic segmentation were classified using a supervised algorithm. the support vector machine (SVM) algorithm was selected to perform the classification because it has been commonly used for HS data classification in medical applications [8]. The goal of the SVM algorithm is to find the best hyperplane to separate different data and compute the probability to belong to each class of study [33]. Different kernel functions can be used to achieve the

best result and each kernel function has different hyperparameters that can be tuned to obtain the optimal configuration.

In this study, the Sigmoid kernel was selected after comparing the performance results with the Linear and Radial Basis Function (RBF) kernels with the optimal hyperparameters (as it will be presented in Section 4.1). Algorithm 3 shows the pseudo-code of the supervised classification where *pix_no_skin* contains the lesion pixels obtained in the output of the previous algorithm.

Algorithm 3. Supervised classification.

Input: *pix_no_skin*, $n_{\text{pix\_no\_skin}}$, *class*, n_{sv}, *sv*, *epsilon*

1. *Stage 3.1: SVM data preparation*
2. **for** i = 0 to $n_{pix\_no\_skin}$-*1* **do**:
3. *Stage 3.2: SVM distance computation*
4. **for** j = 0 to n_{sv}-*1* **do**:
5. $prod = sv(j) * pix\_no\_skin(i)$;
6. $dist(j) = \tan h(slope * prod + intercept)$
7. **end**
8. *Stage 3.3: SVM binary classification*
9. **for** j = 0 to *class-2* **do**:
10. **for** $z = j$ to *class-1* **do**:
11. Solve binary classification problem between class j and class z
12. **end**
13. **end**
14. *Stage 3.4: SVM multiclass probability*
15. $Pc1 = \ldots = Pcclass = \frac{1}{class}$;
16. Computing the matrix *Qp* using the binary probabilities
17. **for** z = *0* to *class-1* **do**:
18. **for** *iter* = 0 to 99 **do**:
19. **if** $Pc_z - Pc_z\_prev <$ epsilon **do**:
20. **break;**
21. **end**
22. Update multiclass probability of the i-th pixel to belong to class z
23. **end**
24. **end**
25. Compute class with maximum probability for the i-th pixel
26. **end**
27. Update similarity evaluation labels with SVM results

Output: Probabilities class.

The pseudo-code of the SVM classification algorithm is characterized by four main phases: data preparation, distance computation, binary classification, and multiclass probability. The probability is computed with an iterative procedure in the *for* loop from line 2 to 26, where the probability of the i-th pixel to belong to a certain class is obtained. To evaluate this probability, it is necessary to calculate the distance using the Sigmoid kernel. In line 5, the pixel is multiplied by a support vector and in line 6 the distance is computed using the parameters of the sigmoid kernel: *slope* and *intercept*. The next stage, from line 9 to 13, performs the binary classification on the basis of probability of a certain pixel to belong to the two classes under study. Finally, the multiclass probability is performed from line 15 to 24, using the probabilities obtained in the previous stage. Line 15 initializes the class probabilities and the matrix *Qp* is computed using the binary probabilities. In an iterative procedure, the pixel probability of belonging to a class is refined. This process ends when the value of the previous

iteration is under a certain threshold, or if the number of iterations reaches 100. When one of these two conditions takes place, the multiclass probabilities of the pixel are computed.

The HS dermatologic classification framework features high computational complexity on serial systems, thus preventing real-time processing. Therefore, the exploration of parallel architectures is mandatory to provide an efficient instrument for the clinical practice.

3. Parallel Classification Pipelines

In order to reduce the processing time of the serial classification pipeline, different parallel strategies targeting multicore and many-core technologies have been explored. The first step has been the classification framework (pre-processing, K-means, and SVM) development in C language. This serial code represents a basis for the parallel versions that are successively developed integrating the OpenMP and the CUDA frameworks for the multicore and many-core philosophy, respectively. Several parallel classification pipelines have been developed, where, for each algorithm, an OpenMP or CUDA version is included to determine the best solution to be considered in the final classification system.

3.1. Parallel Pre-Processing Versions

As already discussed, the pre-processing chain consists of four steps: calibration, extreme bands removal, filtering, and normalization. Two parallel versions of this algorithm (multicore and many-core) have been developed and, in both cases, a pixel-wise parallelization is carried out. A serial code profiling has shown that the most time-consuming phases are filtering and normalization. Therefore, only these two steps are parallelized. Each thread performs the filtering and normalization of a single HS pixel. These two steps are included in a *for* loop that iterates over the number of pixels. The iterations are elaborated simultaneously since the *pragma omp parallel for* directive has been introduced before the loop. Moreover, the loop variable is declared as private, while the HS image is shared among all the threads.

The same parallelization strategy has been adopted in the CUDA version development. After the image calibration and the bands removal, data are allocated and transferred to the device global memory. Among the transferred data, there are the reduced image and an array storing groups of five contiguous bands for each pixel. This array is used in the smooth filtering step to avoid data overwrite during the moving average computation. The filtering is performed by a kernel through a grid containing a number of threads equal to the number of pixels. The grid includes blocks of 32 threads. This number has been chosen according to the warp definition given by NVIDIA. If the number of pixels is not an integer multiple of 32, the last block will contain some threads that are not related to a pixel. In this case, these threads do not perform any computation. It is worth noticing that these inactive threads do not slow down the computation because their number is negligible compared to the total number of pixels. Another kernel, with the same grid and block parameters, computes the maximum and minimum values of each pixel across the bands. These values are then used in the normalization step, performed by a further kernel. The normalized image overwrites the original one, which is initially transferred to the device global memory. The result of the pre-processing is transferred back to the host memory only if the K-means is performed using a serial or an OpenMP processing. Otherwise, the normalized image is left in the device memory to be used by the CUDA version of the K-means. The flowchart of the CUDA pre-processing is shown in Figure 2.

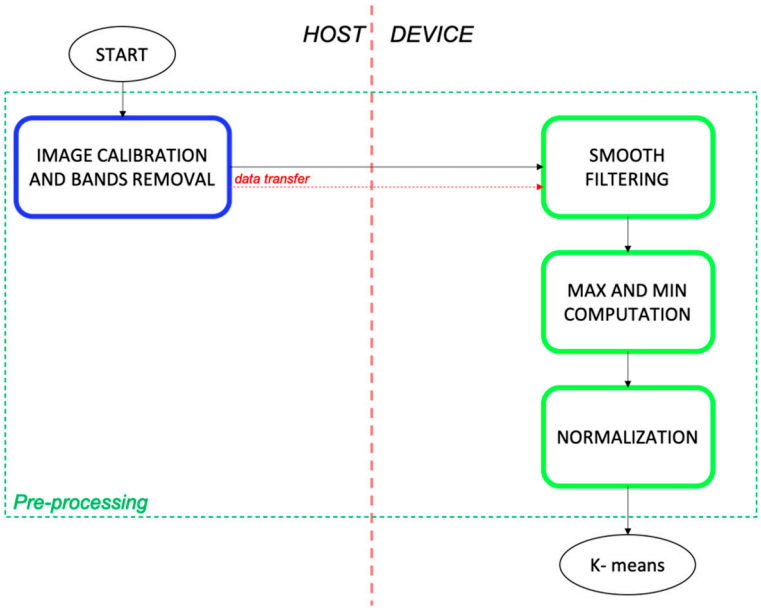

Figure 2. Flowchart of the pre-processing CUDA version.

3.2. Parallel K-Means Versions

The most time-consuming part of this algorithm is the distance computation between each pixel and each centroid. It should be emphasized that this computation must be performed a number of times equal to the number of pixels by the number of clusters. Since the other operations have a negligible computational cost when performed on a serial processor, only the distance computation has been pixel-wise parallelized using OpenMP. Also, in this case, the *pragma omp parallel for* has been introduced before the *for* loop iterating over the pixels. Again, the loop variables are declared private and the HS image is shared. Moreover, also the centroids are shared among the pixels.

In the parallel CUDA version, a different strategy has been adopted. In this case, all the steps have been performed on the device to minimize data transfers between host and device memories. The flowchart of this parallel version is shown in Figure 3. The flow starts on the host, where the indexes for the centroid's initialization are generated. These indexes are used to identify which pixels are selected to define the initial centroids. Then, these values are transferred to the device. It must be noticed that if the pre-processing has been performed on the device, the image is already stored in the GPU global memory. Otherwise, it is also transferred from the host.

The first task performed by the GPU is the centroid initialization, which consists of copying the values of the selected pixels into the centroids. This step is elaborated by a kernel, whose threads number is equal to the cluster number. The error computation is split between device and host. At first, a kernel computes the difference between the actual and the previous centroids (initialized to zero). Then, the *cublasSasum* function is used to sum the absolute values of these differences. This function directly transfers the output on the host, where the division by the number of clusters is performed using a serial thread. At this point, the iterative K-means process starts on the host. The next steps are repeated until the error converges to a fixed threshold or the maximum number of iterations is reached. In this iterative part, the first step concerns the distance computation performed on the device by a kernel, whose number of threads is equal to the number of pixels. In particular, each thread simultaneously computes the distance between the pixels and the centroids. Then, the clusters and the centroids are updated with two different kernels. The former provides a pixel-wise parallelization since each thread finds the nearest centroids for the assigned pixel. The latter includes a number of threads equal to the number of clusters to perform the update. At this point, the error is evaluated as already explained. Once the condition of the *while* loop is false, the flow continues with the similarity evaluation step, which assigns a biological meaning to each cluster (PSL or normal skin). In this phase,

the difference among the centroids and the six reference spectral signatures are computed. All the elements belonging to the same cluster are labelled with the class of the reference spectral signature with the minimum distance. It is worth noticing that this computation involves a restricted number of data, allowing its efficient elaboration on the host.

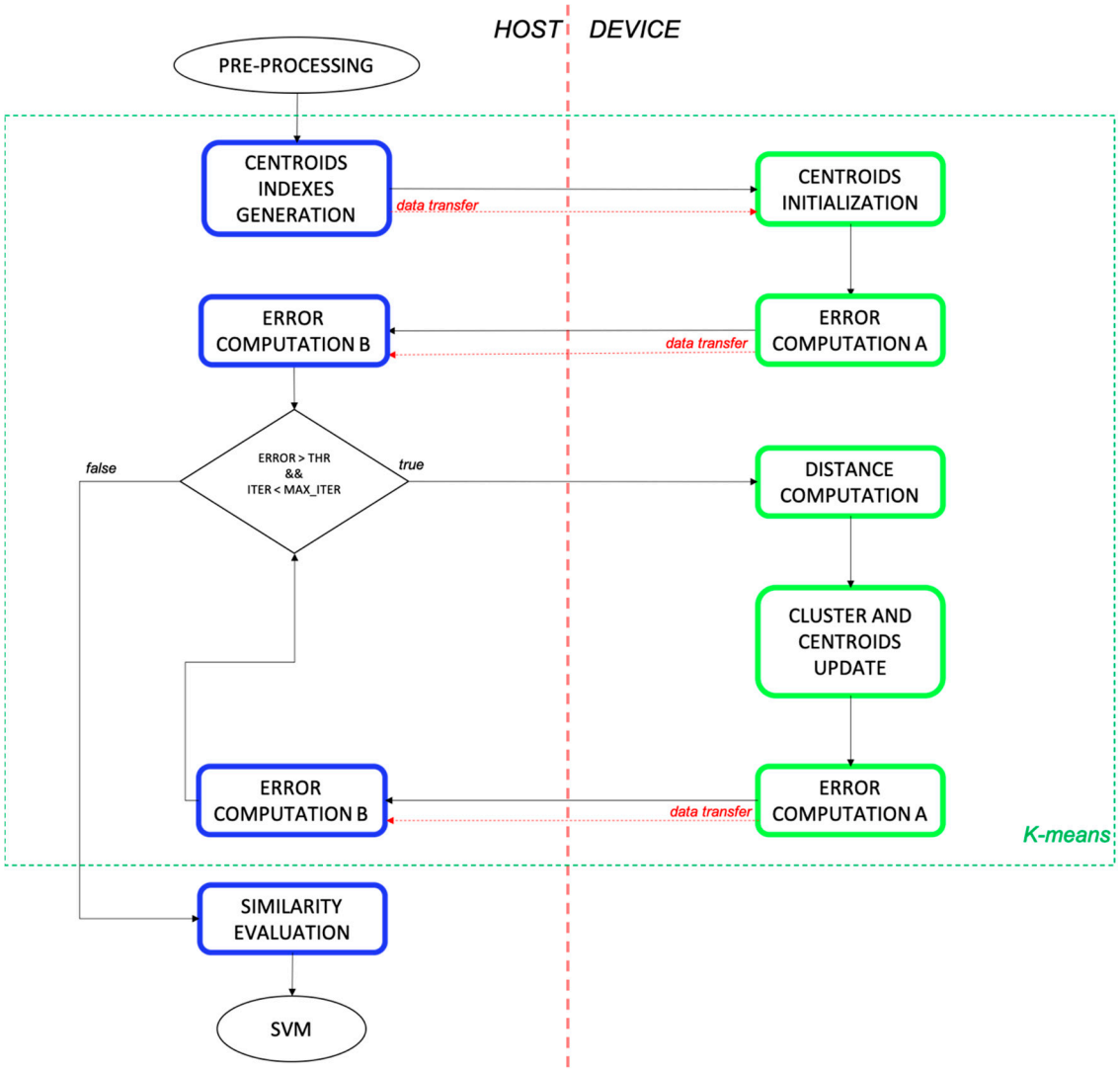

Figure 3. Flowchart of the K-means CUDA version.

3.3. Parallel Support Vector Machine Versions

As introduced before, the SVM algorithm is composed of three main steps: distance and non-linear function evaluation, binary classification, and multiclass probabilities computation. The first phase is the most time-consuming part, and for this reason, an OpenMP parallelization of this routine has been developed. It must be considered that only a subset of the original HS image is sent to the SVM algorithm as input. The SVM training has generated a model with 9242 support vectors, which is higher than the number of pixels of each image. For this reason, the *for* loop that iterates over the support vectors to improve the performance has been parallelized. In particular, each thread performs the dot product between the assigned support vector and the pixel. Then, it applies the hyperbolic tangent to the product result, after considering the slope and intercept values. In this case, the shared variables are the pixels to be classified and the support vectors. The private variables are the *for* loops indexes.

When considering the CUDA parallelization, three different versions have been developed to find the most efficient one. The first and second CUDA versions flowchart is shown in Figure 4.

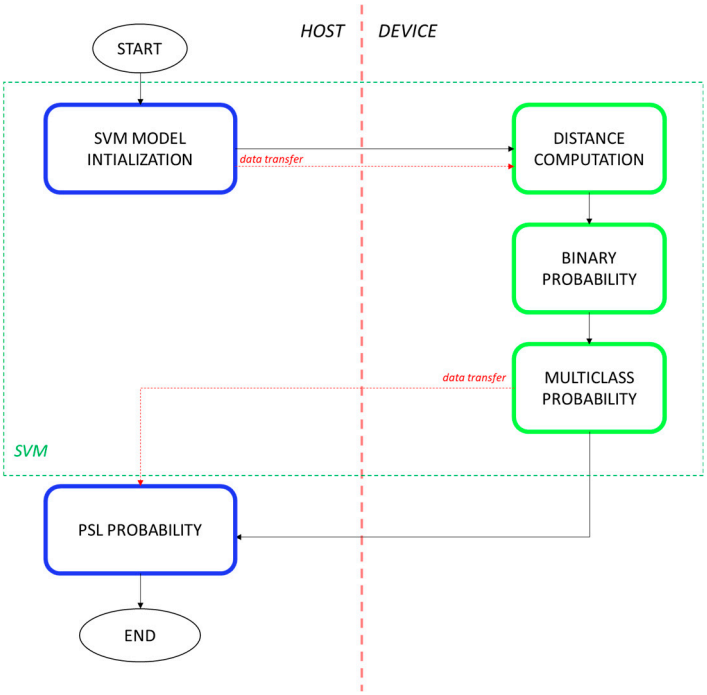

Figure 4. Flowchart of the first and second CUDA SVM versions.

The flow starts on the host, where the SVM model parameters and the pixels to be classified are transferred to the device memory. Here, a kernel called *distance computation* is performed, where the dot product between support vectors and pixels is computed. Moreover, the hyperbolic tangent is evaluated. In this kernel, the number of threads is equal to the number of the support vectors for the same reason explained in the OpenMP version. Again, each block contains 32 threads. The binary probability is computed in another kernel, whose grid dimension represents the difference between the first and second SVM CUDA versions. In the first case, the number of threads is equal to the number of support vectors, while in the second case, the kernel is processed by a single thread. The main reason for this choice is that the binary probability computation is a very efficient task to be processed in serial. The idea is to reproduce a serial processing on the device (avoiding a further memory transfer) even if the GPU working frequency is lower than the CPU one.

The last kernel computes the multiclass probabilities. In this case, the number of threads is equal to the number of classes: each thread evaluates the probability of a pixel to belong to that class. Then, the *cublasIsamax* function determines the class with the highest probability for each pixel. This function also transfers the output (i.e., the pixel labels) to the host, where the percentage of pixels classified to each PSL class is shown (*PSL probability*).

Concerning the third CUDA version, it is again based on the consideration that the binary classification performs very efficiently on serial processors. Therefore, this computation has been moved to the host side to evaluate if it is better to transfer back data, performing the elaboration on the host, or if a serial kernel is the best solution. The flowchart of this version is shown in Figure 5, where it is possible to see that, after the distance computation, there are data transfer to the host to allow the binary computation on the CPU. In particular, for each pixel, an array with a number of elements equal to the number of support vector is transferred to the host. The binary probability computation result is then transferred back to the device. This result has a dimension equal to the number of pixels by the number of binary classification problems. The kernels related to the distance computation and the evaluation of multiclass probabilities are not changed compared to the previous CUDA versions. Moreover, in this case, the SVM result will be the output of the system, indicating the user the percentage of pixels classified to each class.

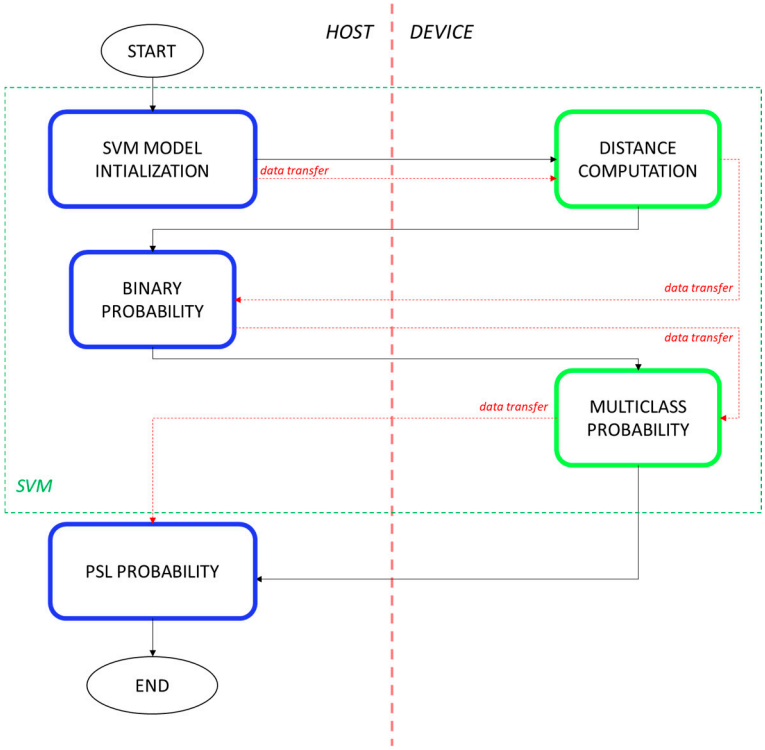

Figure 5. Flowchart of the third CUDA SVM version.

3.4. Complete Classification System

After the single algorithm parallelization, several complete system versions have been developed, integrating the serial, multicore and many-core codes. Table 1 presents fifteen versions developed in this work. In this integration, the basic idea is to find the best configuration in terms of processing time. Analyzing the single algorithms profiling, it can be concluded that all the pre-processing versions provide equivalent performance. For this reason, all the versions for the final configuration have been evaluated. In particular, even if the CUDA performance is similar to the serial one, it was decided to quantify if an initial data transfer can give benefits to the subsequent steps of the processing chain. In fact, concerning the K-means clustering, only the CUDA version has been included in the different complete systems, since it largely outperforms the serial and OpenMP processing. Thus, the speedup of the multicore and many-core K-means versions, compared to the serial processing, are about 1.5× and 6×, respectively. It is worth noticing that the similarity evaluation is always performed on the host. This is because it is part of the final output that is always managed by a CPU. Finally, all the SVM versions are considered in the integration for the final system. It has been also developed a configuration (not included in Table 1) that considers all the serial versions and that is used as a reference to compute the final speedup.

Considering the first ten versions (Figure 6a), the pre-processing is managed by the CPU since in the V1-V5 and in the V6-V10 versions, the serial (yellow box in Figure 6a) and OpenMP (orange box in Figure 6a) codes are included, respectively. In all these cases, the pre-processed image is transferred to the device before the K-means execution. On the other hand, in V11-V15 (Figure 6b), after the calibration and bands removal (*pre-processing A* in Figure 6b), the image is transferred to the GPU for the next pre-processing steps (*pre-processing B* in Figure 6b). As already mentioned, the K-means is performed on the device and its output is transferred to the host to elaborate the similarity evaluation. On the base of the considered SVM version, its parameters are transferred on the GPU (violet boxes in Figure 6a,b) or kept on the host memory for a serial or OpenMP elaboration (yellow and orange boxes in Figure 6a,b). At the end, if the SVM is performed on the GPU, the final result is transferred to the host to generate the final segmentation map.

Table 1. Different versions of the classification framework, integrating the serial (S), OpenMP (O), and CUDA (C) codes of the single algorithms. C1, C2, and C3 refer to the three SVM CUDA versions.

	Pre-Processing			K-Means		SVM			
	S	**O**	**C**	**C**	**S**	**O**	**C1**	**C2**	**C3**
V1	×			×	×				
V2	×			×		×			
V3	×			×			×		
V4	×			×				×	
V5	×			×					×
V6		×		×	×				
V7		×		×		×			
V8		×		×			×		
V9		×		×				×	
V10		×		×					×
V11			×	×	×				
V12			×	×		×			
V13			×	×			×		
V14			×	×				×	
V15			×	×					×

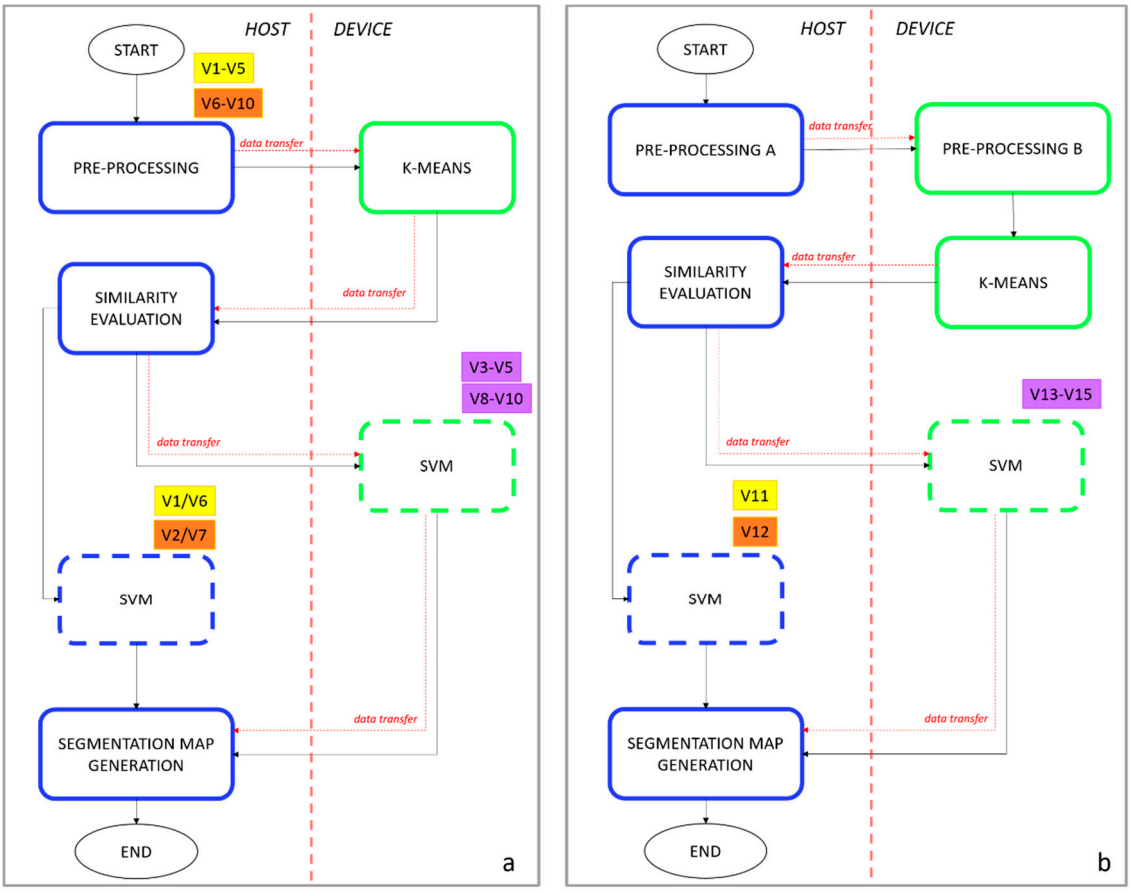

Figure 6. (**a**) Flowchart of the first ten versions (V1–V10). (**b**) Flowchart of the last five versions (V11–V15). The dashed boxes indicate that there is the possibility to perform the algorithm on the host or on the device. The yellow boxes refer to the serial processing, the orange boxes refer to the OpenMP processing, and the violet boxes refer to the CUDA processing. The notation "*Vi-Vj*" indicates all the versions from *Vi* to *Vj*.

4. Experimental Results and Discussion

4.1. Skin Cancer Classification Performance

In order to find the optimal configuration of the SVM, the hyperparameters for each type of kernel were adjusted using a genetic algorithm (GA) [26]. The methodology proposed was a patient stratified assignment where the labeled data was divided into three independent sets: test, validation, and training. A custom figure of merit (*FoM*) was conceived to evaluate the GA performance. This *FoM* is based on the accuracy per class (*ACC*) results to find the most balanced accuracy results for each class, where i and j are the indices of the classes that are being computed and n is the number of classes, as it can be seen in Equation (1). Finally, to evaluate the results obtained for the optimized classifier, the false negative rate per class (*FNRc*) was computed. *FNRc* reveals the misclassifications produced by the classifier and discovers which classes were misclassified. Equation (2) shows the mathematical expression of the *FNRc*, where FN_i is the number of false negatives in the *i-th* class and P is the total number of positive samples.

$$FoM = \frac{1}{2} \cdot \left(\sum_{\substack{i,j \\ i<j}}^{n} \frac{ACC_i + ACC_j}{|ACC_i - ACC_j| + 1} \right) \cdot \binom{n}{2}^{-1} \tag{1}$$

$$FNRc = \frac{FN_i}{P} \tag{2}$$

Table 2 shows the *FoM* results and the values of the optimized hyperparameters obtained with the GA algorithm for each kernel classifier. The obtained results show the SVM Sigmoid algorithm achieved the best *FoM* (60.67%), followed by the SVM Linear and the RBF (38.82% and 29.98%, respectively). Additionally, random forest (RF) [34] and artificial neural network (ANN) [35] classifiers were tested, achieving very low *FoM* performance, 27.25% and 33.55%, respectively. The MATLAB® Machine Learning and Deep Learning ToolBox™ were employed to implement the RF and ANN classifiers using as hyperparameters $nTress = 2431$ and $neurons_{per\ layer} = [1; 255; 3; 184]$. In both classifiers the hyperparameters used were the optimal ones after performing an automatic optimization procedure using a Genetic Algorithm. Considering these results, the SVM with Sigmoid kernel was selected for the HS dermatologic classification framework.

Table 2. Validation Classification Results.

Classifier	Hyperparameters	*FoM* (%)
SVM Linear	$C = -21.56$	38.82
SVM RBF	$C = 9.85; \gamma = 4.83$	29.98
SVM Sigmoid	$C = 1.54; s = -20.79; cf = -1.97$	60.67

C: *Cost*;γ: *Gamma*; cf: *Intercept Constant*;*s*: *Slope*.

Figure 7a illustrates the *FNRc* results for each validation HS image, where it is possible to observe that images *P15_C1*, *P15_C2*, *P20_C2* and *P113_C1* present an accurate identification of the diagnosed PSL, while images *P96_C1* and *P99_C1* have some pixels that were misclassified but clearly reveal the correct diagnosis. On the contrary, images *P60_C1*, *P60_C2* and *P68_C1* misclassified more than 50% of the labeled pixels. Image *P68_C1* classified 58.2% and 9.9% of the pixels as benign and atypical classes, respectively, being a malignant PSL. In summary, six out of nine images of the validation set were correctly diagnosed with the proposed classification framework based on the optimized SVM Sigmoid classifier. Figure 8a shows the qualitative classification maps obtained for the validation set where green color indicates the skin pixels, while red, orange, and blue colors represent the pixels

classified as malignant, atypical, and benign PSLs, respectively. These results also include the detailed percentage of pixels classified as each PSL in each HS cube.

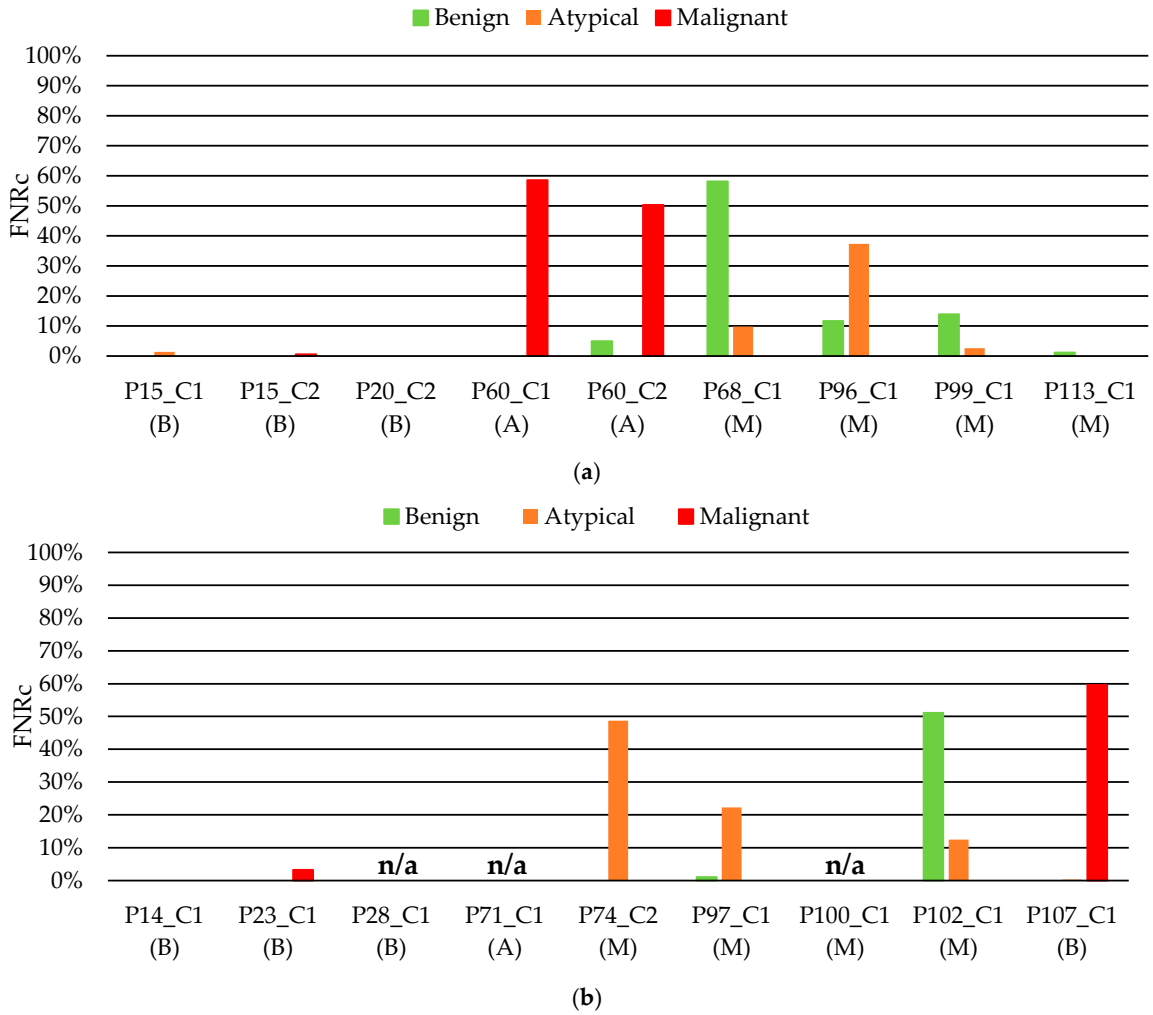

Figure 7. Classification FNRc results per each HS image obtained with the SVM Sigmoid classifier. (**a**) Validation classification results. (**b**) Test classification results. Below each patient ID, the correct diagnosis of the PSL is presented. B: Benign; A: Atypical; M: Malignant.

In order to assess the results obtained with the SVM Sigmoid classifier optimized with the validation set, the classifier was evaluated on the test set. Figure 7b shows the *FNRc* results of each HS test image. On the one hand, in the images *P28_C1*, *P71_C1*, and *P100_C1*, no pixels were identified as PSL by the segmentation stage and the classification stage could not provide the results. The lack of identification of PSL pixels in such cases occurs because the PSL spectral signatures of these HS images were highly similar to the normal skin references employed in the K-means segmentation [26]. This can be appreciated in the gray scale images of the PSLs presented in Figure 8b, where the PSL pixels of such images are quite similar to the skin pixels. These results could indicate the necessity of increasing the HS skin database for including high inter-patient variability of data. On the other hand, the PSL images *P14*_C1, *P23_C1*, and *P97_C1* were correctly identified, having in the latter one only 22.3% of pixels misclassified as atypical class. In the case of image *P74_C1*, 48.7% of the pixels were misclassified as atypical class, but the remaining 51.3% were correctly identified as malignant PSL. In the remaining images (*P102_C1*, and *P107_C1)* the misclassifications values were above 50%. *P102_C1* misclassified 51% and 12.5% of pixels as benign and atypical classes, respectively, being a

malignant PSL. Finally, in *P107_C1*, 59.6% of pixels were classified as malignant class, being a benign PSL. Figure 8b shows the classification maps of the test set.

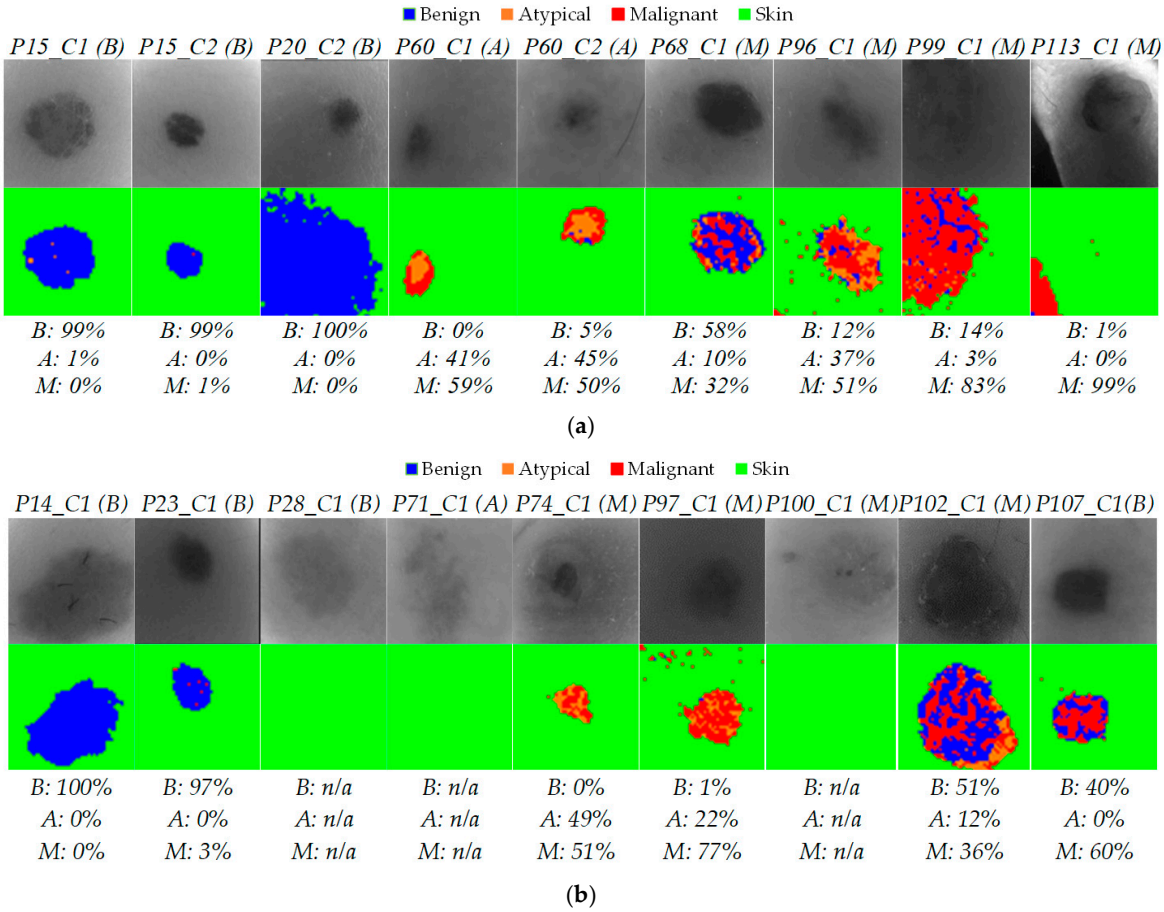

Figure 8. Qualitative classification results of each HS image. (**a**) Validation set. (**b**) Test set. On the right side of each patient ID, the correct diagnosis of the PSL is presented between brackets (B: Benign; A: Atypical; M: Malignant). The first row shows the grayscale image, while the second row shows the classification map, where skin, malignant, benign, and atypical pixels are represented in green, red, blue, and orange colors, respectively. Below the classification map, the percentages of PSL pixels classified to each class are detailed.

Figure 9 shows the processing time of the complete HS dermatologic classification framework using the test set implemented in MATLAB®. These data were obtained using an Intel i7-4790K with a working frequency of 4.00 GHz and a RAM of 8 GB.

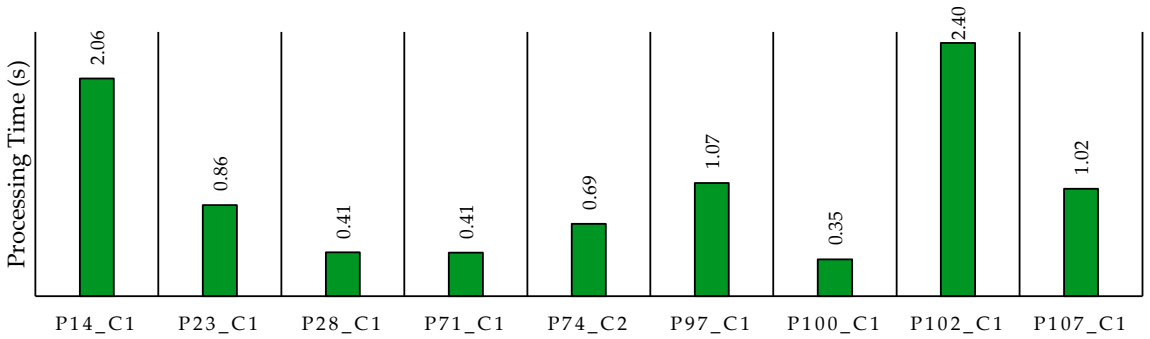

Figure 9. Processing time (in seconds) of the MATLAB execution for each HS image.

4.2. Real-Time Elaboration

The acquisition system, shown in Figure 1, takes ~1 s to capture an image with 50 × 50 pixels and 125 bands. As can be seen from the results in Section 4.1, the MATLAB implementation cannot always guarantee real-time processing. For this reason, parallel computing has been exploited to provide a real-time compliant solution. As a first step, the C serial code development provides a basis for the parallel implementation and ensures the same classification results as MATLAB. Intending to find the most efficient solution, several parallel versions presented in Section 3 have been developed, exploiting OpenMP API and CUDA framework. Moreover, different hardware devices to identify the system that best meets the processing constraints have been evaluated. Specifically, two systems have been considered: the first is a desktop PC (Test System 1-TS1) equipped with an Intel i9-9900X CPU, working at 3.5 GHz and with 128 GB of RAM. The system is also equipped with a NVIDIA RTX 2080 GPU (Turing architecture), with 2944 cores, working at 1.71 GHz and with 8 GB of RAM. The second system (Test System 2-TS2) is equipped with an Intel i7-3770 CPU, working at 3.4 GHz, with 8 GB of RAM and connected to a NVIDIA Tesla K40 GPU (Kepler architecture). This device has 2880 cores, 12 GB of RAM and its working frequency is 875 MHz.

All the code versions have been developed using Microsoft Visual Studio 2019, under Microsoft Windows 10. For all the versions, suitable compiler options have been set to generate an executable code optimized for processing speed. Moreover, the versions elaborated by a GPU also include the compute capability option (3.5 and 7.5 for Kepler and Turing architectures, respectively).

The processing times have been measured as the mean of five different executions. It is worth noticing that for the GPU versions also the data transfer time has been considered. Figure 10 shows the processing times for the described test systems using each HS image of the test set. As can be seen from these processing times, not all the versions are real-time compliant. An exhaustive discussion of the obtained results is given in the next section.

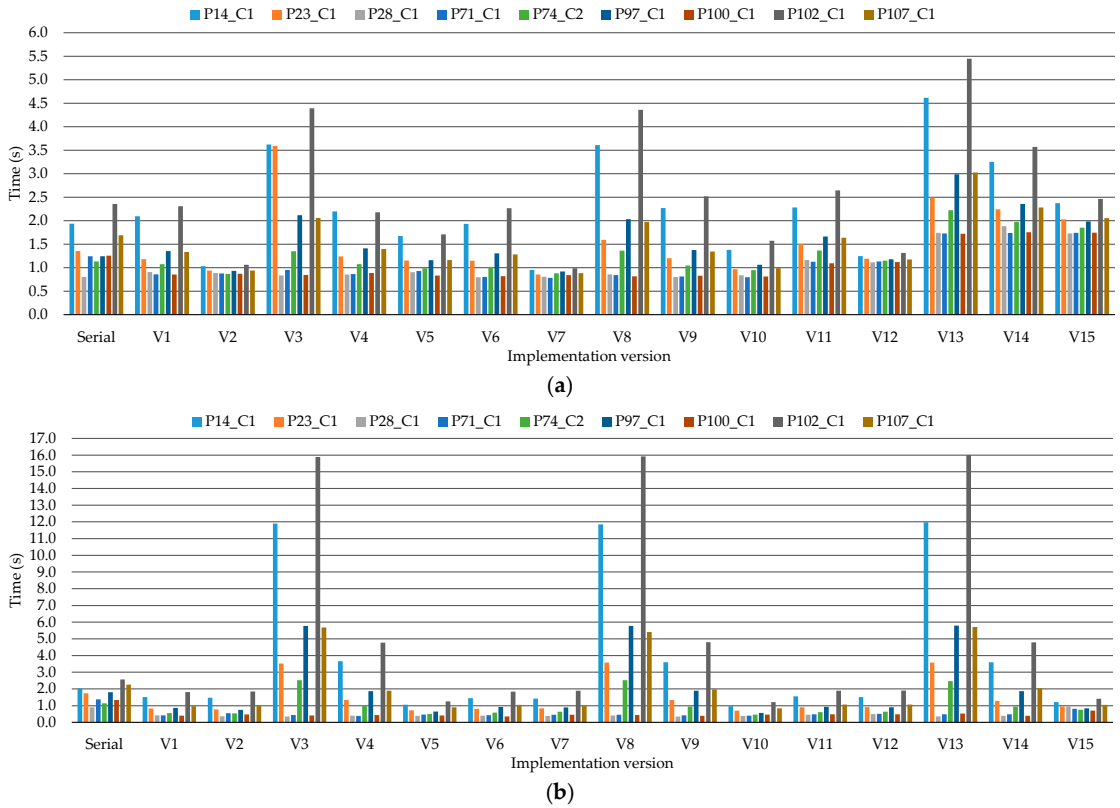

Figure 10. Processing times (in seconds) of the serial and parallel versions using the (**a**) TS1 and (**b**) TS2.

4.3. Comparison and Discussion

In the previous paragraphs, the processing chain that has been validated in MATLAB has been described. In many cases, this version is not real-time compliant, even if it exploits automatic code parallelization. In fact, some elaborations take more than one or two seconds to classify the image (Figure 9). In Figure 10, it is possible to observe that even writing the serial code in C language is not enough to achieve a real-time processing even if, in this version, memory management and mathematical operations have been optimized by hand. This version has only been developed as a basis for the parallel implementations.

Concerning the parallel processing times, the first consideration that can be made is that the results vary not only among the different versions, but also considering different images within the same implementation. In fact, since all the images have the same dimensions, this variability depends on two main factors: the former is that the number of the K-means iterations depends on the random initialization; the latter is the number of pixels to be classified by the SVM stage. Particularly, this last factor greatly changes among the images. As an example, considering the processing times of the images P28_C1, P71_C1, and P100_C1, they do not contain pixels labelled as PSL from the K-means stage and, therefore, no pixels are classified by the SVM. The elaboration of these images is real-time compliant considering both the test systems, except in the cases where the pre-processing is performed on the RTX 2080 GPU (V11-V15 on TS1). This is due to the low performance of the pre-processing on this specific GPU.

Concerning the pre-processing step of all the images, the OpenMP elaboration always outperforms the CUDA one on TS1. The same trend can be observed in most of the cases on TS2, where the multicore elaborations slightly outperform the CUDA ones. In the other cases, times are comparable. Analyzing these results in both the test systems, it can be verified that the OpenMP pre-processing version is the most efficient one (especially V7 and V10 in TS1 and TS2, respectively). It is possible to conclude that an efficient parallelization of the filtering and normalization steps adopting a multicore approach is preferable than transferring the image to perform the pre-processing on the device. The larger gap between the OpenMP-CUDA versions in TS1, compared to the one of the other system, is due to the presence of an Intel i9-9900X CPU, equipped with ten physical cores equivalent to twenty logical ones, working at a higher frequency than the Intel i7-3770 CPU. As a final consideration about the pre-processing, this phase features a lower computational complexity than the others. Therefore, its efficient parallelization does not significantly impact on the final classification time.

Since all the parallel versions include the K-means algorithm developed in CUDA, the impact of the different SVM versions on the total processing time will be explained. Considering the TS1, whether the pre-processing is elaborated exploiting OpenMP (V6-V10) or CUDA (V11-V15), the best SVM version is the multicore one (V7 and V12). This is mainly due to the reduced number of pixels to classify with the SVM (i.e., about 45% of the 2500 pixels in the worst case). The classification on the Intel i9 CPU, with twenty cores working at a high frequency, provides better performance than the elaboration on the device, which requires also a data transfer. Moreover, in this last case, the computational load is not enough to efficiently exploit the GPU cores. Finally, comparing V7 and V12, the former is faster than the latter and it is always real-time compliant since the OpenMP pre-processing is more efficient than the CUDA one.

On the other hand, if the pre-processing is performed in serial, the V2 version is the best solution. The same consideration about the SVM parallel implementation can be done also in this case. As a final remark, the V2 and V7 are the two best solutions. However, only the V7 is always real-time compliant.

These considerations cannot be made on TS2. In fact, in this case, there is not a significant gap between the performance of the multicore elaboration compared to the many-core one. Thus, let consider the versions with an OpenMP pre-processing (V6-V10): the implementation with the best performance is the one including the CUDA SVM with the binary probability computation on the host side (V10). Despite this, the elaboration times are not significantly lower than the V7 version, containing

the OpenMP SVM. Moreover, if the V11-V15 versions are considered (CUDA pre-processing), the elaboration time of V12 (OpenMP SVM) and V15 (CUDA SVM, version 3) are comparable.

It is worth noticing that, considering both the test systems, the first and second SVM CUDA versions show the worst performance (V3-V4, V8-V9, V13-V14). Considering the SVM CUDA versions, the binary probability computation is the task that mostly impacts on the performance. As said in Section 3, the binary probability computation is a very efficient task to be processed in serial. Thus, the results demonstrate that it is convenient to perform this step on the host even if the number of data transfer is increased.

Comparing the performance of the two test systems and considering the images where the SVM is not performed (P28_C1, P71_C1, P100_C1), it should be emphasized that TS2 is always faster than TS1. The Tesla K40 GPU features a lower processing time on the K-means clustering than the RTX 2080 one. The former board does not manage the graphical context of the operating system and can use all the resources to perform the computation. The latter is a standard GPU that shares resources among the graphical context management and the computation.

Summarizing, the best solutions for this application are the V7 and V10 versions for the TS1 and TS2, respectively. Even if the V7 version (TS1) shows slightly lower performances than V10 (TS2), it is the only version that always meets the real-time constraint. This parallel framework can be included in the existing prototype for its use in the routine patient examination.

5. Conclusions

In this paper, a parallel classification framework based on HSI has been presented. This framework exploited the K-means and the SVM algorithms to perform an automatic in-situ PSL identification. The framework has been validated using an in-vivo dataset and the parameters of the algorithms have been tuned in MATLAB for a later implementation of the processing framework on high-performance computing platforms (multicore CPUs and GPUs).

To ensure a real-time classification, several parallel versions, exploiting multicore and many-core technologies have been developed. Firstly, OpenMP and CUDA parallel versions of the single algorithms have been developed, which were successively integrated to provide the full parallel classification pipeline. Tests have been conducted on two different systems, equipped with an Intel i9-9900X with an NVIDIA RTX 2080 GPU (TS1) and an Intel i7-3770 with an NVIDIA Tesla K40 GPU (TS2), respectively. The best solution performed the pre-processing and the SVM stages in OpenMP, while the K-means was executed in CUDA. This version, on the TS1, is always real-time compliant since it processed 50×50 pixels with 125 bands images in less than 1 s.

This preliminary study demonstrates the potential use of HSI technology to assist dermatologists in the discrimination of different types of PSLs. However, additional research must be carried out to validate and improve the results obtained before being used during clinical routine practice using a real-time and non-invasive handheld device. Particularly, a multicenter clinical trial whereby more patients and samples are included in the database will be necessary to further validate the proposed approach.

Author Contributions: Conceptualization, E.T., H.F., and M.L.S.; methodology, E.T., R.L., and G.F.; software, E.T., B.M.-V., R.L., M.L.S., and G.F.; validation, R.L., H.F., and S.O.; formal analysis, E.T. and M.L.S.; investigation, B.M.-V. and R.L.; data curation, B.M.-V. and R.L.; writing—original draft preparation, E.T., R.L., and G.F.; writing—review and editing, M.L.S., B.M.-V., H.F., S.O., G.M.C., and F.L.; supervision, G.M.C. and F.L.; project administration, G.M.C. and F.L.; funding acquisition, G.M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Canary Islands Government through the ACIISI (Canarian Agency for Research, Innovation and the Information Society), ITHACA project "Hyperspectral Identification of Brain Tumors" under Grant Agreement ProID2017010164 and it has been partially supported also by the Spanish Government and European Union (FEDER funds) as part of support program in the context of Distributed HW/SW Platform for Intelligent Processing of Heterogeneous Sensor Data in Large Open Areas Surveillance Applications (PLATINO) project, under contract TEC2017-86722-C4-1-R. This work was completed while Raquel Leon, Beatriz Martinez-Vega and Samuel Ortega were beneficiary of a pre-doctoral grant given by the "*Agencia Canaria de*

Investigacion, Innovacion y Sociedad de la Información (ACIISI)" of the "*Conserjería de Economía, Industria, Comercio y Conocimiento*" of the "*Gobierno de Canarias*", which is part-financed by the European Social Fund (FSE) (*POC 2014-2020, Eje 3 Tema Prioritario 74 (85%)*).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Kamruzzaman, M.; Sun, D.-W. Introduction to Hyperspectral Imaging Technology. In *Computer Vision Technology for Food Quality Evaluation*; Elsevier: Amsterdam, The Netherlands, 2016; pp. 111–139. ISBN 9780128022320.
2. Torti, E.; Fontanella, A.; Plaza, A.; Plaza, J.; Leporati, F. Hyperspectral Image Classification Using Parallel Autoencoding Diabolo Networks on Multi-Core and Many-Core Architectures. *Electronics* **2018**, 411. [CrossRef]
3. Shimoni, M.; Haelterman, R.; Perneel, C. Hypersectral imaging for military and security applications: Combining Myriad processing and sensing techniques. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 101–117. [CrossRef]
4. Teke, M.; Deveci, H.S.; Haliloglu, O.; Gurbuz, S.Z.; Sakarya, U. A short survey of hyperspectral remote sensing applications in agriculture. In Proceedings of the 2013 6th International Conference on Recent Advances in Space Technologies (RAST), Instambul, Turkey, 12–14 June 2013; pp. 171–176.
5. Legnaioli, S.; Lorenzetti, G.; Cavalcanti, G.H.; Grifoni, E.; Marras, L.; Tonazzini, A.; Salerno, E.; Pallecchi, P.; Giachi, G.; Palleschi, V. Recovery of archaeological wall paintings using novel multispectral imaging approaches. *Herit. Sci.* **2013**, *1*. [CrossRef]
6. Lorente, D.; Aleixos, N.; Gomez-Sanchis, J.; Cubero, S.; Garcia-Navarrete, O.L.; Blasco, J.; Gómez-Sanchis, J.; Cubero, S.; Garc\'\ia-Navarrete, O.L.; Blasco, J. Recent Advances and Applications of Hyperspectral Imaging for Fruit and Vegetable Quality Assessment. *Food Bioprocess. Technol.* **2011**, *5*, 1121–1142. [CrossRef]
7. Khan, M.J.; Khan, H.S.; Yousaf, A.; Khurshid, K.; Abbas, A. Modern Trends in Hyperspectral Image Analysis: A Review. *IEEE Access* **2018**, *6*, 14118–14129. [CrossRef]
8. Lu, G.; Fei, B. Medical hyperspectral imaging: A review. *J. Biomed. Opt.* **2014**, *19*, 10901. [CrossRef]
9. Ortega, S.; Halicek, M.; Fabelo, H.; Callico, G.M.; Fei, B. Hyperspectral and multispectral imaging in digital and computational pathology: A systematic review [Invited]. *Biomed. Opt. Express* **2020**, *11*, 3195. [CrossRef]
10. Halicek, M.; Fabelo, H.; Ortega, S.; Callico, G.M.; Fei, B. In-Vivo and Ex-Vivo Tissue Analysis through Hyperspectral Imaging Techniques: Revealing the Invisible Features of Cancer. *Cancers* **2019**, *11*, 756. [CrossRef] [PubMed]
11. Ortega, S.; Fabelo, H.; Iakovidis, D.; Koulaouzidis, A.; Callico, G.; Ortega, S.; Fabelo, H.; Iakovidis, D.K.; Koulaouzidis, A.; Callico, G.M. Use of Hyperspectral/Multispectral Imaging in Gastroenterology. Shedding Some–Different–Light into the Dark. *J. Clin. Med.* **2019**, *8*, 36. [CrossRef]
12. Panasyuk, S.V.; Yang, S.; Faller, D.V.; Ngo, D.; Lew, R.A.; Freeman, J.E.; Rogers, A.E. Medical hyperspectral imaging to facilitate residual tumor identification during surgery. *Cancer Biol. Ther.* **2007**, *6*, 439–446. [CrossRef] [PubMed]
13. Akbari, H.; Uto, K.; Kosugi, Y.; Kojima, K.; Tanaka, N. Cancer detection using infrared hyperspectral imaging. *Cancer Sci.* **2011**, *102*, 852–857. [CrossRef] [PubMed]
14. Akbari, H.; Halig, L.V.; Schuster, D.M.; Osunkoya, A.; Master, V.; Nieh, P.T.; Chen, G.Z.; Fei, B. Hyperspectral imaging and quantitative analysis for prostate cancer detection. *J. Biomed. Opt.* **2012**, *17*, 0760051. [CrossRef] [PubMed]
15. Liu, Z.; Wang, H.; Li, Q. Tongue tumor detection in medical hyperspectral images. *Sensors* **2011**, *12*, 162–174. [CrossRef] [PubMed]
16. Halicek, M.; Fabelo, H.; Ortega, S.; Little, J.V.; Wang, X.; Chen, A.Y.; Callico, G.M.; Myers, L.; Sumer, B.D.; Fei, B. Hyperspectral imaging for head and neck cancer detection: Specular glare and variance of the tumor margin in surgical specimens. *J. Med. Imaging* **2019**, *6*, 1. [CrossRef]
17. Florimbi, G.; Fabelo, H.; Torti, E.; Ortega, S.; Marrero-Martin, M.; Callico, G.M.; Danese, G.; Leporati, F. Towards Real-Time Computing of Intraoperative Hyperspectral Imaging for Brain Cancer Detection Using Multi-GPU Platforms. *IEEE Access* **2020**, *8*, 8485–8501. [CrossRef]

18. Lazcano, R.; Madronal, D.; Florimbi, G.; Sancho, J.; Sanchez, S.; Leon, R.; Fabelo, H.; Ortega, S.; Torti, E.; Salvador, R.; et al. Parallel Implementations Assessment of a Spatial-Spectral Classifier for Hyperspectral Clinical Applications. *IEEE Access* **2019**, *7*, 152316–152333. [CrossRef]
19. Torti, E.; Florimbi, G.; Castelli, F.; Ortega, S.; Fabelo, H.; Callicó, G.M.; Marrero-Martin, M.; Leporati, F.; Torti, E.; Florimbi, G.; et al. Parallel K-Means Clustering for Brain Cancer Detection Using Hyperspectral Images. *Electronics* **2018**, *7*, 283. [CrossRef]
20. Wu, Z.; Shi, L.; Li, J.; Wang, Q.; Sun, L.; Wei, Z.; Plaza, J.; Plaza, A. GPU Parallel Implementation of Spatially Adaptive Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 1131–1143. [CrossRef]
21. Bray, F.; Ferlay, J.; Soerjomataram, I.; Siegel, R.L.; Torre, L.A.; Jemal, A. Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA. Cancer J. Clin.* **2018**, *68*, 394–424. [CrossRef]
22. LeBoit, P.E.; Burg, G.; Weedon, D.; Sarasin, A. *Pathology and Genetics of Skin Tumours*; IARC: Lion, France, 2006; Volume 6.
23. Perkins, A.; Duffy, R.L. Atypical Moles: Diagnosis and Management. *Am. Fam. Phys.* **2015**, *91*, 762–767.
24. Johansen, T.H.; Møllersen, K.; Ortega, S.; Fabelo, H.; Garcia, A.; Callico, G.M.; Godtliebsen, F. Recent advances in hyperspectral imaging for melanoma detection. *Wiley Interdiscip. Rev. Comput. Stat.* **2019**, e1465. [CrossRef]
25. Fabelo, H.; Carretero, G.; Almeida, P.; Garcia, A.; Hernandez, J.A.; Godtliebsen, F.; Melian, V.; Martinez, B.; Beltran, P.; Ortega, S.; et al. Dermatologic Hyperspectral Imaging System for Skin Cancer Diagnosis Assistance. In Proceedings of the 2019 XXXIV Conference on Design of Circuits and Integrated Systems (DCIS), Bilbao, Spain, 20–22 November 2019; pp. 1–6.
26. Leon, R.; Martinez-Vega, B.; Fabelo, H.; Ortega, S.; Melian, V.; Castaño, I.; Carretero, G.; Almeida, P.; Garcia, A.; Quevedo, E.; et al. Non-Invasive Skin Cancer Diagnosis Using Hyperspectral Imaging for In-Situ Clinical Support. *J. Clin. Med.* **2020**, *9*, 1662. [CrossRef] [PubMed]
27. Moncrieff, M.; Cotton, S.; Claridge, E.; Hall, P. Spectrophotometric intracutaneous analysis: A new technique for imaging pigmented skin lesions. *Br. J. Dermatol.* **2002**, *146*, 448–457. [CrossRef] [PubMed]
28. Monheit, G.; Cognetta, A.B.; Ferris, L.; Rabinovitz, H.; Gross, K.; Martini, M.; Grichnik, J.M.; Mihm, M.; Prieto, V.G.; Googe, P.; et al. The performance of MelaFind: A prospective multicenter study. *Arch. Dermatol.* **2011**, *147*, 188–194. [CrossRef] [PubMed]
29. Fabelo, H.; Ortega, S.; Szolna, A.; Bulters, D.; Pineiro, J.F.; Kabwama, S.; J-O'Shanahan, A.; Bulstrode, H.; Bisshopp, S.; Kiran, B.R.; et al. In-Vivo Hyperspectral Human Brain Image Database for Brain Cancer Detection. *IEEE Access* **2019**, *7*, 39098–39116. [CrossRef]
30. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
31. Caliński, T.; Harabasz, J. A Dendrite Method Foe Cluster Analysis. *Commun. Stat.* **1974**, *3*, 1–27.
32. Davies, D.L.; Bouldin, D.W. A Cluster Separation Measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *PAMI-1*, 224–227. [CrossRef]
33. Wang, L.; Zhao, C. *Hyperspectral Image Processing*; Springer: Berlin, Germany, 2016; ISBN 978-3-662-47455-6.
34. Dietterich, T.G. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*; Springer Nature: Berlin, Germany, 2000; pp. 1–15.
35. Jahed Armaghani, D.; Hasanipanah, M.; Mahdiyar, A.; Abd Majid, M.Z.; Bakhshandeh Amnieh, H.; Tahir, M.M.D. Airblast prediction through a hybrid genetic algorithm-ANN model. *Neural Comput. Appl.* **2018**, *29*, 619–629. [CrossRef]