Universitatea Transilvania din Brașov

ȘCOALA DOCTORALĂ INTERDISCIPLINARĂ

Facultatea de Inginerie Electrică și Știința Calculatoarelor

Ing. Alina-Luminița FECHETE (căs. MACHIDON)

Algoritmi și metode statistice pentru analiza datelor multidimensionale

Algorithms and statistical methods for multidimensional data analysis

REZUMAT / ABSTRACT

Conducător ştiințific

Prof.dr.ing. Petre Lucian OGRUȚAN

BRAȘOV, 2020

Universitatea Transilvania din Brașov

D-lui (D-nei) .....

COMPONENȚA

Comisiei de doctorat

Numită prin ordinul Rectorului Universității Transilvania din Braşov
Nr. ..... din .....

PREŞEDINTE:	Conf.dr.ing. Carmen GERIGAN
CONDUCĂTOR ŞTIINȚIFIC:	Prof.dr.ing. Petre Lucian OGRUȚAN
REFERENȚI:	Conf.dr.ing. Fabio DEL FRATE
	Prof.dr.ing. Romulus Mircea TEREBEȘ
	Prof.dr.ing. Cristian GRAVA
	Conf.dr. Daniel Tudor COTFAS

Data, ora şi locul susținerii publice a tezei de doctorat: .....,

.....ora ....., sala .....

Eventualele aprecieri sau observații asupra conținutului lucrării vor fi transmise electronic, în timp util, pe adresa alina.machidon@unitbv.ro

Totodată, vă invităm să luați parte la şedința publică de susținere a tezei de doctorat.

Vă mulțumim.

Universitatea Transilvania din Brașov

CUPRINS

Universitatea Transilvania din Brașov

CONTENTS

Introduction

Novelty and importance of the research topic

Today's world is overwhelmed by an unprecedented amount of data (Figure 1) often characterized by high dimensionality, which makes handling such large real-world datasets a tedious task. The solution for an effective processing and analysis in this case is applying a pre-processing task consisting of dimensionality reduction, which "mitigates the curse of dimensionality and other undesired properties of high-dimensional spaces" [22].

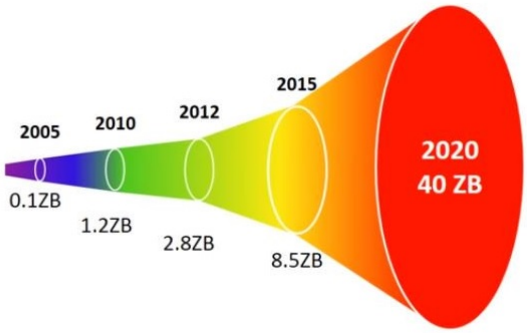


Figure 1: The explosive growth of data [6].

In particular, one type of multidimensional data which is undergoing an explosive growth is the remote sensing data. Remote sensing is generally defined as the field or practice of gathering information from distance about an object (usually the Earth's surface by measuring the electromagnetic radiation) [17].

The constant developments in the area of Remote Sensing opens up new possibilities and at the same time presents challenges regarding the analysis and processing of the large amounts of multidimensional data [20]. Consequently, a key task in the field of hyperspectral image analysis becomes managing this high volume of data; this task is currently performed by either applying dimensionality reduction techniques or selecting a subset of the available spectral bands [19].

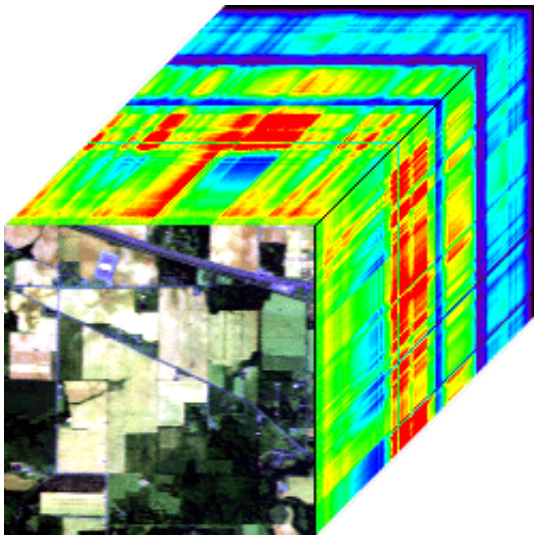


Figure 2: Data cube from an AVIRIS dataset

Dimensionality reduction (DR) techniques [64] are mathematical procedures that transform the "high-dimensional data into a meaningful representation of reduced dimensionality" [65]. Dimensionality reduction techniques are also referred to as projection methods, and are the widely used exploratory tools for applications in remote sensing due to a set of benefits that are bringing for remote sensing data analysis [15].

Research objectives

The main goal of this thesis is the design, implementation, testing, validation and optimization of a novel dimensionality reduction method, related to the Principal Component Analysis technique.

The main motivation behind this research is the desire to contribute to multidimensional data analysis by means of improving methods and techniques for dimensionality reduction, data processing and analysis, visualization and knowledge extraction.

From this main objective, several related– *specific objectives* derive:

1. *Identifying the current state of the dimensionality reduction techniques*, following the study of the literature, thus highlighting the key points that can benefit from innovative solutions.

2. *Designing a new dimensionality reduction method* that will be able to address the shortcomings of the dimensionality reduction techniques, previously identified.

3. *Implementing the new method.*

4. *Testing and validating the new method* using both synthetic and real world data to prove its efficiency, by comparing its results with other state-of-the-art methods.

5. *Optimizing the new method* in terms of both quality of the results and time of computation.

Structure and organization of the doctoral thesis

Regarding the structure of the thesis, it is organized in six chapters and includes 62 figures, 47 tables and 195 bibliographical references.

- **Introduction** presents the opportunity and motivation of the approached topic, the main objective and the specific objectives proposed for solving it within the doctoral research, as well as the description of the thesis structure.
- **Chapter 1**, "Principal Component Analysis (PCA)" presents the current state-of-the-art of the methods used for performing dimensionality reduction, with focus on the Principal Component Analysis method, the most well-established dimensionality reduction techniques. Its advantages and disadvantages, the range of its application in various domains and various adaptations are summarized in this chapter.
- **Chapter 2**, "Geometrical Approximated Principal Component Analysis" introduces a novel method, the gaPCA, as an alternative to the canonical Principal Component Analysis, based on a geometrical construction.
- **Chapter 3**, "Hyperspectral image analysis and classification" presents the results achieved after validating the novel gaPCA method in the field of hyperspectral images, for purposes of image analysis and classification. The performance of the gaPCA method was evaluated using several different metrics, for both image quality assessment, quality of the reconstruction, redundancy of the information, and accuracy of the classification and the results have been benchmarked against the canonical PCA.
- **Chapter 4**, "Face recognition" presents the results of the novel gaPCA method in the field of face recognition and its performances in terms of accuracy of the recognition, with the canonical PCA as a benchmark, are discussed.
- **Chapter 5**, "Parallelization" presents the implementation of the novel gaPCA method using parallel computing principles for accelerating the times of computation, in order to obtain significant speed-ups and improved energy efficiency.
- **Chapter 6**, "Final conclusions and original contributions", presents in a systematic approach the conclusions of the previous chapters and reviews the original contributions made in this doctoral thesis research and the issues addressed. It also presents a number of future research directions that will be pursued to continue and develop existing contributions.

CHAPTER 1

Principal Component Analysis (PCA)

According to the Merriam-Webster dictionary, the term "multidimensional" means "having or relating to multiple dimensions or aspects" [8]. Although, in the strict meaning, data sets that have more than two dimensions are considered multidimensional, the word "multidimensional" is generally used only for data sets with more than three dimensions [21].

Dimensionality reduction is "the transformation of high-dimensional data into a meaningful representation of reduced dimensionality" [37]. The goal is to find a new, reduced representation that is able to represent the original data in a lower dimensional space in such a way that the intrinsic key relations of the data are preserved.

Principal component analysis belongs to the family of Projection Pursuit (PP) methods [27], a class of statistical non-parametric methods for data analysis proposed initially by Friedman and Tukey in 1974 [27], which involves searching for directions in the multidimensional space onto which the data can be projected to reveal meaningful low-dimensional understanding of the data patterns and configuration.

1.1 Method description

The PCA algorithm produces a new set of variables, in which each one represents a linear combination of the initial features [7]. This coordinates, which are also called principal components (PCs), are perpendicular one to another and present a specific ordering (i.e. each PC points in the direction given by the maximum variance and is orthogonal to each of the previous PCs). The first PC is a vector in the original space, defined by the minimum average squared Euclidean distance between itself and the data points. The second PC is another line in space, orthogonal to the first and satisfying the same minimum squared Euclidean distance metric and so on [61] (Figure 1.1).

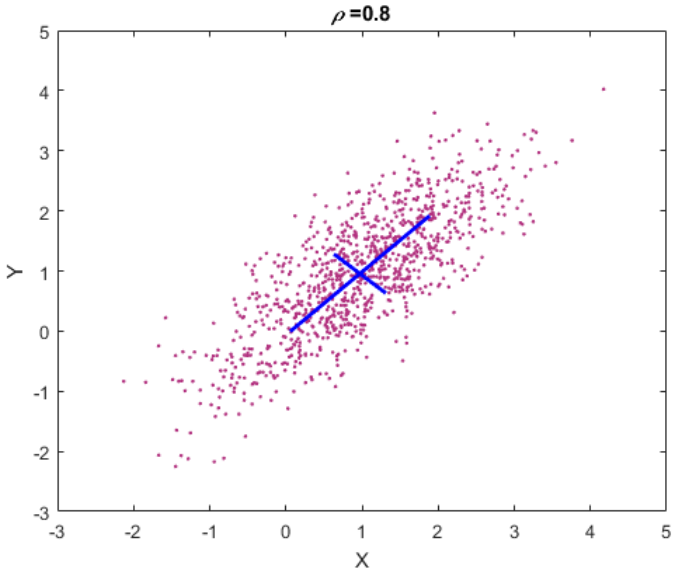


Figure 1.1: Principal component axis on a bidimensional correlated normally distributed cloud of points

1.2 PCA applications

PCA can be applied in almost all scientific disciplines for task such as dimensionality reduction, knowledge extraction, noise-reduction, visualization, compression. For example, in the field of image enhancement, PCA was used to perform contrast enhancement and denoising for night images [57]. In biology, this method was used for the prediction of protein-protein interactions [66]. In forestry, PCA was used for the analysis of soil-vegetation interrelationships [24]. In sociology, PCA was used for the analysis of socioeconomic factors and their association with malaria and arbovirus risk in Tanzania [29]. The benefits from using PCA, also extend to other fields such as business [13], forensics and art restoration, for example.

1.3 PCA strong points and limitations

Among the main advantages of PCA ([63], [28], [34]), one can enumerate:

- PCA provides a reduced representation of the data, since only a few prinicpal components are needed to retain most of the information in the original dataset.
- PCA allows the (partial) reconstruction of the input data. All dimensionality reduction method loose some information, but PCA is recognized for its ability to minimize this loss of information.
- PCA is generic. Since this method is not adjusted for a specific task, it has a high flexibility and uses in a broad field of data and applications.

- PCA efficiently reduces the correlation and redundancy in the data. The resulting variables are uncorrelated one to another.

Nevertheless, there are also some issues related to PCA ([63], [28], [34], [61]) are:

- PCA makes the assumption that the largest variances are the most important. So, principal components corresponding to greater variances are more interesting than those with smaller ones. Considering that one of the PCA goals is to find the direction of maximum variance of projection on each component (along with finding the projections that minimize the reconstruction error), means that in certain cases data with less variance can be ignored from the representation and wrongly labeled as "noise", although it may actually be of interest;

- PCA's flexibility also comes with a price, consisting in relatively higher computational requirements as compared to, e.g., the fast Fourier transform, and also inherent difficulties of parallelizing the algorithm for benefiting of the existing highly-parallel computing platforms and technologies.

1.4 PCA adaptations and approximations

Several research have explored the best ways to overcome this limitations by extending, adapting or even reinventing PCA. As a consequence, over the past decades several adaptations of the canonical PCA method were developed, focusing on specific data types, structures and applications [32], resulting in numerous PCA extensions or variants.

Various attempts to "robustifying" the canonical PCA method were proposed by the scientific literature aiming at rendering the method more immune to outliers and as a result also to the errors in the datasets [35][18]. Independent Component Analysis produces "a representation of the new variables that are independent to each other, not only uncorrelated" [38]. The Nonlinear PCA [62] addresses the linearity issue. Nonlinear PCA nonlinearly transforms the ordinal data into quantitative data [52]. This method is based on backpropagation for training a multi-layer perceptron (MLP), updating both the weights and the inputs [62].

1.5 Conclusions

This chapter provided an overview of the most well-established dimensionality reduction techniques, the Principal Component Analysis, was presented, with focus on the method description, advantages and limitations and its application in various domains.

This chapter states the importance of dimensionality reduction techniques, in particular PCA, in various fields for processing large multidimensional datasets. In the context of Big Data and the continuous provisioning of large amounts of data from sensors, acquisition devices, etc., there is an increasing amount of interest for such technique,s especially PCA based adaptations that allow improved knowledge extraction capabilities. In light of the above, this doctoral

research focused on developing enhanced algorithms for dimensionality reduction able to provide new insights and extract meaningful information from multidimensional datasets.

CHAPTER 2

Geometrical Approximated Principal Component Analysis (gaPCA)

2.1 Introducing the novel gaPCA method

gaPCA is an innovative and original method that computes the principal components of a given multidimensional dataset based on the direction given by the extremities of the distribution (the points in the dataset separated by the maximum distance), consequently providing an estimation of the direction of the canonical principal components. The standard PCA method computes the principal components based on the direction of the data's maximum variance, by calculating the eigenvectors of the covariance matrix of the dataset. The downside is, however, that these eigenvectors (being characterized by the signal's magnitude) have a tendency to overlook the information given by the (apparently) minor elements in the dataset, which are considered less contributive to the total variance.

A unique feature that differentiates gaPCA from the canonical PCA is the fact that the ordering for the gaPCA components (which are each mutually orthogonal) is the one resulted from the algorithm iterations (they are not ranked in any way), whereas in the case of canonical PCA, they are ranked according to variance. Consequently, unlike standard PCA (where the compressed information is found preponderantly in the first very few components), in the case of gaPCA, this information is more dispersed among the first components [33].

2.2 Method description

The initial step of gaPCA consists of normalizing the input dataset, by subtracting the mean. Given a set of n-dimensional points, $\mathbf{P_0} = \{\mathbf{p_{01}}, \mathbf{p_{02}}, \ldots\} \subset \mathbb{R}^n$, the mean μ is computed and

subtracted.

$$\mathbf{P_1} = \mathbf{P_0} - \mu; \tag{2.1}$$

The first gaPCA principal component is computed as the vector $\mathbf{v_1}$ that connects the two points: $\mathbf{v_1} = \mathbf{e_{11}} - \mathbf{e_{12}}$, separated by the maximum Euclidean distance:

$$\{\mathbf{e_{11}}, \mathbf{e_{12}}\} = \arg \max_{\mathbf{p_{1i}}, \mathbf{p_{1j}} \in P1} d(\mathbf{p_{1i}}, \mathbf{p_{1j}}) \tag{2.2}$$

where $d(\cdot, \cdot)$ stands for the Euclidean distance.

The second principal component vector is computed as the difference between the two projections of the original elements in $\mathbf{P_1}$ onto the hyperplane $\mathbf{H_1}$, determined by the normal vector $\mathbf{v_1}$ and containing $\mathbf{o}$, the origin:

$$H_1 = \{\mathbf{x} \in \mathbb{R}^n | < \mathbf{v_1}, \mathbf{x} >=< \mathbf{v_1}, \mathbf{o} >\} \tag{2.3}$$

with $< \cdot, \cdot >$ denoting the dot product operator. $P_2 = \{\mathbf{p_{21}}, \mathbf{p_{22}}, \ldots\}$ represents the projected original points, computed using the following formula:

$$\mathbf{p_{2i}} = \mathbf{p_{1i}} + (< \mathbf{v_1}, \mathbf{o} > - < \mathbf{v_1}, \mathbf{p_{1i}} >) \cdot \mathbf{v_1} / ||\mathbf{v_1}||^2 \tag{2.4}$$

As such, the i-th basis vector is calculated by projecting $\mathbf{P_{i-1}}$ onto the hyperplane $\mathbf{H_{i-1}}$, finding the maximum distance-separated projections and calculating their difference, $\mathbf{v_i}$.

The gaPCA algorithm is comprised of 2 major iterative phases, with each phase being executed several times, according to the intended number of principal components to be computed:

1. Determining the projection vector defined by the two extremities of the dataset (separated by the maximum distance);
2. Performing dimensionality reduction on the dataset by projecting it onto the subspace orthogonal to the previous projection.

In order to reconstruct the original data, the components scores $\mathbf{S}$ (computed by projecting all points onto the principal components) are computed (in a similar manner as for the canonical PCA) by multiplying the original mean-centred data by the matrix of (retained) projection vectors.

$$\mathbf{S} = \mathbf{P_1} \cdot \mathbf{v} \tag{2.5}$$

The original data can be reconstructed by multiplying the scores S by the transposed principal components matrix and adding the mean.

$$\mathbf{P_0} = \mathbf{S} \cdot \mathbf{v^T} + \mu \tag{2.6}$$

The final result is the set of basis vectors $\mathbf{V} = \{\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}\}$. The representation of n-dimensional data using $\mathbf{V}$ is done using the dot product operator; thus, in the new representation, the i-th component $\mathbf{c_i}$ of a vector $\mathbf{x} = [x_1, x_2, ..., x_n]$ is computed as: $\mathbf{c_i} =< \mathbf{x}, \mathbf{v_i} >$.

2.3 gaPCA algorithm

Algorithm 1 contains the pseudocode for the gaPCA method.

Algorithm 1: gaPCA

Input: $P_1 = [p_{11}, p_{12}, \ldots, p_{1m}]$, k where:
m = number of pixels of the image ;
p_{1i} = the pixel vector (n bands) of the i-th pixel ;
k = the number of principal components to be computed ;
Output: $v = [v_1, v_2, \ldots v_k]$;
e_{11}, e_{12} = computeMaximumDistance(P_1);
v_1 = (e_{11}-e_{12}) / $||v_1||$;
o = mean(P_1);
for $i \leftarrow 2$ **to** k **do**
    P_i = computeProjectionsHyperplane(P_{i-1}, o, v_{i-1});
    e_{i1}, e_{i2} = computeMaximumDistance(P_i);
    v_i = (e_{i1}-e_{i2})/ $||v_i||$;
end
return v

Algorithm 2 illustrates the pseudocode for the method that computes all the Euclidean distances between each points of a matrix P.

Algorithm 2: computeMaximumDistance

Input:$P = [p_1, p_2, \ldots, p_m]$;
Output: e_1, e_2 ;
$e_1 = 0$;
$e_2 = 0$;
$distMax = 0$;
for $i \leftarrow 1$ **to** $m - 1$ **do**
    for $j \leftarrow i + 1$ **to** m **do**
        $dist$ = EuclideanDistance($P[i]$, $P[j]$) ;
        if *($dist > distMax$)* **then**
            $distMax = dist$;
            $e_1 = P[i]$;
            $e_2 = P[j]$;
        end
    end
end
return e_1, e_2

Algorithm 3 depicts the pseudocode for the routine that calculates the Euclidean projections of each point of matrix P, on the hyperplane given by the orthogonal vector v and including

the mean point of the dataset md.

Algorithm 3: computeProjectionsHyperplane

Input: $P = [p_1, p_2, \ldots, p_m]$, o, v ;
Output: $R = [r_1, r_2, \ldots, r_m]$;
for $i \leftarrow 1$ **to** m **do**
  $R[i] = P[i] + (< v, o > - < v, P[i] >) \cdot v/||v||^2$;
end
return R

Algorithm 4 contains the pseudocode for the method that computes the Euclidean distance between vector P and vector R.

Algorithm 4: EuclideanDistance

Input: $P = [p_1, p_2, \ldots, p_m]$, $R = [r_1, r_2, \ldots, r_m]$;
Output: d ;
$d = 0$;
for $i \leftarrow 1$ **to** m **do**
  $d = d + (P[i] - R[i])^2$;
end
return sqrt(d)

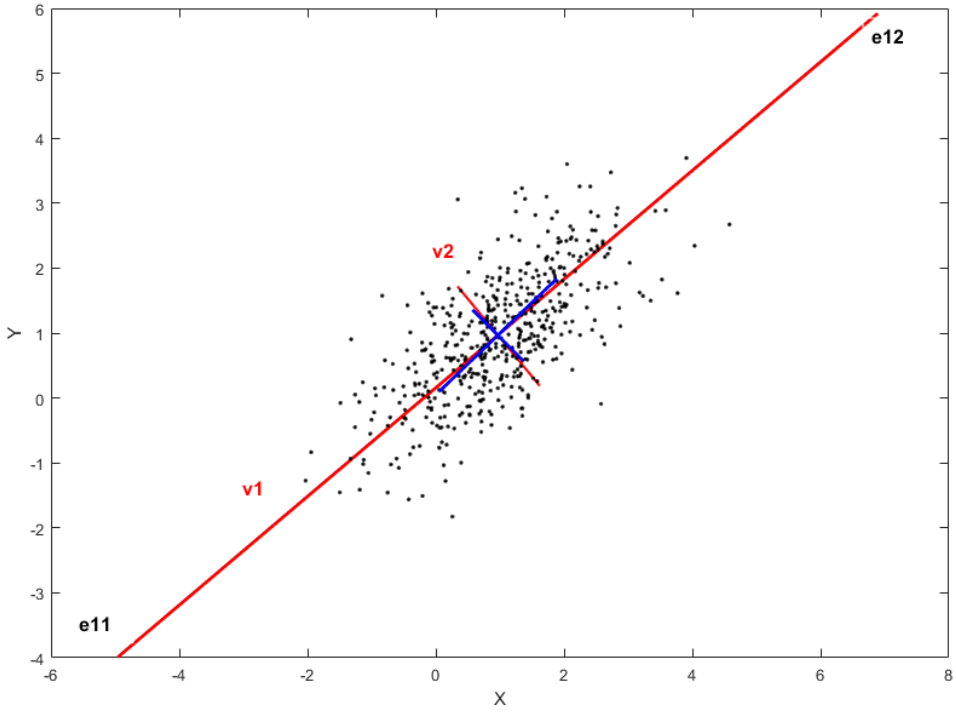


Figure 2.1: PCA (blue) vs. gaPCA (red) axes on a 2D cloud of points.

Figure 2.1 and Figure 2.2 depict a graphical illustration of how gaPCA and canonical PCA calculate the principal components for a randomly generated set of 2D points having a normal

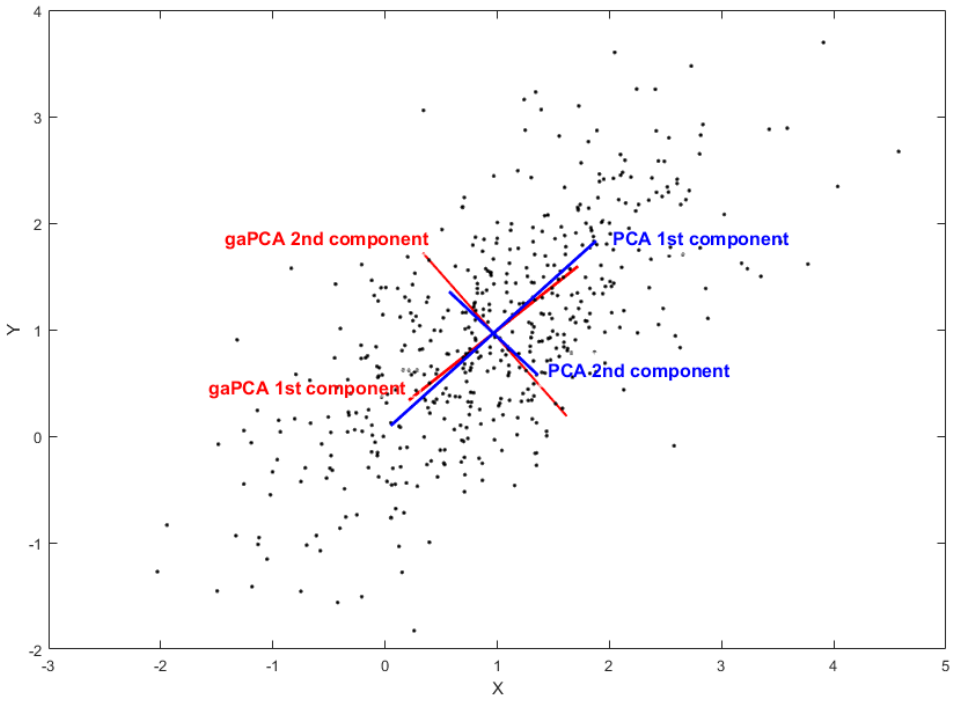


Figure 2.2: PCA (blue) axes vs. gaPCA (red) normed axes on a 2D cloud of points.

distribution. Figure 2.3 depicts the computation of the basis vectors for a generated three-dimensional set of points.

2.4 Validation on synthetic data

For validating the gaPCA method on synthetic data, 100 random sets of 1000 two-dimensional points each were generated, with a Gaussian distribution (this distribution was chosen since it is widely accepted as a statistical model for most types of data, like for example optically remotely sensed data [51]). The generated data was correlated with a correlation coefficient ranging in steps of 0.1 from 0.5 to 1. For this experiments, the data was not mean-centered prior to the computation of the gaPCA method.

In the case of each random generation, we assesed two types of metrics evaluating geometrically the closeness between the canonical PCA and gaPCA:

(i) the angle formed by the first canonical PCA component and the gaPCA one, i.e. the error angle determined by the line given by the points separated by the max distance and the 1st PCA component;

(ii) the distance between the mean of the dataset and the midpoint of the segment given by the points with the largest distance in the dataset.

Figure 2.4 illustrates three randomly-generated two-dimensional clouds of points (black)

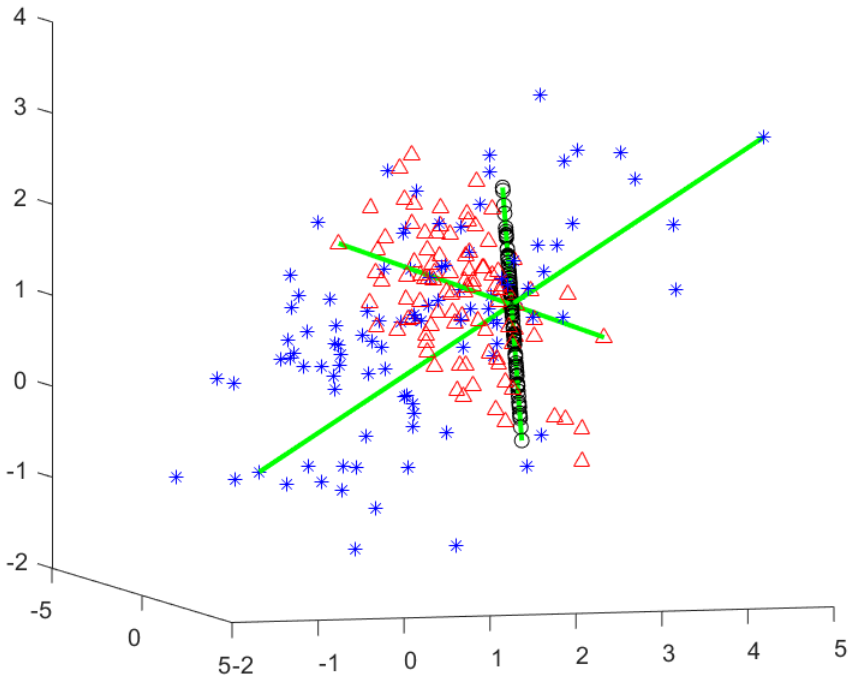


Figure 2.3: gaPCA axes (green) on a three-dimensional correlated cloud of points.

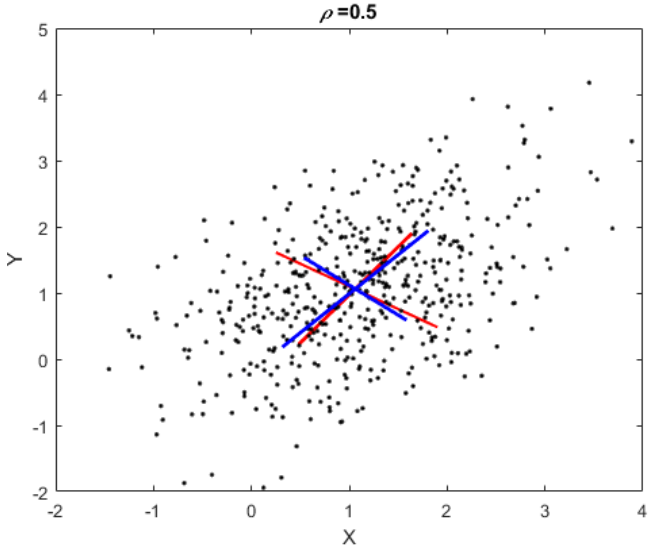
having different correlation coefficient values. For each dataset, the first 2 principal components were computed, for both gaPCA (red) and canonical PCA (blue). A decrease in the angle deviation towards very small values is noticeable for higher values of ρ. This phenomenon proves that the accuracy of the PCA-approximation provided by gaPCA increases as the variables are more correlated.

Figure 2.5 depicts the error angle and error distance as boxplot representations, varying with the correlation coefficient ρ of the dataset. These results show that on average, both in metrics (angle and distance) show a decreasing trend with increasing ρ.

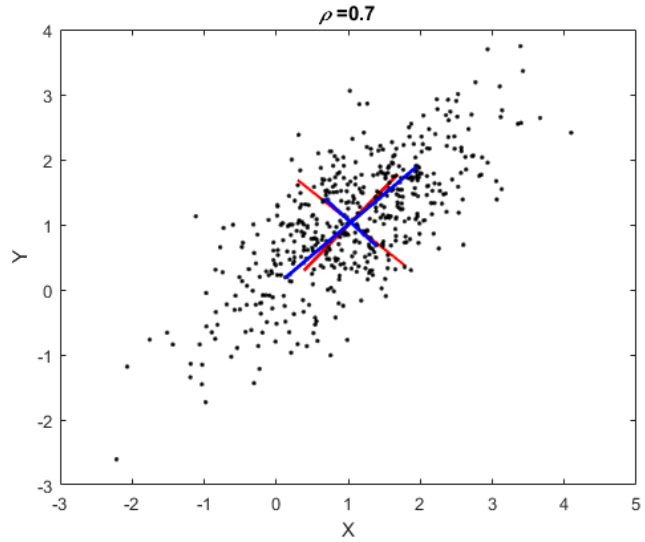
2.5 Conclusions

In this chapter, an alternative PCA algorithm based on a geometrical construction was introduced, namely the gaPCA method, which is based on the consideration that in a multidimensional dataset the segment connecting the furthest points gives a direction relatively close to the one given by the first principal component. The preliminary validation of the gaPCA method was presented, the baseline for comparison was the standard PCA algorithm.

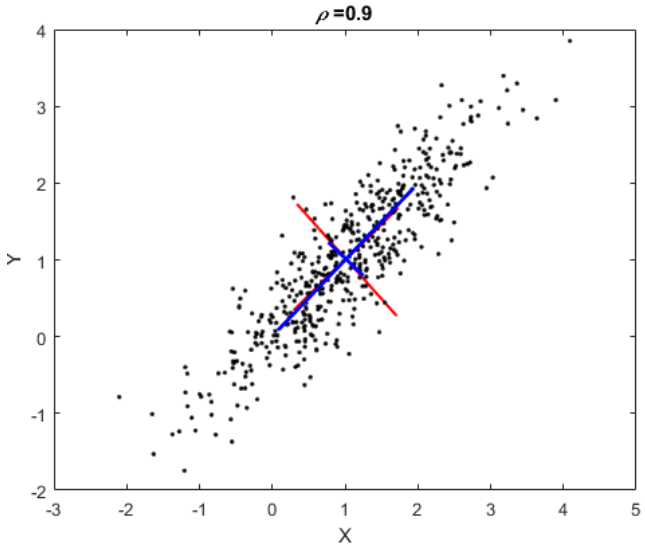
Thus, gaPCA was validated on synthetic datasets consisting of two-dimensional point clouds randomly-generated with a Gaussian distribution ensuring the normality condition. For a comparative assessment of the gaPCA performance on the synthetic data, its results were analysed versus the results of the standard PCA, by computing the error angle and error distance between the principal component axes of the two methods. The results obtained show that both error metrics, for most of the datasets are under 10%, which confirms a high fidelity



(a) ρ=0.5

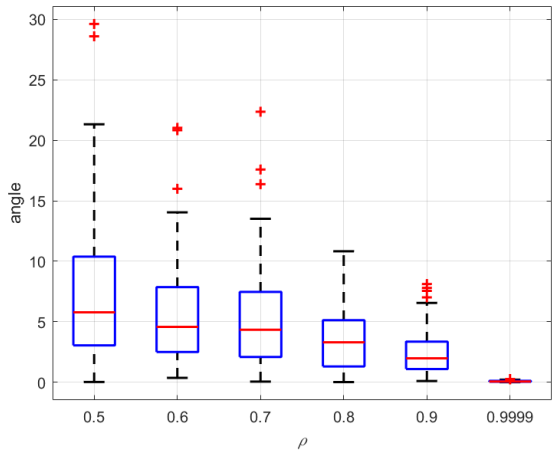


(b) ρ=0.7

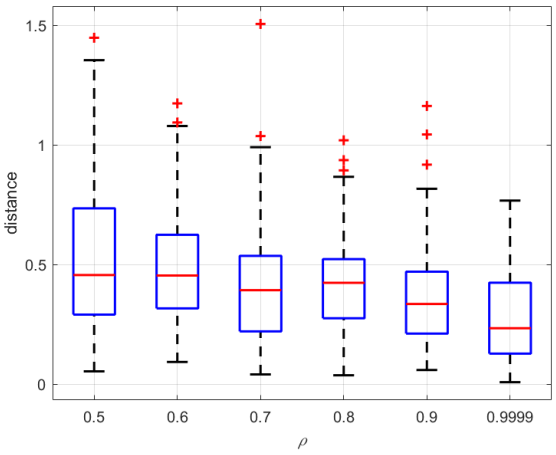


(c) ρ=0.9

Figure 2.4: gaPCA vs. PCA axes for 2D datasets having various correlation coefficients (ρ).



(a)



(b)

Figure 2.5: Angle (a) and distance (b) between gaPCA and PCA axes - boxplot representation.

of the gaPCA method compared to the standard PCA, on synthetic data.

The research and experiments presented in this chapter have been validated and disseminated in scientific publications, as indicated in the following bibliographic record:

- [46] A. L. Machidon, F. Del Frate, M. Picchiani, O. M. Machidon, and P. L. Ogrutan. Geometrical Approximated Principal Component Analysis for Hyperspectral Image Analysis. *Remote Sensing*, 12(11), 2020
- [26] L. Fasano, D. Latini, A. L. Machidon, C. Clementini, G. Schiavon, and F. Del Frate. SAR Data Fusion Using Nonlinear Principal Component Analysis. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2019 also indexed in IEEE Xplore Digital Library
- [45] A. L. Machidon, R. Coliban, O. Machidon, and M. Ivanovici. Maximum Distance-based PCA Approximation for Hyperspectral Image Analysis and Visualization. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–4, 2018 also indexed in IEEE Xplore Digital Library
- [25] L. Fasano, F. Del Frate, D. Latini, A. L. Machidon, and C. Clementini. Classification of Urban areas by Means of Multiband SAR Data Fusion. In *European Space Agency 2nd Mapping Urban Areas from Space 2018 - MUAS 2018*. ESA, 2018

CHAPTER 3

Hyperspectral image analysis and classification

The latest developments in the field of Remote Sensing (with regard to both airborne and spaceborne hyperspectral images) opens the door to new technological capabilities and opportunities, together with their inherent challenges related to the processing and analysis of large amounts of data. Hence, the increased availability of hyperspectral images is responsible for a widening of the information spectrum, but also for a considerable increase in the computational complexity that comes with large datasets. One of the main challenges is represented by the requirement to handle a large number of hyperspectral bands, which consequently increases substantially the processing time and the associated computational complexity. In this context, the efficient and accurate reduction of the remotely sensed data (or the optimum selection of the relevant hyperspectral bands for a particular application) represents a critical task in hyperspectral image analysis [60].

PCA, as a well-established dimensionality reduction technique, is used in hyperspectral imaging / remote sensing applications as a pre-processing method for several purposes. The most widely-used applications for PCA in the domain of remote sensing, comprise of land and image classification [60] [53], feature recognition [14] and change detection (identifying changes in specific areas within multi-temporal images) [58], but also on image visualization [39] and image compression [23].

In this context, this chapter presents the applications of the novel gaPCA method in the area of hyperspectral remote sensing data analysis and visualization [45], [25], [47].

3.1 Performance evaluation

The gaPCA method's results have been assessed qualitatively but also quantitatively. The former criteria was applied for evaluating the principal components images (in terms of variance, Mean Absolute Error -MAE, Gray level co-occurrence matrix -GLCM textural analysis metrics:

contrast, entropy, energy), quality of the reconstruction (Signal to Noise Ratio -SNR, Peak Signal to Noise Ratio -PSNR, Root Mean Square Error -RMSE, Spectral Angle Mapper -SAM and Correlation Coefficient -CC) and redundancy of the principal components (by computing the mutual information index - MI), while the latter approach involved assessing the land classification accuracy obtained on the gaPCA principal components.

Given that PCA (among other similar algorithms) is efficaciously used in remote sensing for reducing the dimensionality of the data (and thus reducing its redundancy), extracting land cover information or for feature extraction [42], in this work an evaluation of the gaPCA effectiveness in the field of land classification was performed, and the standard PCA algorithm was used as a baseline (reference) for benchmarking.

For each data set, a different number of principal components was computed. This number was determined in each case based on the requirement to attain a very high amount of variance explained (e.g. 98-99%) through a minimum number of principal components. Based on this rule, the following number of components was computed for each dataset: Indian Pines – 10, Pavia University – 4, DC Mall – 3 and AHS – 3. The first principal components computed by both gaPCA and canonical PCA correspond to the bands of the images on which the classification was performed, using the ENVI software [4]. Regarding the classification algorithms, for all datasets and both methods, the Maximum Likelihood Algorithm (ML) and the Support Vector Machine Algorithm (SVM) were used. For evaluating the classification accuracy in each case, a set of pixels was randomly generated from the input image and these were visually inspected with regard to the groundtruth image of each dataset.

The actual classification accuracy score was computed using two metrics: the overall accuracy (OA consisting of the number of correctly classified samples divided by the number of test samples) and the kappa coefficient of agreement (k which stands for the percentage of agreement corrected by the amount of agreement that could be expected due to chance). To evaluate the statistical significance of the classification outcomes given by the two techniques, the McNemar's test [50] was performed.

3.2 Experimental results and discussion

3.2.1 GLCM textural analysis metrics

The GLCM textural analysis metrics were applied for evaluating the quality of the principal component images calculated with both methods, since the land classification task is actually performed on them. The aim of this analysis is to assess both the quality and the amount of information of the principal component images corresponding to both methods, since these factors influence the accuracy of the land classification results. The three GLCM metrics computed (contrast, energy, entropy) were calculated on both gaPCA and standard PCA component images, where the number of components calculated was the same for both methods, and it was consistent with the number used in all experiments, (Indian Pines – 10, Pavia University – 4, DC Mall – 3, AHS – 3).

Table 3.1 presents the values of the contrast metric calculated for each principal component image and averaged, in both cases (gaPCA and standard PCA). In case of the Indian Pines

and Pavia University datasets, the gaPCA principal components presented higher averaged contrast values compared to the canonical PCA, while for the remaining datasets the trend is reversed. The averaged energy metric values calculated on the principal component images produced by both gaPCA and canonical PCA for all 4 datasets are displayed in Table 3.2. The numbers illustrate that the gaPCA principal components score better in terms of image quality (the equivalent of having lower energy values) than the canonical PCA for all datasets, except the Indian Pines. With regard to the entropy metric, the results presented in Table 3.3 are perfectly correlated with the contrast metric numbers, with gaPCA having higher entropy values (and thus better scores) than canonical PCA for DC Mall and AHS, while in the other two cases the situation is reversed.

The GLCM analysis on the performance of both gaPCA and standard PCA showed that with regard to the contrast and entropy metrics, the two method behave similarly, while the energy metric proved a superior image spatial quality for the gaPCA principal components compared to the standard PCA ones, a fact that would support better land classification results.

Table 3.1: GLCM contrast metric for both methods on all datasets.

	Indian Pines	**Pavia University**	**DC Mall**	**AHS**
PCA	0.96	0.14	0.25	0.17
gaPCA	0.34	0.12	0.32	0.18

Table 3.2: GLCM energy metric for both methods on all datasets.

	Indian Pines	**Pavia University**	**DC Mall**	**AHS**
PCA	0.14	0.58	0.29	0.23
gaPCA	0.21	0.53	0.20	0.21

Table 3.3: GLCM entropy metric for both methods on all datasets.

	Indian Pines	**Pavia University**	**DC Mall**	**AHS**
PCA	6.95	5.28	6.07	6.72
gaPCA	6.61	5.17	6.38	6.75

3.2.2 Quality of the reconstruction metrics

In this subsection, the aim was to evaluate the relationship between the reconstructed and the original dataset. Because one of the main characteristics of the PCA method is its ability to preserve the main features in the data, in just a few components, several well-established metrics were used to assess the quality of the reconstructed images using both PCA and gaPCA. In the first part of this subsection, the experiments aimed to evaluate the connection between the number of principal components used for reconstruction and the quality of the information preserved. The Indian Pines dataset was used for performing the comparative analysis. The

number of principal components that have been used for reconstruction ranged between 1 and 200 for both methods, in order to better see how the number of principal components used for the reconstruction affects the quality of the reconstructed image. The first metric evaluated was the RMSE, computed between the initial image and the reconstructed one from the canonical PCA and the gaPCA principal components. The results are displayed in Table 3.4 and represented in Figure 3.1(a).

Table 3.4: RMSE for the Indian Pines dataset.

	1PC	**2PC**	**10PC**	**100PC**	**200PC**
gaPCA	0.06	0.05	0.02	0.00	0.00
PCA	0.22	0.13	0.08	0.05	0.00

The results show that the RMSE of gaPCA are significantly lower than those of PCA. While the errors converge to zero for both methods, for gaPCA the results are slighlty better, especially in the lower dimensional spaces (up to 10 PCs computed, which is often the case in real world applications).

The second metric used was the SNR. The values computed between the initial image and the reconstructed one from the PCA and the gaPCA principal components are displayed in Table 3.5 and illustrated in Figure 3.1(b).

Table 3.5: SNR for the Indian Pines dataset.

	1PC	**2PC**	**10PC**	**100PC**	**200PC**
gaPCA	13.47	15.39	24.84	42.19	275.66
PCA	10.97	24.33	26.46	35.86	303.67

One can see that gaPCA outperforms PCA in terms of SNR when one principal component is used, and in the case of 100 principal components, for example, while PCA has better results for 2 PCs and when all the PCs are used.

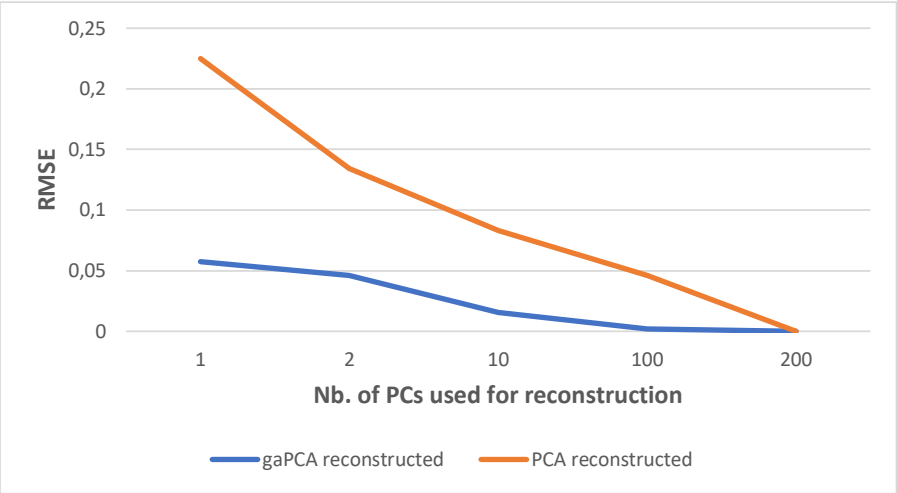
Table 3.6 and Figure 3.1(c) are showing the PSNR between the initial image and the reconstructed one using both PCA and gaPCA principal components.

Table 3.6: PSNR for the Indian Pines dataset.

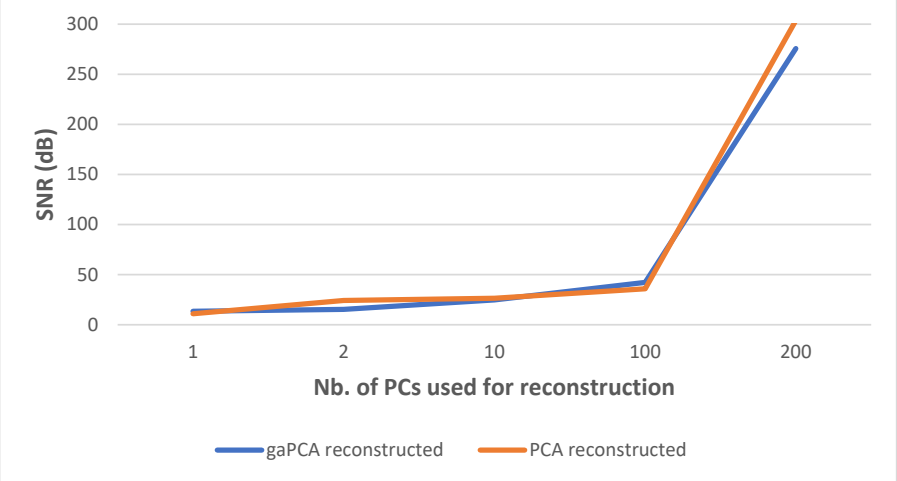
	1PC	**2PC**	**10PC**	**100PC**	**200PC**
gaPCA	24.87	26.79	36.24	53.60	287.06
PCA	22.37	35.73	37.86	47.26	315.03

The results for the PSNR are similar with those achieved with the SNR metric. Again, gaPCA shows better results when one principal component is used, and in the case of 100 principal components, while PCA scores higher for 2 and 200 PCs.

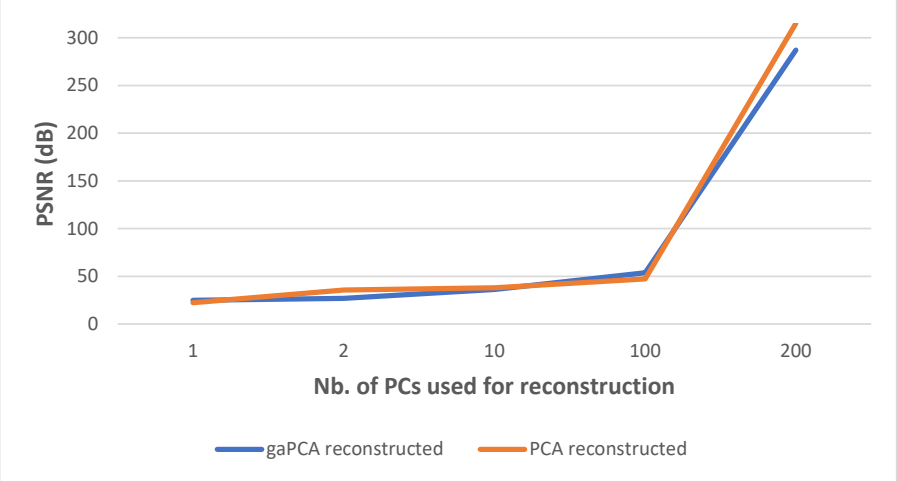
In the second part of this subsection, the experiments aimed to asses the quality of the reconstruction for all the datasets evaluated, using a fixed number of principal components, the same



(a) RMSE



(b) SNR



(c) PSNR

Figure 3.1: RMSE (a), SNR (b) and PSNR (c) computed between the original image and the canonical PCA, respectively the gaPCA reconstructions.

number that was used for classification (in Section 3.2.4). All the previous mentioned datasets were used for performing the comparative analysis. The number of principal components used for reconstruction was the same number of principal components that was used for classification: 10 for Indian Pines, 4 for Pavia University, 3 for DC Mall and 3 for AHS.

The SNR computed between the original image and the image reconstructed from the standard PCA or gaPCA principal components for all the four datasets is provided in Table 3.7 and in Figure 3.2.

Table 3.7: SNR for all datasets.

	PaviaU	**Indian Pines**	**DC Mall**	**AHS**
gaPCA	24.84	12.75	14.66	26.32
PCA	26.46	15.13	9.38	32.03

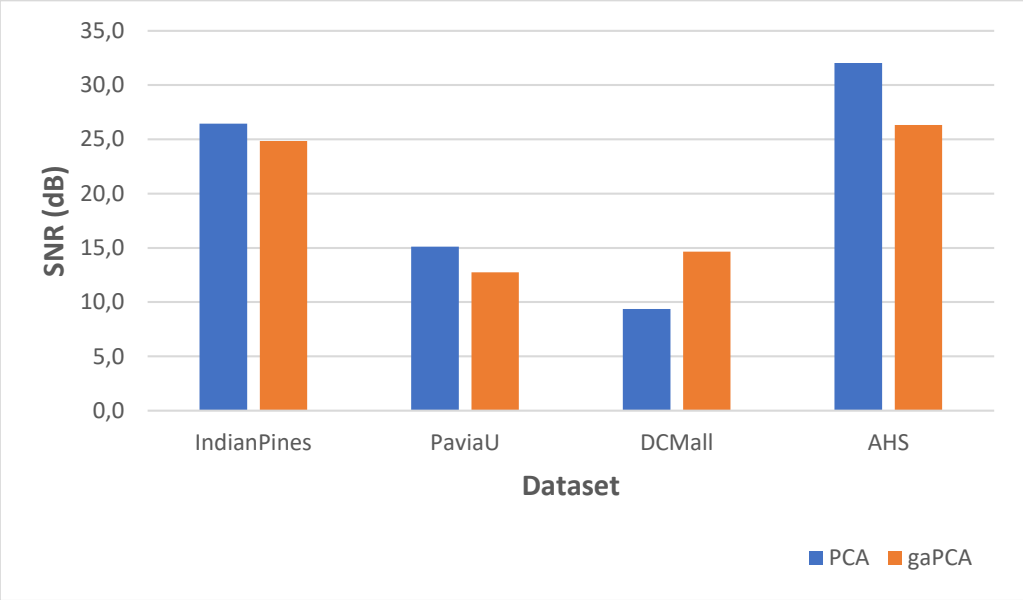


Figure 3.2: SNR between the original and reconstructed images using PCA and gaPCA for all datasets.

The results show similar scores for PCA and gaPCA in terms of SNR for all datasets. Although the differences are not significant, for three datasets, PCA has better results, while for the remaining one dataset, gaPCA scores better.

The SAM (in radians) computed between the original image and the image reconstructed from the standard PCA or gaPCA principal components is provided in Table 3.8 and in Figure 3.3. The number of principal components used for reconstruction was the same as the one used for classification: 10 for Indian Pines, 4 for Pavia University, 3 for DC Mall and 3 for AHS.

The SAM between the reconstructed spectra and the original ones is similar for both methods. Although PCA scores slightly better in terms of SAM, the differences are very small.

The CC computed between the original image and the image reconstructed from the standard PCA or gaPCA principal components is provided in Table 3.9 and in figure Figure 3.4.

Table 3.8: SAM for all datasets.

	PaviaU	Indian Pines	DC Mall	AHS
gaPCA	0.03	0.18	0.08	0.04
PCA	0.02	0.14	0.03	0.02

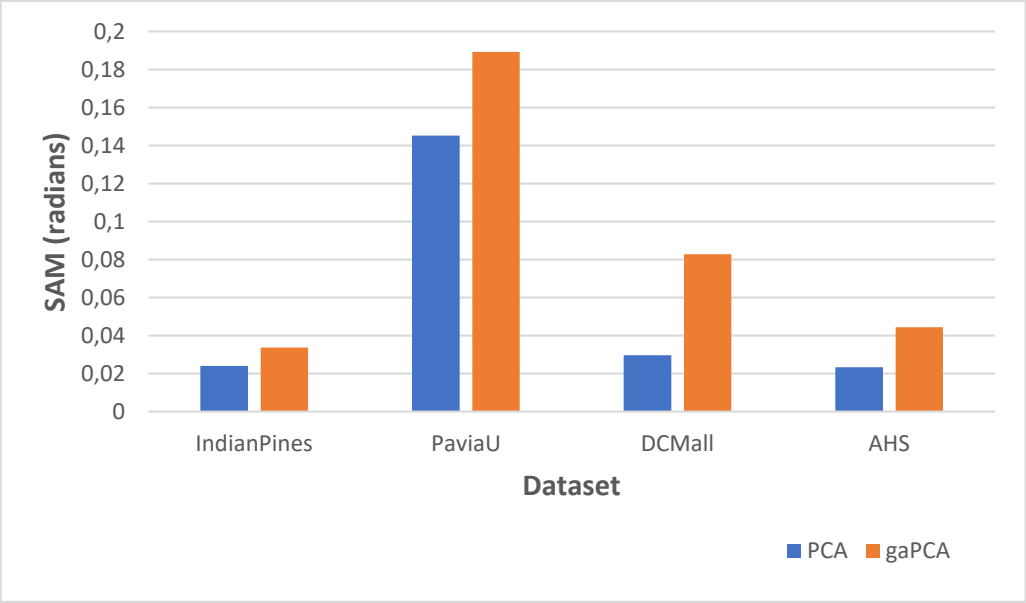


Figure 3.3: SAM between the original and reconstructed images using PCA and gaPCA for all datasets.

Table 3.9: CC for all datasets.

	PaviaU	Indian Pines	DC Mall	AHS
gaPCA	0.998	0.951	0.994	0.944
PCA	0.999	0.997	0.999	0.923

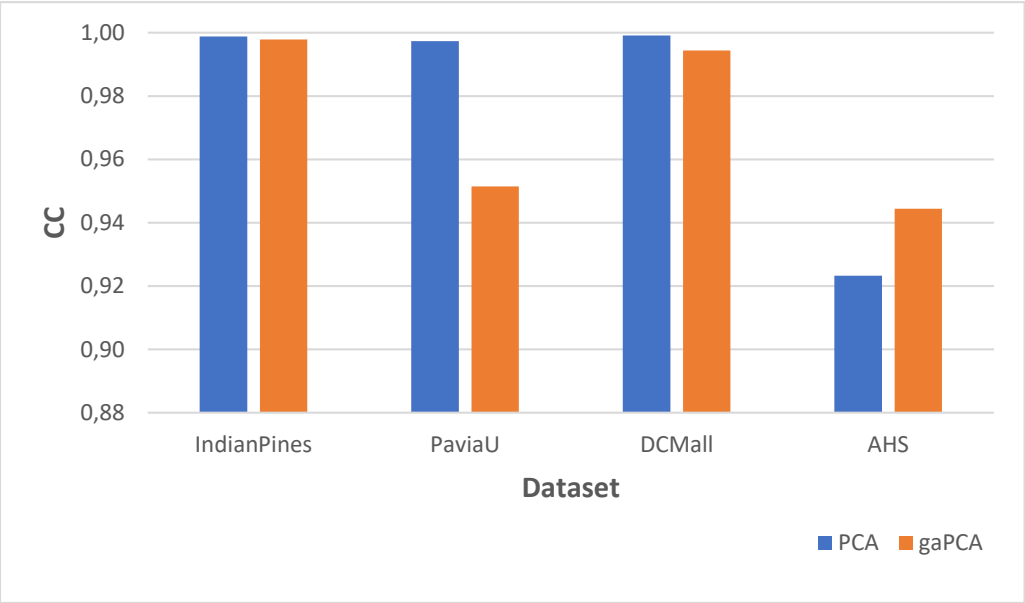


Figure 3.4: CC between the original and reconstructed images using PCA and gaPCA for all datasets.

As the results show, for most of the datasets, the CC of gaPCA and PCA are in the same range of values.

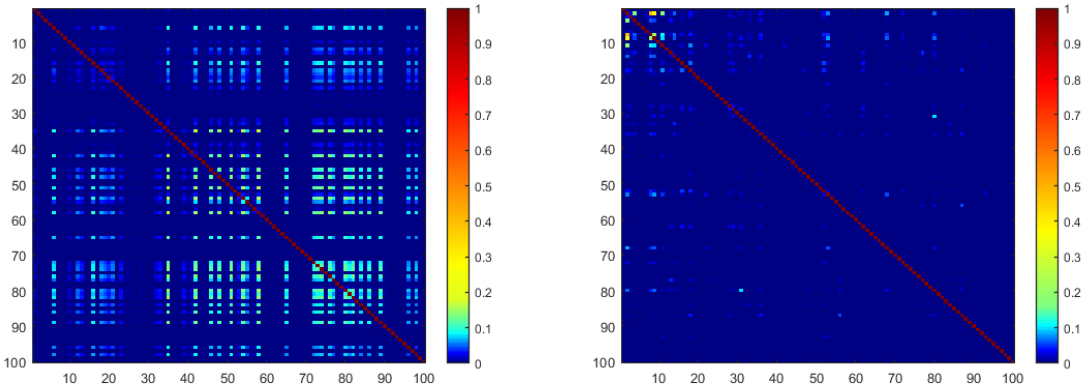
These numbers confirm that both gaPCA and canonical PCA produced similar results with regard to the SNR, PSNR, RMSE, SAM and CC reconstruction metrics, a fact confirmed also by the nearly identical slope shape in both cases and the behavior with respect to the number of principal components involved in the reconstruction.

Moreover, the results highlight that gaPCA outperforms PCA when performing the reconstruction with just the first principal components, while the situation is reversed with PCA producing more accurate results when all the principal components are used.

3.2.3 Redundancy of the principal components metric

The Mutual Information score for both standard PCA and gaPCA principal components is illustrated in Figure 3.5 as a matrix depicting the MI for each pair of components calculated with the two methods on the Indian Pines data set. The figure highlights increased MI values between the canonical PCA components (yellow and orange patches) compared to the gaPCA ones, indicating that more common information is shared by the canonical PCA components and consequently less new information is contained, which also has an impact on the classification performance, a subject that will be addressed in detail in the following sections.

Since the gaPCA components are not ranked by any metric like in the case of standard PCA, a certain degree of redundancy can be observed for the first few components, however as the visual MI matrix representation highlights, more new information is contained by the gaPCA components compared to the standard PCA ones.



(a) PCA MI

(b) gaPCA MI

Figure 3.5: MI representation for (a) canonical PCA vs. (b) gaPCA images of the Indian Pines dataset.

3.2.4 Hyperspectral image classification

As a well-established dimensionality reduction method, the standard PCA is widely used as a pre-processing routine in various Remote Sensing applications. Hence, the majority of research in this field studied the usage of PCA for achieving efficient image classification [19] [53], feature recognition [14] and identification of areas of change with multitemporal images (change detection) [58], but also on image visualization [40] and image compression [16].

Given the wide application range of the standard PCA (and PCA-based methods) in the field of Remote Sensing for dimensionality reduction, eliminating redundant data, land cover information extraction or feature extraction [42], in this work the canonical PCA method was selected as a reference for comparing and benchmarking the gaPCA method in the field of land classification.

In order to ensure consistency in the research methodology, the number of principal components computed for each dataset was chosen for ensuring the maximum variance amount explained (98-99%) with the lowest possible number of principal components. By applying this principle, the following number of components in each case were calculated: 10 - Indian Pines, 4 - Pavia University, 3 - DC Mall and 3 - AHS.

After computing the principal components using both the gaPCA and canonical PCA methods, land classification was performed on the principal component images on all 4 datasets, and the performance in terms of classification accuracy of the gaPCA method was comparatively evaluated with regard to the scores obtained by the canonical PCA and also the Nonlinear PCA method.

For this research, several classification methods have been used: Maximum Likelihood, Support Vector Machine and Neural Network. Each of these methods will be briefly described below.

The classification results were quantitatively validated using two metrics: overall accuracy (OA) and the Kappa coefficient (k). The assumption under test is that gaPCA produces

superior land classification results in terms of classification accuracy compared to the canonical PCA since it preserves a higher degree of spectral information than the standard PCA by not being focused on maximizing the variance of the data, but the range.

Each of the two PCA-based methods (canonical PCA and gaPCA) compute a predefined number of principal components which will be considered as the bands of the new multi-dimensional image which becomes the subject (input) of the land classification task. The same classification algorithms, Maximum Likelihood (ML) and Support Vector Machine (SVM), were used for all datasets and for the images obtained from the principal components computed with both methods. The classification accuracy in each case was evaluated by randomly generating a set of pixels from the input image and performing a visual inspection of the classification result vs. the groundtruth image of the dataset at the acquisition moment.

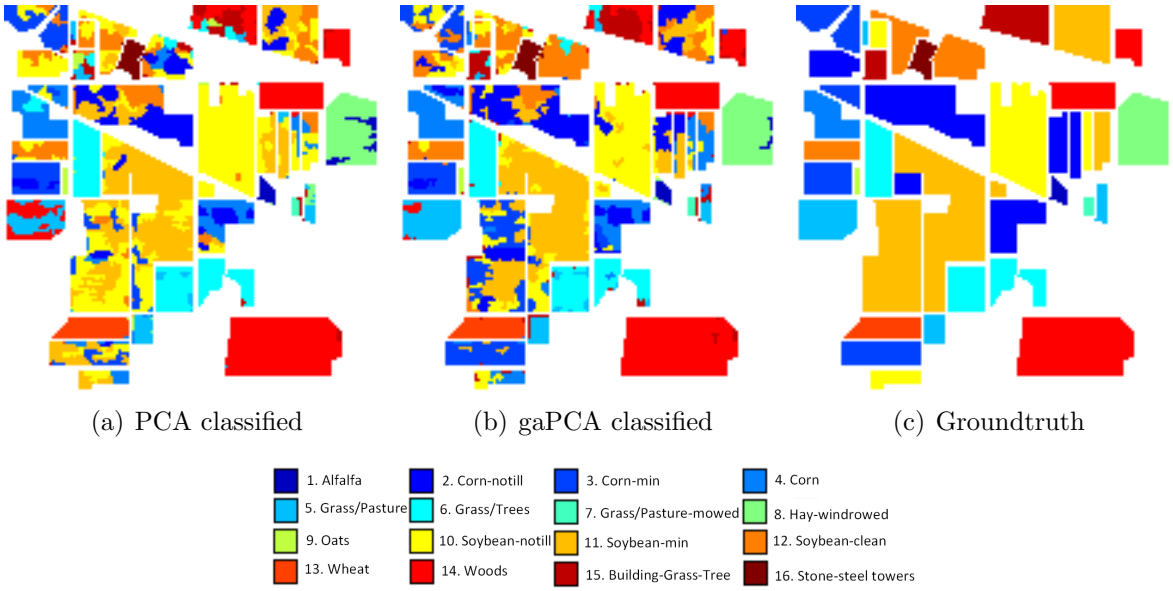
To evaluate the statistical significance of the classification results given by the two methods, the McNemar's test was performed for each classifier (ML and SVM).

3.2.4.1 Indian Pines data set

Performing land classification on the Indian Pines dataset is a difficult task given the considerable number of classes in the scene, the 20 m spatial resolution (which is quite moderate), but especially due to the high spectral similarity among the existing classes. This high similarity is caused by the site's configuration at the time of acquisition, with soybeans and corn (the two main crops in the scene) being in a rather early growth stage at that moment. Figure in Figure 3.6 (a) and (b) illustrates the classification results obtained with the canonical PCA and the gaPCA methods, together with the site's groundtruth image at the time of acquisition (c). While the figure shows there is an abundance of mixed pixels which reflect in rather noisy classified images (for both methods), the gaPCA's classification map is visibly more accurate than the one obtained by the canonical PCA method.

The classification accuracy results are presented analytically in Table 3.10, where the accuracy of both methods are summarized for each class in the scene and overall classification accuracy is also included, for both methods with both classification algorithms (ML and SVM). For testing, a set of 2000 pixels was randomly generated. The results show that the overall accuracy is higher for the gaPCA than the canonical PCA, which is also reflected by gaPCA scoring higher accuracies for the majority of classes.

One reason behind the gaPCA better classification results is that, unlike the canonical PCA method, it is not based on the hypothesis that the high variance features from the dataset are responsible for ensuring an effective discrimination among different classes while low variance features are discarded as redundant. This hypothesis is not always accurate, especially in land classification applications where the scene is comprised of spectrally similar classes, as is the case for Indian Pines. Here, one can notice the considerable differences in the scores for the two methods for sets of similar class labels; more specifically, gaPCA proved to better discriminate among the similar subsets (Corn, Corn notill, Corn mintill) and (Grass-pasture, Grass-pasture mowed) than the canonical PCA, thus confirming its increased capacity of distinguishing among similar spectral signatures.



(a) PCA classified (b) gaPCA classified (c) Groundtruth

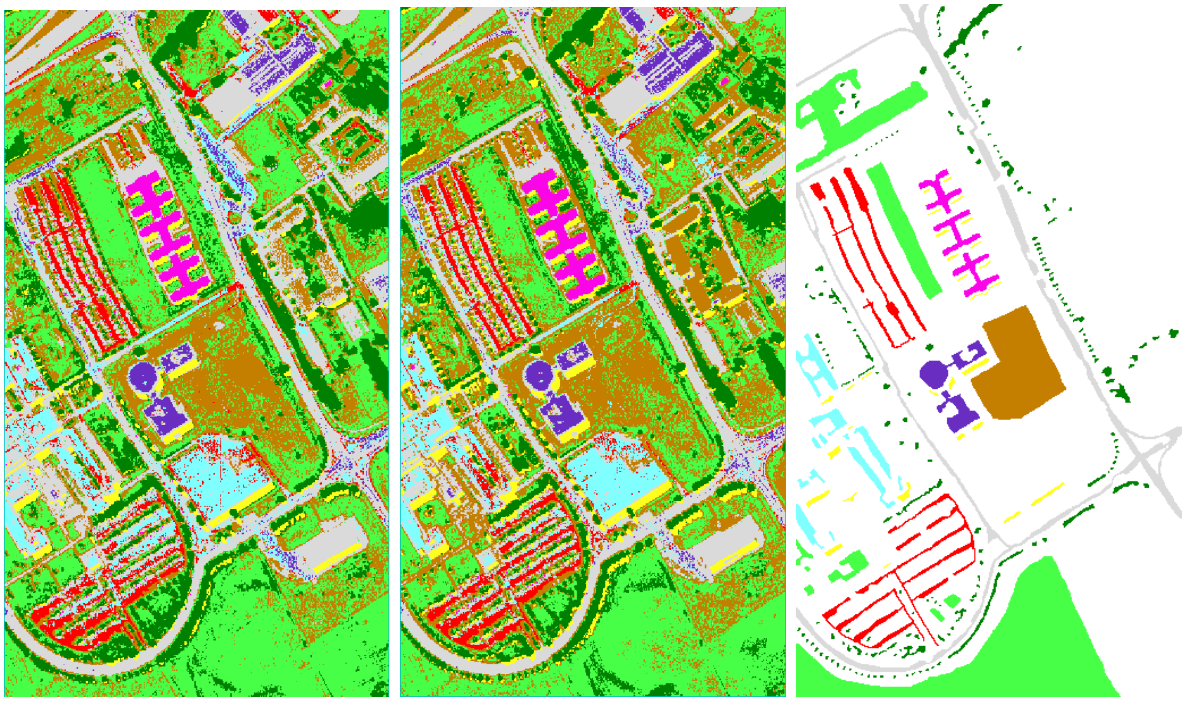
Figure 3.6: Canonical PCA (a) and gaPCA (b) images classified (Maximum Likelihood) vs. the groundtruth image (c) of the Indian Pines dataset.

Table 3.10: Classification results for the Indian Pines dataset.

Class	Training pixels	PCA ML	gaPCA ML	PCA SVM	gaPCA SVM
Alfalfa	32	98.7	80.5	18.2	18.2
Corn notill	1145	30.6	47.6	65.2	69.3
Corn mintill	595	51.6	69.2	34.9	46.1
Corn	167	84.9	100	31.4	37.7
Grass pasture	328	55.7	80.5	64.6	71.9
Grass trees	463	96.1	90.6	91.2	92.5
Grass pasture mowed	19	68.3	71.7	60	60
Hay windrowed	528	88.5	96.7	99.5	99.6
Oats	20	100	96.9	15.6	6.3
Soybean notill	681	83.7	77.1	40.9	56.1
Soybean mintill	1831	46.6	47.7	79.4	78.3
Soybean clean	457	36.9	77.7	11.8	36.1
Wheat	150	97.2	97	91.1	93.1
Woods	884	98.7	96.9	97.3	97.3
Buildings Drives	263	33.7	61.4	45.5	52.1
Stone Steel Towers	103	100	100	95.5	97.2
z_{ML}=25.1 (signif=yes)	**OA(%)**	**62.1**	**70.2**	**67.2**	**72.1**
z_{SVM}=24.8 (signif=yes)	**Kappa**	**0.57**	**0.67**	**0.62**	**0.68**

3.2.4.2 Pavia University dataset

For the Pavia University dataset, the classification results are depicted in Figure 3.7 based on the principal component images calculated with the canonical PCA (a) and the gaPCA (b) methods using the ML algorithm from Envi, compared to the scene's groundtruth (c).



(a) PCA classified (b) gaPCA classified (c) Groundtruth

Figure 3.7: Canonical PCA (a) and gaPCA (b) images classified (Maximum Likelihood) vs. the groundtruth (c) of the Pavia University dataset.

The complete classification results (including classification accuracy per class and the overall accuracy), obtained based on a 1000 set of randomly generated pixels, using both algorithms (ML and SVM) are presented in Table 3.11. These numbers show that, as in the case of the Indian Pines dataset, gaPCA outperformed the canonical PCA in the overall accuracy and in the classification accuracy for the majority of classes.

One can notice again that gaPCA is shown to have superior performance (higher classification accuracy) than the canonical PCA in the case of classes comprised of small structures and complex shapes, like asphalt or bricks. This is explained by the higher focus dedicated by the gaPCA method to smaller object and spectral classes, thus decreasing the number of false predictions for such cases, compared to the canonical PCA. For example, the confusion matrix highlights misinterpretation in the case of the standard PCA method, e.g. bricks confused with gravel, asphalt confused with bitumen.

Such confusions are caused by the increased spectral similarity between the respective classes, and not due to their spatial proximity, as highlighted in Table 3.12. The figures support the assumption that gaPCA has an enhanced ability to discriminate among similar spectral classes, since unlike the canonical PCA it is not aimed preponderantly towards classes that dominate the signal variance.

Table 3.11: Classification results for the Pavia University dataset.

Class	Training pixels	PCA ML	gaPCA ML	PCA SVM	gaPCA SVM
Asphalt (grey)	1766	60.5	61.5	67.2	78.3
Meadows (light green)	2535	68.3	80	65	86.9
Gravel (cyan)	923	100	100	33.3	40
Trees (dark green)	599	88.2	89.7	100	67.7
Metal sheets (magenta)	872	100	100	100	100
Bare soil (brown)	1579	77.8	79.4	53.2	68.3
Bitumen (purple)	565	89.7	89.7	89.7	55.2
Bricks (red)	1474	68.3	72	81.7	86.6
Shadows (yellow)	876	100	100	100	100
z_{ML}=4.87 (signif=yes)	**OA(%)**	**72.2**	**78**	**69**	**78**
z_{SVM}=5.97 (signif=yes)	**Kappa**	**0.65**	**0.72**	**0.61**	**0.72**

Table 3.12: Confusion matrix for the Pavia University dataset.

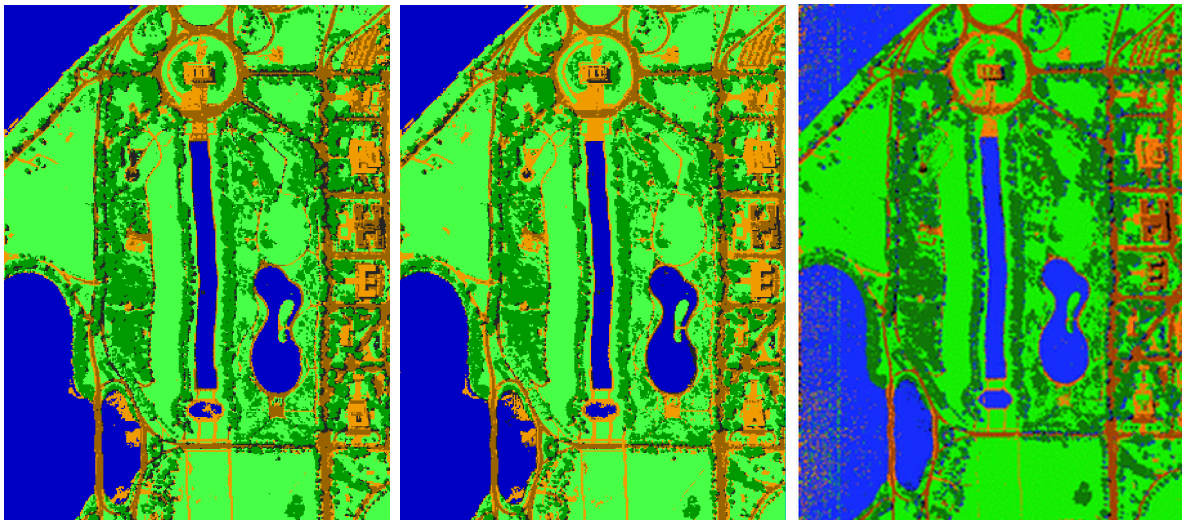
Class	**True**	**False**
Asphalt (PCA)	60.5 Asphalt	29.5 Bitumen
Asphalt (gaPCA)	61.5 Asphalt	21.8 Bitumen
Meadows (PCA)	68.3 Meadows	25.8 Bare soil
Meadows (gaPCA)	80 Meadows	17.6 Bare soil
Bricks (PCA)	68.3 Bricks	25.6 Gravel
Bricks (gaPCA)	72 Bricks	24.3 Gravel

Considering these results, gaPCA is proven to have increased accuracy in classifying smaller objects or spectral classes, reinforcing the assumption that it has a enhanced ability in retaining information related to smaller signals' variance.

3.2.4.3 DC Mall dataset

The classification results for the DC Mall dataset are depicted in Figure 3.8 based on the principal component images calculated with the canonical PCA (a) and the gaPCA (b) methods using the ML algorithm from Envi, compared to the DC Mall scene's groundtruth (c).

A complete picture of the classification results in term of overall accuracy and accuracy per each class, for both ML and SVM classification algorithms and both gaPCA and canonical PCA methods is illustrated in Table 3.13. For performing the classification assessment, a set of 140 randomly generated pixels was used, and the results show gaPCA scoring higher accuracies than canonical PCA (both overall accuracy and kappa coefficient). The gaPCA method performs consistent with the other datasets, achieving superior accuracies compared to the standard PCA in the case of small structures with complex shapes, like the Roofs & paths class (in which case it outperforms canonical PCA with over 30. Another largely spectral class for which gaPCA



(a) PCA classified (b) gaPCA classified (c) Groundtruth

Figure 3.8: Canonical PCA (a) and gaPCA (b) images classified (Maximum Likelihood) vs. the groundtruth of the DC Mall dataset.

Table 3.13: Classification results for the DC Mall dataset.

Class	Training pixels	PCA ML	gaPCA ML	PCA SVM	gaPCA SVM
Road (dark brown)	862	90	100	100	100
Trees (dark green)	413	75.9	82.7	75.9	75.9
Water (blue)	466	86.7	83.3	86.7	86.7
Grass (light green)	992	86.9	91.3	67.4	71.7
Shadows (black)	121	87.5	75	37.5	50
Roofs&paths(brown)	358	64.7	94.1	52.9	52.9
z_{ML}=2 (signif=yes)	**OA(%)**	**82**	**88**	**72**	**74**
z_{SVM}=1.13 (signif=no)	**Kappa**	**0.77**	**0.85**	**0.65**	**0.67**

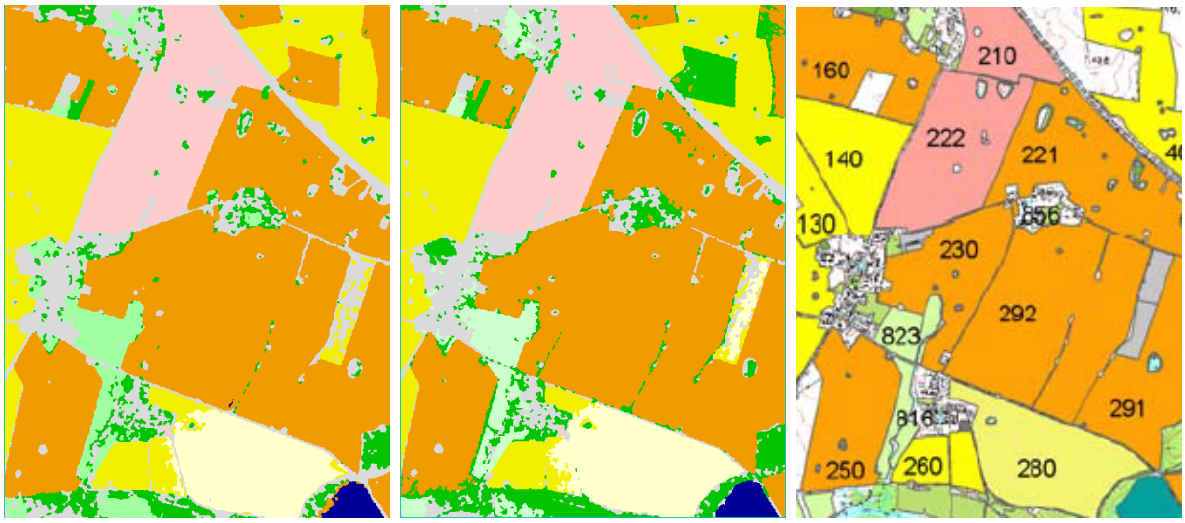
scores better classification accuracy than the canonical PCA is Trees, confirming its greater capability to preserve information associated to this specific class. Finally, gaPCA outperforms canonical PCA with regard to the overall accuracy score with over 5%.

3.2.4.4 AHS dataset

In the case of the AHS dataset, the classification results obtained based on both gaPCA and canonical PCA methods produced relatively homogeneous regions, as revealed by the classification maps shown in Figure 3.9.

The complete classification results, both in terms of class accuracy and overall accuracy, for the two methods using the ML and SVM classification algorithms are shown in Table 3.14. These numbers, obtained after assessing the results vs. the groundtruth image based on a set of 100 randomly generated pixels, report higher classification accuracies for the gaPCA method for most classes in the scene.

These results also highlight the difference in classification accuracies for both methods with regard to each class. As such, one can notice that in the case of the most extensively represented classes (e.g. oil seed rape, maize, set aside:oil seed rape) both gaPCA and canonical PCA performed very similarly. A slightly better accuracy is obtained by the canonical PCA in the case of the winter wheat class, however when moving to preponderantly spectral classes such as grassland or cutting pasture, gaPCA clearly outperforms its counterpart. The urban class is the most difficult to classify and rather confusing due to its specific nature for this particular scene (consisting of mixed structures such as buildings, local roads and the adjacent vegetation in the rural area). Overall, the results confirm gaPCA's ability to correctly classify smaller spectral classes or classes with similar or mixed pixels.



(a) PCA classified (b) gaPCA classified (c) Groundtruth

Figure 3.9: Canonical PCA (a) and gaPCA (b) images classified (Maximum Likelihood) vs. the groundtruth (c) of the AHS dataset.

The McNemar's test (z score) was computed on the classification results for all datasets and it confirms (with one isolated exception) that the gaPCA higher classification accuracy compared to the canonical PCA is statistically significant.

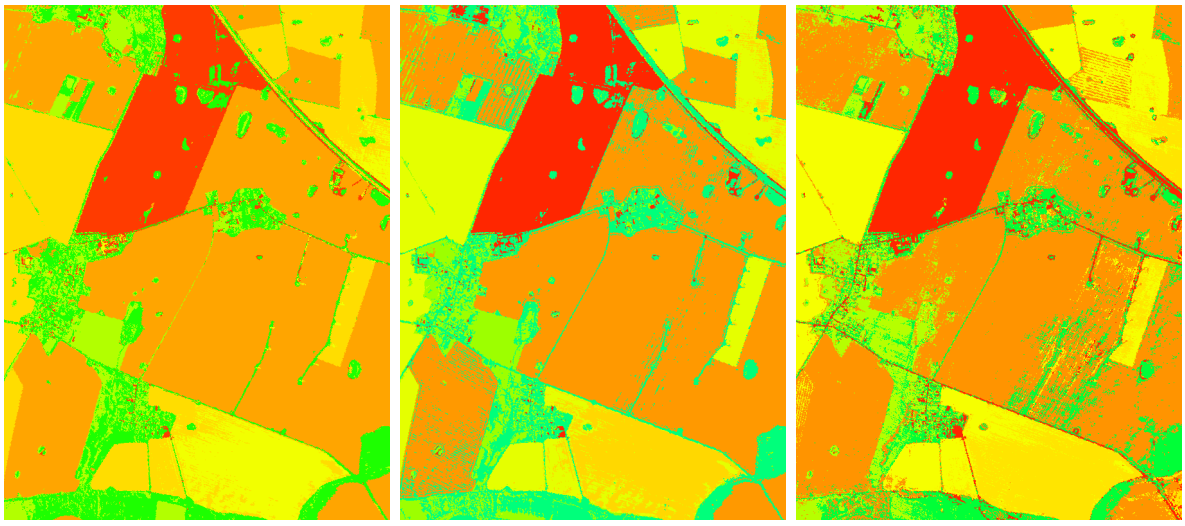
Table 3.14: Classification results for the AHS dataset.

Class	Training pixels	PCA ML	gaPCA ML	PCA SVM	gaPCA SVM
Rape (dark yellow)	2786	93.3	93.3	93.2	97.7
Rape (light yellow)	1013	80	80	90	95
Maize (pink)	969	100	100	100	100
Winter wheat (orange)	4429	100	98.1	97.3	97.3
Pasture (light green)	1788	66.7	66.7	84.6	92.3
Grassland (dark green)	1242	60	80	52.4	95.2
Urban (grey)	1079	60	90	64	92
z_{ML}=1.97 (signif=yes)	**OA**	**90.6**	**93.8**	**90**	**96.6**
z_{SVM}=3.92 (signif=yes)	**Kappa**	**0.86**	**0.91**	**0.86**	**0.95**

For achieving these results, all experiments were performed in Matlab R2018b and ENVI 5.5, running on an Intel (R) Xeon (R) X3440 CPU with 2.53 GHz and 8 GB installed memory.

3.2.4.5 gaPCA comparison with other methods

An additional experiment was performed on the AHS dataset, aiming to compare the result of the classification by means of neural networks (using Neumapper) of the standard PCA, the gaPCA and the Nonlinear PCA method.



(a) PCA (b) gaPCA (c) NLPCA

Figure 3.10: Classified images of standard PCA (top), gaPCA (center) and Nonlinear PCA (bottom) of the AHS dataset.

It is interesting to note the high similarities between methods for the most extensive represented classes of the scene (oilseed rape, maize, set aside: oilseed rape). Low differences arise in the classes winter wheat, grassland and cutting pasture, while the urban class seems to be the most confusing and difficult to classify also due to the specific of this class comprising a mix of buildings, country roads and vegetation in a rural area. The gaPCA approach's scores in terms

of classification are comparable with those achieved by the standard PCA and Nonlinear PCA aproaches.

3.3 Conclusions

In this chapter, an original PCA-based method entitled "Geometric Approximated Principal Component Analysis" (or gaPCA) was presented, and it's performance was illustrated in Remote Sensing applications involving land classification performed on multidimensional hyperspectral images. The applications described involved applying gaPCA on four hyperspectral datasets for dimensionality reduction, and evaluating its performance and results both qualitatively (by computing metrics to assess the quality of the principal component images obtained using gaPCA) and quantitatively (assessing the accuracy of the land classification task for each dataset). The evaluation of the gaPCA method was performed by calculating several objective metrics and taking as a baseline for comparison the canonical PCA method.

With regard to the performance achieved by gaPCA on the hyperspectral image land classification task for the 4 datasets, this novel method outperformed the canonical PCA in each case with regard to the overall accuracy computed for each dataset, and also scored higher than its counterpart for the majority of classes. A detailed analysis of the results in terms of class accuracy confirmed the initial hypothesis that gaPCA, unlike the standard PCA method, does not disregard the information with small contribution to the overall signal variance, which is considered as redundant or unimportant by the canonical PCA. Hence, gaPCA has a superior ability to discriminate small objects or similar classes, a feature confirmed by its performance in the experiments for the preponderantly spectral classes in each dataset, where it outperformed the standard PCA.

To conclude, the experimental results and the analysis described in this chapter showed that the novel gaPCA method is more suitable (than the canonical PCA) in Remote Sensing land classification tasks involving hyperspectral images with small structures or objects that need to be detected or where preponderantly spectral classes or spectrally similar classes are present.

The research and experiments presented in this chapter have been validated and disseminated in the following publication:

- [46] A. L. Machidon, F. Del Frate, M. Picchiani, O. M. Machidon, and P. L. Ogrutan. Geometrical Approximated Principal Component Analysis for Hyperspectral Image Analysis. *Remote Sensing*, 12(11), 2020
- [45] A. L. Machidon, R. Coliban, O. Machidon, and M. Ivanovici. Maximum Distance-based PCA Approximation for Hyperspectral Image Analysis and Visualization. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–4, 2018 also indexed in IEEE Xplore Digital Library
- [47] A. L. Machidon, M. Ivanovici, R. Coliban, and F. Del Frate. A Geometrical Approximation of PCA for Hyperspectral Data Dimensionality Reduction. In *The ESA Earth Observation Phi-week EO Open Science and FutureEO*. ESA, 2018

CHAPTER 4

Face recognition

4.1 Dimensionality reduction for face recognition

Another well-established application domain for the PCA method in the field of computer science is the Eigenfaces project [68]. The usage of Principal Component Analysis in Face Recognition applications was first advocated by Kirby and Sirovich [36], who illustrated how PCA can be applied to form a set of basis features from a face images collection. Eigenfaces or principal component analysis (PCA) methods have demonstrated their success in face recognition, detection, and tracking [67]. Consequently, we applied gaPCA for face recognition and compared the recognition accuracy with the one obtained by using standard PCA, on four different face databases. In the rest of this chapter, we describe the datasets used, the methodology involved in performing the experiments, the accuracy metrics used, and finally present and analyze the experimental results obtained.

For analyzing the efficiency of the gaPCA and canonical PCA algorithms in the field of face recognition, four well-known open-source face datasets were used: FEI [5], Yale [12], Cambridge [3] and Labeled Faces in the Wild (LFW) [30]. For all four datasets we computed the eigenfaces using both algorithms.

4.2 Methodology

4.2.1 FEI, Yale and Cambridge datasets

In this work the face recognition task was applied on the first 3 datasets, employing first the canonical PCA method followed by the gaPCA. The eigenfaces resulted after computing each method were utilized to search, for all the faces in each test set, for the most resembling face in the related training set.

For computing the similarity score, the following formula was used (based on the inverse Euclidean distance).

The eigenfaces computed with the two methods (gaPCA and canonical PCA) were used to identify, for each face in the test set, the best matching face in the training set, based on the computed similarity score (introduced in the formula above). Finally, for both methods the overall accuracy score was calculated, for each dataset.

4.2.2 LFW dataset

The fourth face recognition experiment involving the gaPCA method was performed on the Labeled Faces in the Wild (LFW) dataset [30]. In this experiment, a sub-set of the LFW dataset was used, encompassing the subjects for which at least 100 faces were available, leading to a total of 1140 faces of 5 individuals. This subset was divided into a training set containing 70% of the total number of images, and a test set comprising of the remaining 30%. Both the gaPCA algorithm and the canonical PCA counterpart were used (separately) to perform dimensionality reduction on the training set and generate the corresponding eigenfaces. On the resulting eigenfaces, a neural network classifier was applied on the training set on which previously the dimensionality reduction was performed, applying the two PCA approaches separately to obtain the eigenfaces. The resulting reduced-dimension dataset (comprising of the eigenfaces calculated with each method) was used to train a neural network classifier, a multilayer perceptron (MLP) with one hidden layer.

The classification accuracy was evaluated for three scenarios: training the MLP with the eigenfaces generated by the canonical PCA, by the gaPCA method, or training it on the original data (raw data, without applying any dimensionality reduction). Aside from the overall classification accuracy, the evaluation also looked at the number of iterations required in each case for the MLP classifier to be successfully trained.

4.3 Results and Discussion

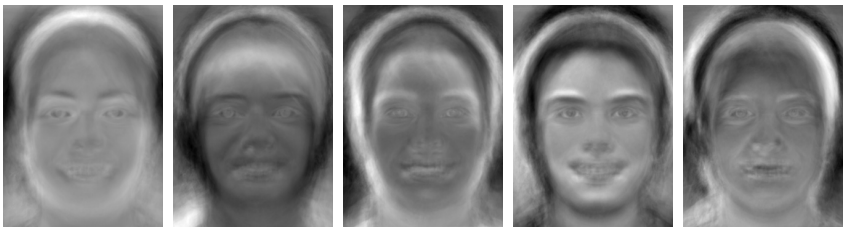
4.3.1 FEI dataset

We performed a comparison between the standard Eigenfaces method (based on the classic PCA approach) and our gaPCA method on the FEI face database [5], and used the Euclidean Distance based metric for computing the recognition accuracy in both cases.

For performing face recognition, the database was divided in 2 subsets: a training subset encompassing approximately 85% of the images and a test set with the rest of 15%. On this training subset, the PCA algorithm was applied, both in its standard form and in the approximated one. Figure 4.1 shows the first 10 eigenfaces obtained with the standard PCA method, while Figure 4.2 displays the corresponding eigenfaces after the implementation of the gaPCA on the training subset of 350 images from the FEI database.

The plot in Figure 4.3 shows the cumulative eigenvalues for the first 200 principal components. As expected, the first 100-150 principal components encompass 95-98% of the variance of the

(a) 1st PCA eigenface (b) 2nd PCA eigenface (c) 3rd PCA eigenface (d) 4th PCA eigenface (e) 5th PCA eigenface



(f) 6th PCA eigenface (g) 7th PCA eigenface (h) 8th PCA eigenface (i) 9th PCA eigenface (j) 10th PCA eigenface

Figure 4.1: First ten eigenfaces from the FEI Database obtained by Eigenfaces standard PCA method.

(a) 1st gaPCA eigenface (b) 2nd gaPCA eigenface (c) 3rd gaPCA eigenface (d) 4th gaPCA eigenface (e) 5th gaPCA eigenface

(f) 6th gaPCA eigenface (g) 7th gaPCA eigenface (h) 8th gaPCA eigenface (i) 9th gaPCA eigenface (j) 10th gaPCA eigenface

Figure 4.2: First ten eigenfaces from the FEI Database obtained with the gaPCA method.

images in the data set.

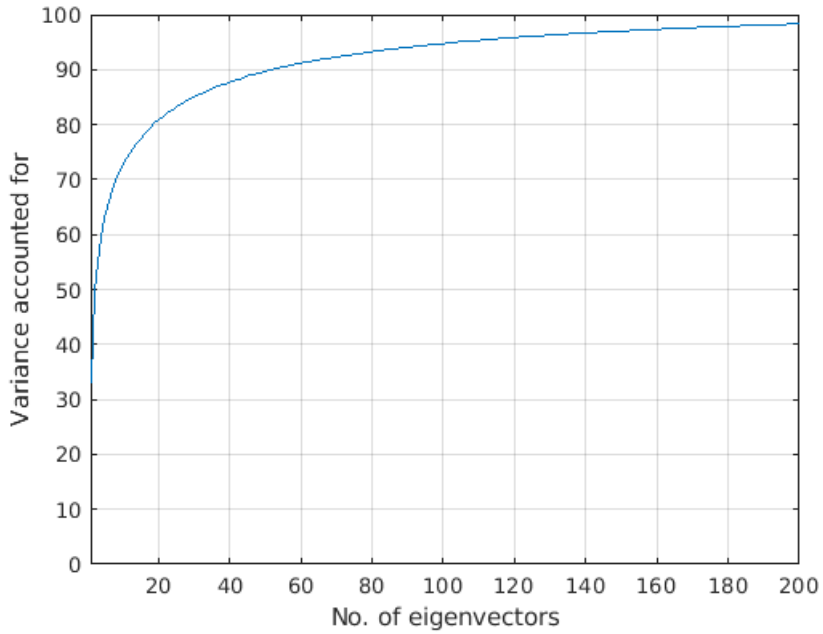


Figure 4.3: Percent of the variance explained by the first eigenvectors of the FEI dataset.

In Figure 4.4 we displayed the mean face obtained with the standard PCA (a) method and the gaPCA (b) on the training subset of 350 images from the FEI database.

(a) Average face PCA (b) Average face gaPCA

Figure 4.4: Average face from the FEI Database using the standard PCA (a) and the gaPCA (b) method.

The face recognition results show that for a relatively high percent of the images tested, (98 % for standard PCA and 92% for gaPCA in the case of 150 eigenvectors retained) the face image provided based on the similarity score comparison was correct (Table 4.1).

A slight decrease of these values is noticed for both methods (standard PCA and gaPCA) if the number of principal components retained is lowered Figure 4.5. However, it is interesting to notice that both methods score the same leap of percentages from 10 eigenvectors to 150 eigenvectors, namely 14 percents.

The analysis of the results shows that a very good performance (92% in accuracy) can be achieved with the gaPCA model. In the particular case of the FEI database, the gaPCA

Table 4.1: Face recognition accuracy for Standard PCA and gaPCA on the FEI face database.

Method	**Face recognition accuracy**
PCA	98%
gaPCA	92%

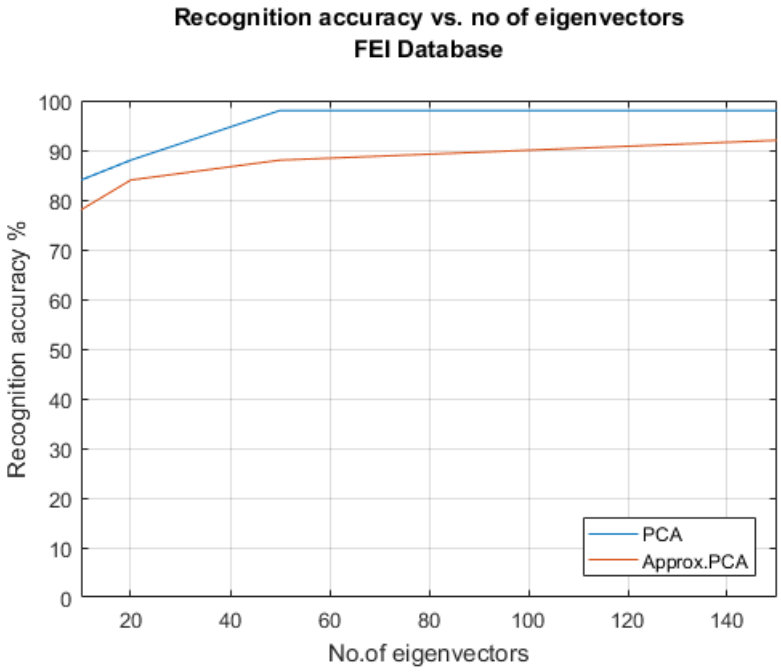
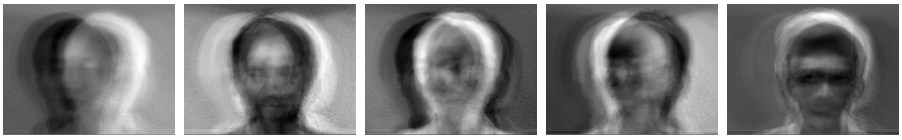


Figure 4.5: Recognition accuracy vs. no. of eigenevectors for the FEI database of faces.

performance was mainly evaluated against factors such as facial expression, hairstyles, adorns (glasses, earrings, etc.). Other factors such as illumination and pose variation tend to be less prominent in relatively controlled environments.

4.3.2 Yale dataset

For the Yale dataset, the same methodology was employed: the gaPCA method was comparatively validated with regard to its face recognition performance by comparing its accuracy results with the ones obtained by the canonical PCA. As such, the eigenfaces of the Yale dataset were computed using both methods. For conducting the assessment of the face recognition performance, the dataset was divided into a training set (containing 135 images) and a test set (comprised of the remaining 30 images). In light of the Yale dataset being rather small, the top 20 eigenvectors were selected, both in the case of gaPCA and in the case of its counterpart. The first 5 principal components of this dataset are illustrated in Figure 4.6 for both methods (gaPCA – bottom, canonical PCA – top), while the face recognition accuracy results are presented in Table 4.2.



(a) 1st PCA eigenface (b) 2nd PCA eigenface (c) 3rd PCA eigenface (d) 4th PCA eigenface (e) 5th PCA eigenface

(f) 1st gaPCA eigenface (g) 2nd gaPCA eigenface (h) 3rd gaPCA eigenface (i) 4th gaPCA eigenface (j) 5th gaPCA eigenface

Figure 4.6: First five eigenfaces from the Yale Database obtained with the standard PCA (top) and the gaPCA (bottom) method.

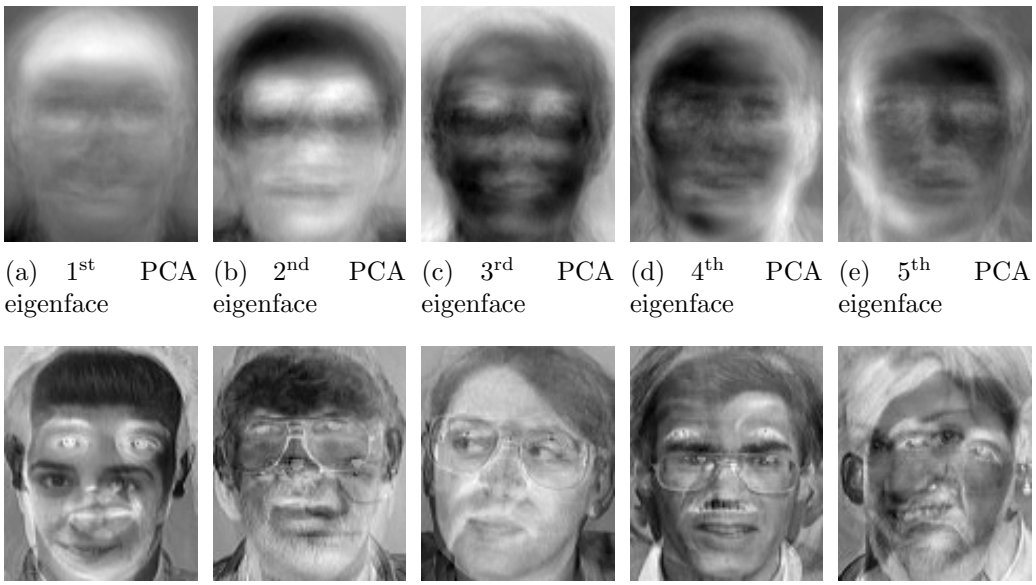
Table 4.2: Face recognition accuracy for Standard PCA and gaPCA on the Yale face database

Method accuracy	Face recognition
PCA	76.66%
gaPCA	73.33%

The percentages are smaller for both methods in the case of the Yale database than in the case of the FEI data set, mainly due to the large variation in pose positioning and illumination of the faces. However, it is interesting to point out that gaPCA's accuracy is nearly the same as the accuracy scored by the standard PCA, in the case of 20 eigenvectors retained.

4.3.3 Cambridge dataset

Figure 4.7 illustrates the first 5 principal components computed on the Cambridge dataset, using both gaPCA (bottom) and canonical PCA (top).



(a) 1st PCA eigenface (b) 2nd PCA eigenface (c) 3rd PCA eigenface (d) 4th PCA eigenface (e) 5th PCA eigenface

(f) 1st gaPCA eigenface (g) 2nd gaPCA eigenface (h) 3rd gaPCA eigenface (i) 4th gaPCA eigenface (j) 5th gaPCA eigenface

Figure 4.7: First five eigenfaces from the Cambridge Database of faces obtained with the standard PCA (top) and the gaPCA (bottom) method.

Table 4.3: Face recognition accuracy for Standard PCA and gaPCA on the Cambridge face database.

Method	**Face recognition accuracy**
PCA	94.14%
gaPCA	93.33%

The face recognition accuracies on the Cambridge dataset for both gaPCA and the canonical PCA methods are presented in Table 4.3. For performing the face recognition task, the original Cambridge dataset was divided into a training set, comprised of 280 images, and a test set, containing the remaining 120 images. On the training set, the eigenfaces were calculated with each of the two methods. The behavior of the recognition accuracy with the variation of the number of eigenvectors retained is illustrated in Figure 4.8, which highlights a rather constant accuracy for both methods regardless of the number of eigenvectors employed.

4.3.4 LFW dataset

For the LFW dataset, Figure 4.9 illustrates the first 5 eigenfaces computed with the canonical PCA method, while the ones calculated with the gaPCA algorithm are displayed in Figure 4.10.

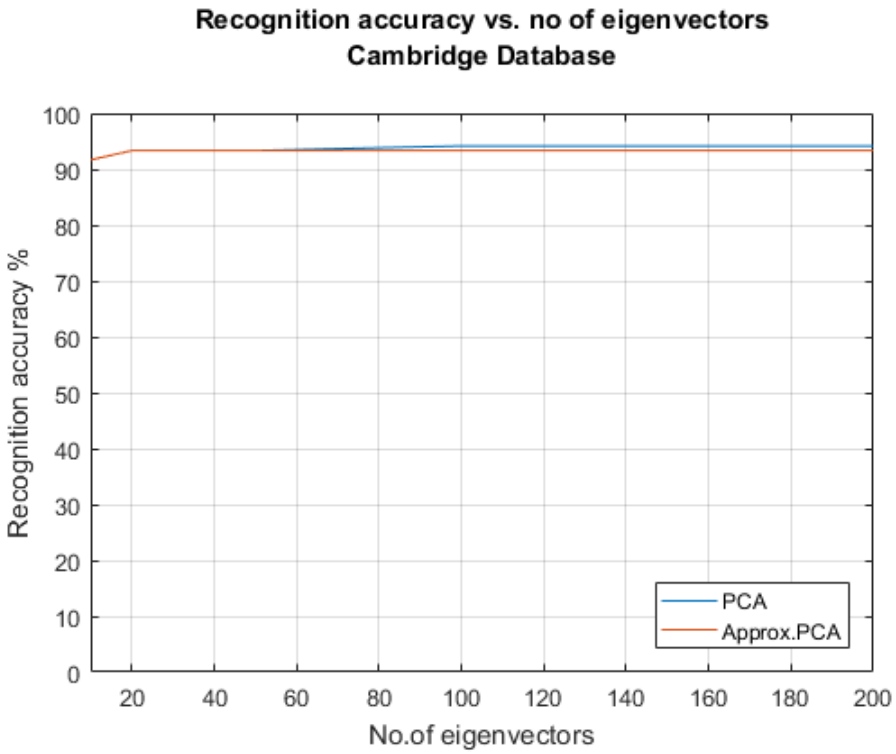


Figure 4.8: Recognition accuracy vs. no. of eigenevectors for the Cambridge database of faces.

The classification accuracy for each of the 5 subjects involved in this experiment is presented in Table 4.4, for the three scenarios: training the MLP classifier with the raw (original) data (no PCA), with the data obtained by applying the canonical PCA or the one resulted after employing the gaPCA method.

Table 4.4: Average Precision for 10 classification with NO PCA, standard PCA and gaPCA.

Personality	**NO PCA**	**PCA**	**gaPCA**
C. Powell	78%	83%	81%
D. Rumsfeld	56%	70%	72%
G.W. Bush	86%	88%	85%
G. Schroeder	53%	74%	68%
T. Blair	67%	71%	68%

The results highlight that the gaPCA method scored an average overall accuracy of 75%, very close to the 77% scored by the canonical PCA, and both methods outperformed clearly the case where no dimensionality reduction method was employed on the dataset, in which case the average overall accuracy is of only 68%. Moreover, the gaPCA outperformed its counterpart for the specific class "D.Rumsfeld ", for which it also scored higher than the NO PCA scenario. With regard to how the dimensionality reduction influences the training process of the classifier, the results in Table 4.5 present the number of iterations required for training in each of the three scenarios detailed above. It can be easily noticed that by using either canonical PCA or gaPCA to reduce the dimensionality of the training dataset, the classifier is trained almost twice as fast.

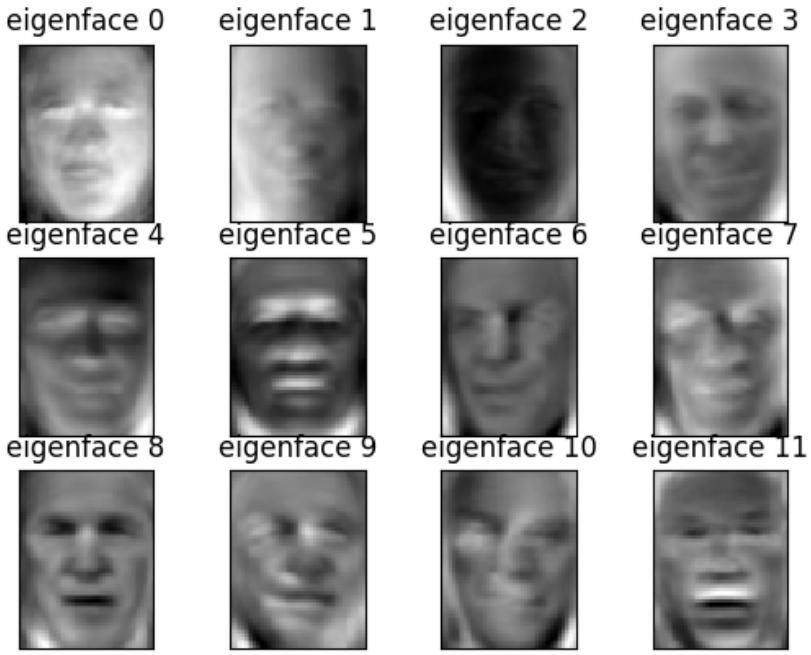


Figure 4.9: Eigenfaces of the LFW database using standard PCA.

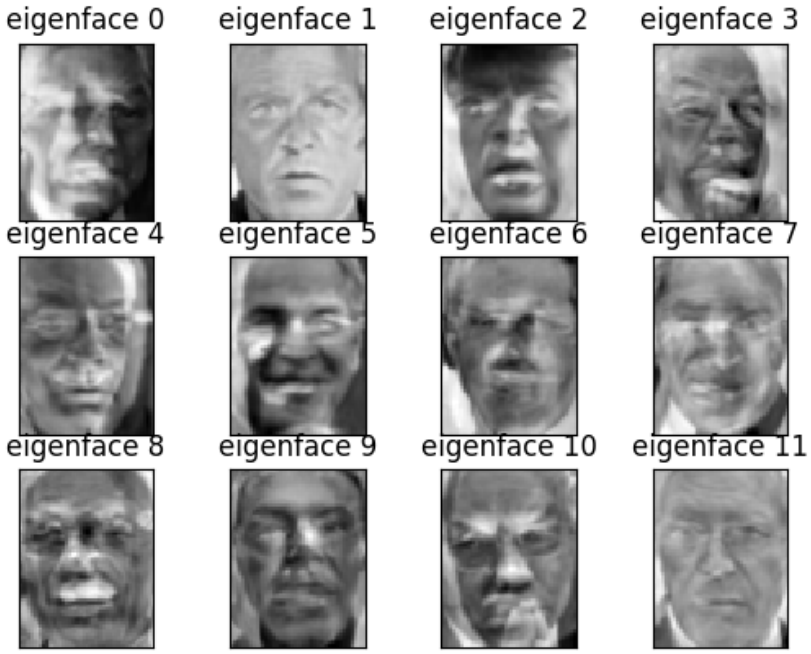


Figure 4.10: Eigenfaces of the LFW database using gaPCA.

Table 4.5: Number of iterations of the classifier with NO PCA, standard PCA and gaPCA

Dimensionlity reduction	Number of iterations
No PCA	48.4
PCA	19.2
gaPCA	25

Training the classifier with the original, raw data requires 48 iterations, compared to just 25 when applying the gaPCA method as a dimensionality reduction technique, and only 19 is the canonical PCA method is used.

4.4 Conclusions

This chapter presented the evaluation and validation of the gaPCA method in the field of face recognition, in addition to the Remote Sensing applications described in the previous chapters. For the experiments presented in this chapter, four datasets of faces openly available for research purposes were used: FEI, Yale, Cambridge and LFW. The face recognition task was performed for the first three datasets using the Eigenfaces approach: i.e. applying both gaPCA and canonical PCA, separately, on the training dataset to compute the eigenvectors and computing the similarity score for the test set images using a formula based on the inverse Euclidean distance. The accuracy of the recognition was computed for both methods and the comparative evaluation illustrated that gaPCA performed very similarly to its more established counterpart, scoring for some cases equally, and in other cases slightly lower (under 10%) than the canonical PCA.

In the case of the LFW dataset, face recognition was performed using a neural network classifier, trained in three scenarios: on the original, raw data, on the data obtained using the canonical PCA, and on the data resulted from applying gaPCA. The overall classification accuracy results placed gaPCA virtually tied to the canonical PCA, with just 2% below in terms of average precision, having even outperformed its counterpart for some classes. In addition, the effectiveness of the gaPCA as a dimensionality reduction technique was confirmed also quantitatively, by showing that applying gaPCA on the training set decreases substantially the number of training iterations required by the neural network classifier.

The research and experiments presented in this chapter have been validated and disseminated in the following publication:

- [49] A. L. Machidon, O. M. Machidon, and P. L. Ogrutan. Face Recognition Using Eigenfaces, Geometrical PCA Approximation and Neural Networks. In *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pages 80–83, 2019, indexed in the Web of Science (Proceedings paper) and in the IEEE Xplore Digital Library.

CHAPTER 5

Parallelization

The recent technological advances in both sensors and computer technology have increased exponentially the volume of remote sensing data repositories. Because a significant amount of this data is defined by a high degree of redundancy, it can be accurately reduced to a much smaller number of variables without any substantial loss of information. This can be accomplished using dimensionality reduction techniques [64], mathematical methods and algorithms that transform the high-dimensional data into "a meaningful representation of reduced dimensionality" [65]. Such techniques however present a high computational complexity and require considerable computing resources, an issue which can affect applications with strict timing constraint; hence high-performance computing architectures are the most likely candidates for a time- and resource-efficient implementation of such methods.

5.1 Parallel PCA methods

Due to the fact that projection pursuit methods execution time increases exponentially with the dimensionality of data, these algorithms tend to be computationally intensive. This motivated researches to find alternative approaches for pursuing interesting projections not only in terms of data structures, but also in terms of computational resources [31], [56]. Many efforts made by the scientific community in the past decade focused on developing parallel implementations of a particular case of Projection Pursuit method, which is PCA, in order to achieve increased performance fostered by parallel architectures such as multi-core CPUs or GPUs.

5.2 Original contributions

5.2.1 Parallelization of the gaPCA algorithm

This section presents the research efforts directed at developing parallel implementations of the gaPCA algorithm on various computing architectures and using several programming and scripting languages: C++, Matlab and Python for the multi-core CPU versions, and PyCUDA and CUDA running on NVIDIA GPUs. To asses how each implementation and computing platform impacts the performance with regard to execution time, a comparative time analysis was performed that shows the net superiority of the parallel implementations with regard to accelerating the computation and reducing the execution time.

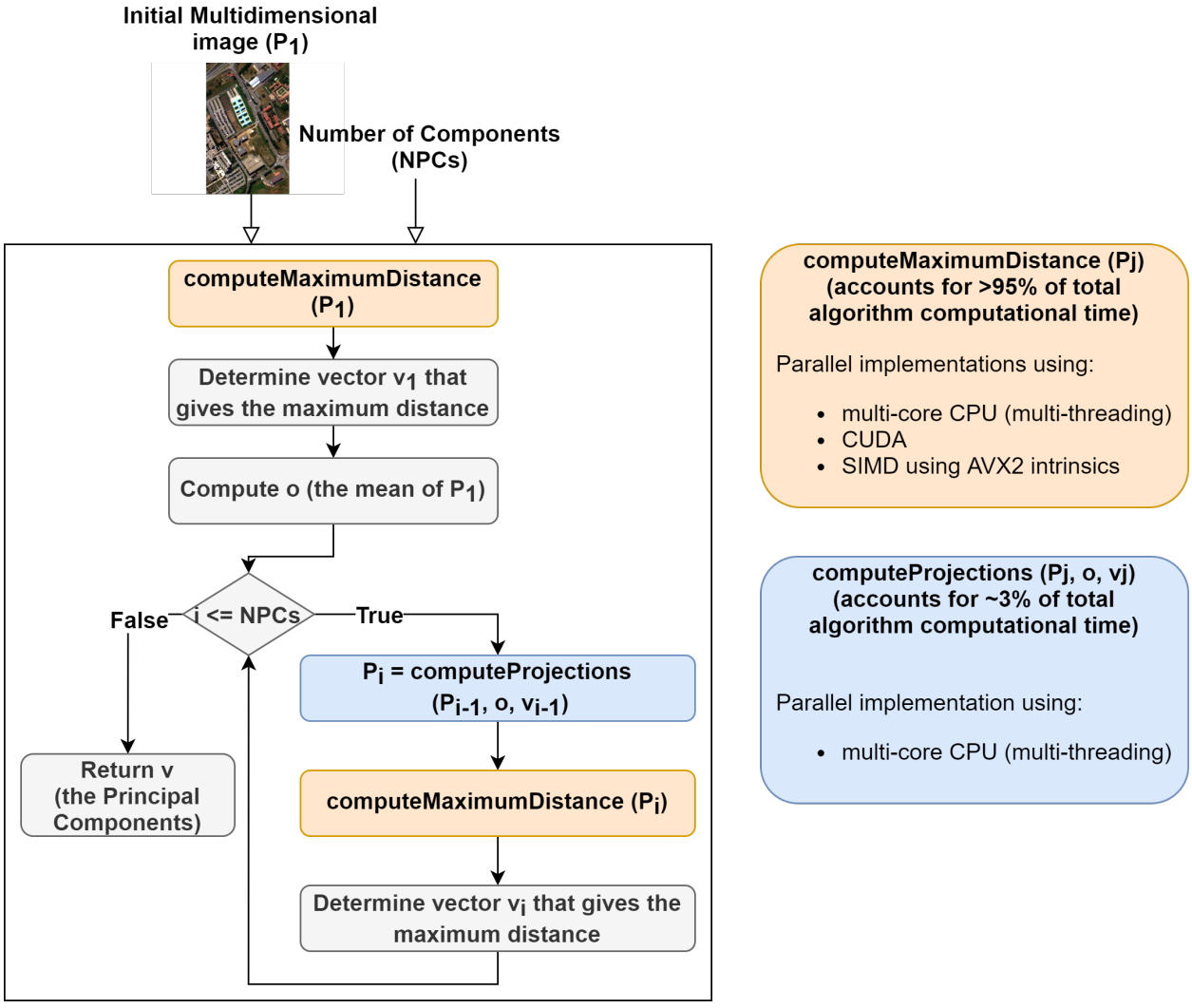


Figure 5.1: Diagram showing gaPCA algorithm design and the two sub-routines with parallel implementations.

The first step in elaborating the parallel implementations of the gaPCA algorithm was profiling the code of the algorithm (the sequential, initial implementation) for analyzing how time consuming each of the method's sub-routines are, and thus establish what parts of the

algorithm are most suitable for parallel implementations. The results of this code profiling stage are illustrated in a schematic in Figure 5.1, where the contribution of each sub-routine of the gaPCA method to the total execution time is highlighted. Considering that just the computation of the Euclidean distances (which is according to Figure **??** around 94.39%) can be the subject of parallel implementation, a maximum speedup of 20× can be expected, which of course scales with the number of processors used.

The research efforts focus on two directions: on one hand, efficiently elaborating the parallel implementations (SIMD, multi-threading and GPU-based) of the gaPCA method, on the other hand providing a relevant analysis on the performance of these implementations with regard to execution time and energy consumption by deploying them on several hardware platforms: NVIDIA Jetson nano development board, Intel Xeon W3670 CPU, NVIDIA GeForce GTX 1050 Ti GPU, AMD Ryzen 5 3600 CPU and NVIDIA GeForce GTX 1650.

5.2.2 Matlab, Python and PyCUDA implementations

The initial Matlab implementation (Listing 5.1) [44] was elaborated using the specific instructions from the Matlab Parallel Computing Toolbox. The experimental runs were done in Matlab R2019a, on the Linux Ubuntu 18.04 operating system.

```
function [i_extreme, j_extreme, dist_max] euclidDist(X)
{
[m, n] = size(X);
parfor i=1: m-1
[d(i),j(i)] = max(pdist2(X(i,:),X(i+1:m,:)));
end
[dist_max,ind] = max(d);
i_extreme = ind;
j_extreme = j(ind)+ind;
return
}
```

Listing 5.1: Matlab implementation for the Euclidean Distances function

The next two Python implementations employ a parallel execution of the gaPCA method on a multi-core CPU and a NVIDIA GPU, respectively. Both are elaborated using the Python Numba and Numpy libraries; the CPU multi-core parallel version uses the Numba JIT compiler, while the GPU CUDA version was elaborated using the PyCUDA Python library [11].

```
@jit(nopython=True, parallel=True, nogil=True)
def euclidDist(A):
dist = numpy.zeros(len(A))
index = numpy.zeros(len(A))
for i in prange(len(A)-1):
  temp_dist = numpy.zeros(len(A))
for j in prange(i+1,len(A)):
  temp_dist[j] = numpy.linalg.norm(A[i]-A[j])
dist[i] = numpy.amax(temp_dist)
index[i] = numpy.argmax(temp_dist)
return (numpy.amax(dist), numpy.argmax(dist), int(index[numpy.argmax(dist)]))
```

Listing 5.2: Python implementation for the Euclidean Distances function

5.2.3 CUDA implementation

In this work, the CUDA kernel was elaborated so that a block of threads calculates one element of the distance matrix, with each thread calculating the square distance between the corresponding elements of the two rows in the input matrix. To better highlight the functionality of the kernel, this is also graphically illustrated in Figure 5.2. In this graphical representation, the dataset a is the input matrix for which each row contains the pixels' values in every spectral band. One block of threads calculates the distance between rows i and j, with each thread tid_q computing $(a_{i,q} - a_{j,q})^2$. Once all threads in a block completed their tasks (passing a synchronization step), a parallel tree reduction mechanism is employed to compute the total distances between the rows i and j of the input dataset a by adding up all the squared values calculated by the threads.

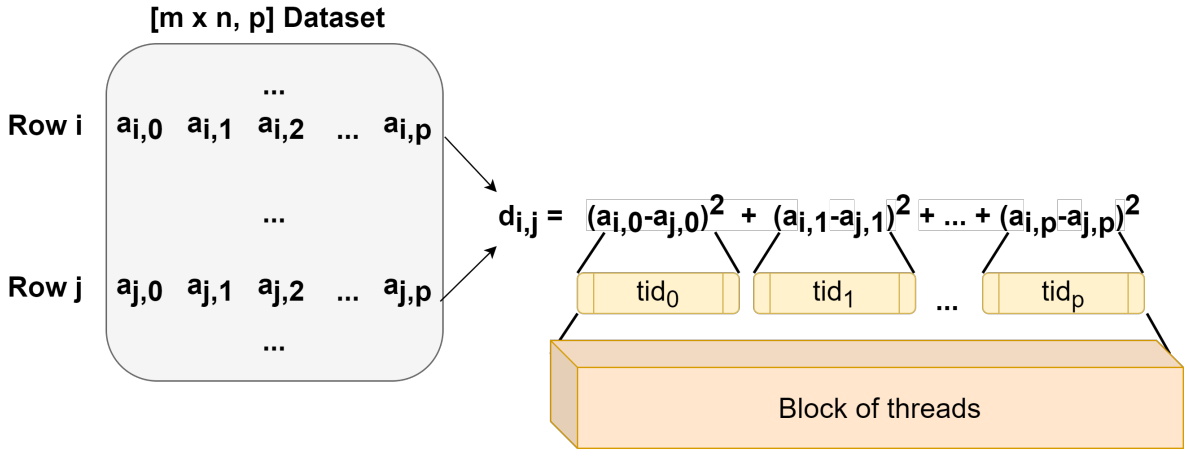


Figure 5.2: Diagram showing the parallel implementation in CUDA of the Euclidean Distance function.

Listing 5.3 shows the pseudocode for the CUDA kernel (the *euclidean* function). This function takes as input matrix X and returns C, a three element array containing the maximum distance (C[0]) and the indexes of the corresponding two furthest points from matrix X (C[1] and C[2]).

```
euclidean kernel(input X, output C):
elem1 = X[blockIdx.x, threadIdx.y];
elem2 = X[blockIdx.y, threadIdx.y];
result = (elem1-elem2)*(elem1-elem2);
accum[threadIdx.y] = result;
Synchronize threads
Perform parallel tree-reduction and compute dist
if (dist>C[0])
C[0] = dist;
update indexes C[1] and C[2];
endif
```

Listing 5.3: Pseudocode for the CUDA kernel computing pairwise Euclidean distance between the rows of the input matrix X

A shared memory vector entitled *accumResult* was added to the CUDA kernel; each thread will compute its corresponding squared difference and store it this vector in the location

determined by the thread's index. The successful completion of this stage by all threads is marked by a synchronization checkpoint (ensured by calling the *__syncthreads*(); method); next the parallel tree reduction is performed (as shown in Listing 5.4) which ultimately gives the final distance value. After this final value is computed, one thread from each block (thread with $threadIdx.y = 0$ was chosen, for uniformity) performs the comparison of this value with the previous computed maximum distance, and if it is the case updates the stored maximum distance accordingly, together with its stored indexes.

```
__syncthreads();
// Parallel tree-reduction
for (int stride = SIZE/2 ; stride > 0 ; stride >>= 1) {
if (ty < stride)
accumResult[tx*SIZE+ty] += accumResult[stride + tx*SIZE+ty];
__syncthreads();
}
```

Listing 5.4: CUDA kernel code for parallel tree-reduction

5.2.4 C++ implementations

The first two C++ versions of the gaPCA algorithm are a basic single-core one and a multi-threading version based on OpenMP, a "simple C/C++/Fortran compiler extension, which allows adding multithreading parallelism into existing source code" [9]. The multi-threading code is shown in Listing 5.5; the "#pragma omp parallel" compiler directive defines the code section designated for parallel execution, this directive causes the task allocation to threads ahead of the section execution; more specifically, this directive allocates the computing tasks to the existing active threads, so it does not "create" the thread pool. Each of the threads launched simultaneously into execution computes a distance between two rows of the input matrix (two pixels with their corresponding values in all spectral bands).

```
void parallelDist(short **X, int n, int m, int& index1, int& index2, long long
    d)
{
long long dist[n] = { 0 };
int index[n] = { 0 };
#pragma omp parallel num_threads(12)
{
#pragma omp for
for (int i =0; i<n-1; i++)
{
long long temp_dist[n] = {0};
for (int j =i+1; j<n; j++)
{
temp_dist[j] = squarediff(m,X[i],X[j]);
}
dist[i] = *max_element(temp_dist,temp_dist+n);
index[i] = distance(temp_dist, max_element(temp_dist,temp_dist+n));
}
}
d = *max_element(dist, dist+n-1);
index1 = distance(dist, max_element(dist,dist+n-1));
```

```
index2 = index[index1];
}
```

Listing 5.5: Source code for the C++ multi-core function computing pairwise Euclidean distances between the rows of the input matrix X

The third C++ implementation was elaborated by extending the multi-threading OpenMP version with help of the SIMD instruction set. The goal was to take advantage of the Data Level Parallelism (DLP) using the SIMD instructions which can work on wide vector registers (ranging from 64 to 512 bits, depending on the SIMD instruction set extension employed: MultiMedia eXtensions (MMX) [55], Streaming SIMD Extensions (SSE) [59], Advanced Vector eXtensions (AVX) [41]). Using this approach, the highest efficiency (towards 100%) can be achieved by ensuring that the input array has a size multiple of the vector register's width (4, 8, 16, or 32 16-bit data elements). The SIMD experiments described in this work were performed on a AMD Ryzen 5 3600 CPU [1] which supports the AVX2 [2] SIMD instruction set extension.

```
alignas(__m256i) short **arr = (short **)malloc(ROWS * sizeof(short *));
std::size_t sz = COLS;
for (i=0; i<ROWS; i++)
arr[i] = static_cast<short*>(aligned_alloc(32,sz*2));
```

Listing 5.6: Source code for the C++ alignment of a two-dimensional matrix of short

```

long long squarediff_avx(int size, short *p1, short *p2)
{
std::size_t sz = size;
long long s = 0;
int i = 0;
for (; i + 16 <= size; i+=16 )
{
// load 256-bit chunks of each array
__m256i first_values = _mm256_load_si256((__m256i*)  &p1[i]);

__m256i second_values = _mm256_load_si256((__m256i*) &p2[i]);

//substract each pair of 16-bit integers in the 256-bit chunks
__m256i substracted_values = _mm256_sub_epi16(first_values, second_values);

// multiply each pair of 16-bit integers in the 256-bit chunks
__m256i multiplied_values_lo = _mm256_mullo_epi16(substracted_values,
    substracted_values);
__m256i multiplied_values_hi = _mm256_mulhi_epi16(substracted_values,
    substracted_values);

s += sum_avx(multiplied_values_lo, multiplied_values_hi);
}

for (; i < size; i++)
{
s+= pow(p1[i] - p2[i],2);
}
```

```
return s;
}
```

Listing 5.7: Source code for the C++ SIMD function computing pairwise Euclidean distances

The sum of the squared differences was calculated with another C++ function based on SIMD instructions, displayed in Listing 5.8.

```
long long sum_avx(__m256i a_part1, __m256i a_part2)
{
short extracted_partial_sums1[16] = {0};
short extracted_partial_sums2[16] = {0};
_mm256_storeu_si256((__m256i*) &extracted_partial_sums1, a_part1);
_mm256_storeu_si256((__m256i*) &extracted_partial_sums2, a_part2);
long long sssum=0;
for(int i=0;i<16;i++) {
int temp = ((extracted_partial_sums2[i]<<16) | ((extracted_partial_sums1[i]) &
    0xffff));
sssum+=temp;
}
return sssum;
}
```

Listing 5.8: Source code for the C++ multi-core function computing the sum of two 256-bit registers

5.3 Results and discussion

To evaluate the gaPCA algorithm's acceleration, we build a benchmark in which we measured the amount of time the algorithm took to execute on every testcase of the two datasets mentioned above, for 1, 3 and 5 principal components. For each of the above mentioned image sizes, we ran the algorithm with the specified number of computed principal components, several times for each test (in order to minimize the overhead time associated with the cache warming [43]) and averaged the result.

5.3.1 Python vs. PyCUDA

Jetson Nano The first comparative assessment between the Python CPU multi-core and GPU CUDA implementations was performed on the Jetson Nano platform.

The experimental runs were conducted on the first five image crops of the Pavia University dataset, from 20×20 to 125×125 and all the spectral bands (103) and a number of 6 fixed computed principal components. The algorithm ran 10 times for each image dimension. The results are shown in Table 5.1 .

Table 5.1: Timing analysis of gaPCA for varying image crop size of the Pavia University dataset on the Jetson Nano platform.

Image dimensions	CPU (seconds)	GPU (seconds)	Speedup
20x20	1.15	1.98	0.58×
40x40	15.40	5.26	2.93×
80x80	229.40	45.88	5×
100x100	556.94	104.45	5.33×
125x125	1347.41	235.59	5.72×

The results show that, as expected, the CUDA implementation achieves an average speed-up of up to 5.72× faster than the CPU version for the biggest crop sizes (80×80, 100×100 and 125×125). The speed-up results obtained are also charted in Figure 5.3. It can be noticed that for the smaller crop sizes the speed-up is lower (for 40x40 just below 3×) or the CUDA version is even slower than the CPU for the smallest crop size (20×20, 0.58× speed-up). This is due to the CUDA computational overhead (i.e. copying the data from CPU memory to GPU memory and reverse). The impact of the overhead is reduced as the dataset size increases.

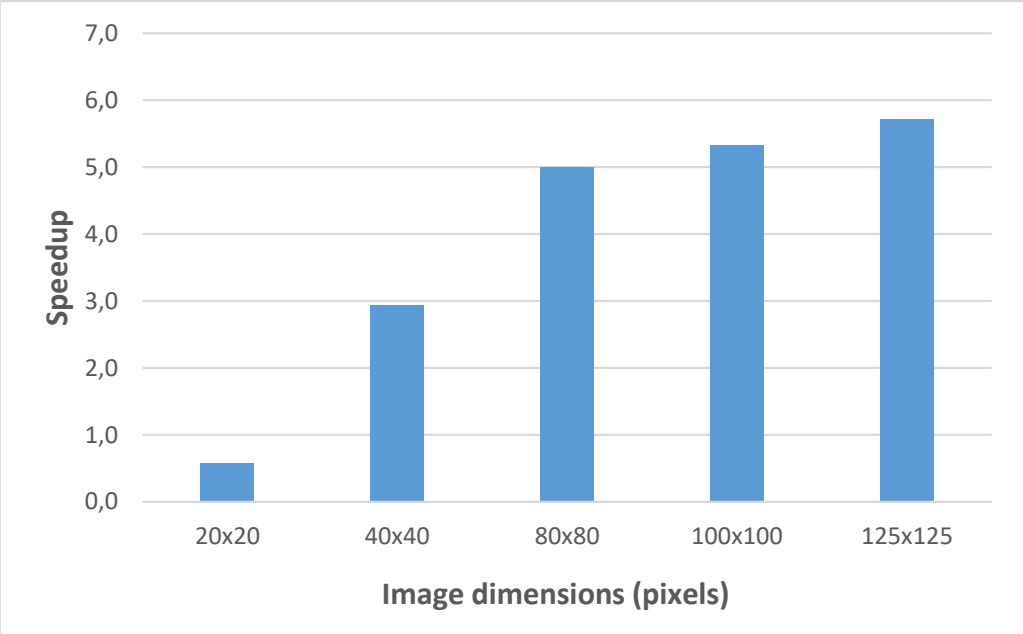


Figure 5.3: Speedup between GPU and CPU gaPCA implementations on Jetson Nano.

Intel Xeon W3670 - GTX 1050Ti The second comparative assessment was performed between the Python CPU multi-core and the GPU CUDA implementation on the Intel Xeon W3670 - GTX 1050Ti platform.

The experimental runs were conducted on all image crops, of both Indian Pines (Table 5.2) and Pavia University (Table 5.4) dataset for a number of 1, 3 and 5 computed principal components.

Table 5.2: gaPCA GPU vs. CPU execution times (seconds) on Indian Pines for different image sizes and number of principal components on the Intel Xeon W3670 - GTX 1050Ti platform.

Image dimensions	GPU 1 PC	CPU 1 PC	GPU 3 PC	CPU 3 PC	GPU 5 PC	CPU 5 PC
20×20	0.45	0.68	0.45	1.18	0.72	1.28
40×40	0.57	1.45	1.01	3.18	1.68	4.55
80×80	2.68	14.61	7.40	33.97	12.02	52.68
100×100	5.46	33.84	15.94	80.12	25.66	126.00
145×145	20.45	154.37	61.25	353.37	102.20	561.91

Table 5.3: GPU vs. CPU speedup table for computing 1, 3 and 5 principal components for the Indian Pines dataset on the Intel Xeon W3670 - GTX 1050Ti platform.

Image dimensions	1 PC	3 PC	5 PC
20×20	1.51×	1.65×	1.80×
40×40	2.56×	3.15×	2.71×
80×80	5.46×	4.59×	4.38×
100×100	6.20×	5.03×	4.91×
145×145	7.55×	5.77×	5.50×

The results show that, once again, the CUDA implementation outperforms the Python CPU one, with an average speedup of up to 7.55× faster than the CPU version. For the bigger crop sizes of the Indian Pines dataset, the speedup is the most significant.

The speed-up results obtained are also charted in Figure 5.4.

Table 5.4 and Table 5.5 show the execution times and speedups for the CPU and GPU implementations for the Pavia University dataset, on the Intel Xeon W3670 - GTX 1050Ti platform. The speedup recorded is up to 11.78× for the GPU version of the algorithm as compared to the CPU one. The speed gain is the most significant for the bigger crop sizes of the Pavia University dataset. The reason is that the amount of floating point operations resulting for small data are not enough to keep the GPU busy, but rather the memory transfer to and from the GPU becomes the bottleneck and increases the total run time. As the image sizes grows, parallelization on the GPU becomes more and more effective in terms of execution times.

The speedup results obtained for the Pavia University dataset, on the Intel Xeon W3670 - GTX 1050Ti platform are also charted in Figure 5.5.

5.3.2 Matlab vs. Python and PyCUDA

Table 5.6 shows the timing results for the three implementations, for each test case of the Indian Pines dataset. Similarly, Table 5.7 shows the timing results for the test cases of the Pavia University dataset. Figures 5.6 and 5.7 present the speedup between the three implementations

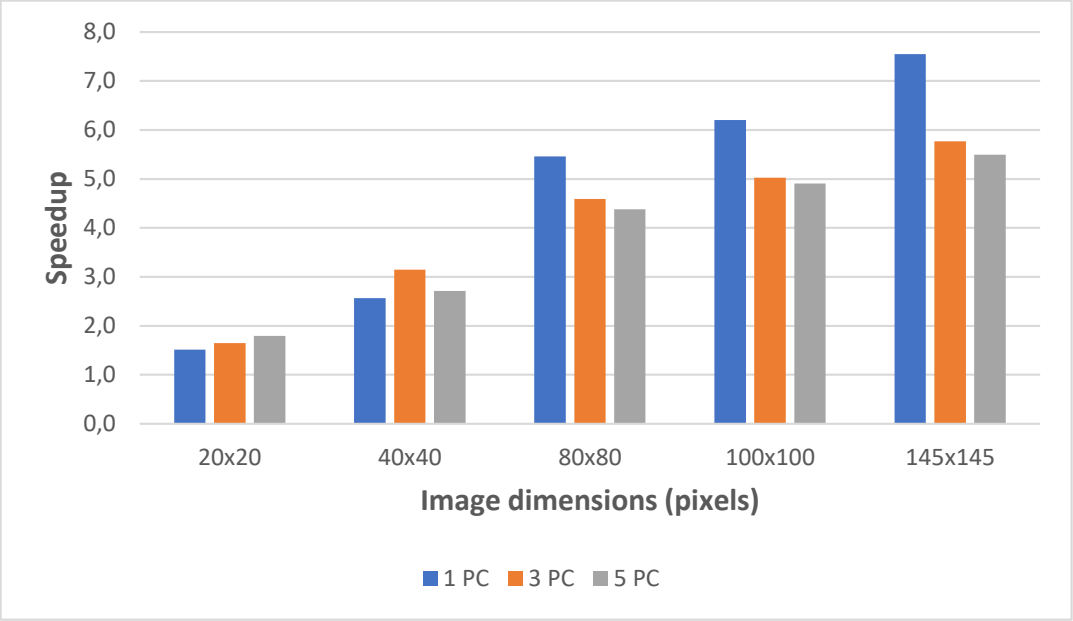


Figure 5.4: CPU vs. GPU speedup plot for computing 1, 3 and 5 principal components for the Indian Pines dataset on the Intel Xeon W3670 - GTX 1050Ti platform.

Table 5.4: gaPCA GPU vs. CPU execution times on Pavia University for different image sizes and number of principal components on the Intel Xeon W3670 - GTX 1050Ti platform.

Image dimensions	GPU 1 PC	CPU 1 PC	GPU 3 PC	CPU 3 PC	GPU 5 PC	CPU 5 PC
20x20	0.23	0.67	0.40	1.14	0.64	1.25
40x40	0.25	1.18	0.68	2.72	1.12	3.80
80x80	1.06	8.70	3.20	26.15	5.35	42.04
100x100	2.16	21.41	6.72	62.27	11.23	102.05
200x200	29.44	343.98	90.17	984.66	150.96	1611.97
250x250	71.72	812.26	217.11	2373.87	362.32	3921.91
300x300	155.01	1719.58	444.90	4936.99	742,93	8179.16
400x340	335.10	3947.68	1007.32	11338.15	1679.32	18633.12
610x340	774.35	9035.44	2336.16	26188.72	3890.46	43145.17

Table 5.5: CPU vs. GPU speedup table for computing 1, 3 and 5 principal components for the Pavia University dataset on the Intel Xeon W3670 - GTX 1050Ti platform.

Image dimensions	1 PC	3 PC	5 PC
20x20	2.93×	1.80×	1.96×
40x40	4.64×	3.99×	3.38×
80x80	8.19×	8.17×	7.86×
100x100	9.90×	9.27×	9.09×
200x200	11.68×	10.92×	10.68×
250x250	11.33×	10.93×	10.82×
300x300	11.09×	11.10×	11.01×
400x340	11.78×	11.26×	11.10×
610x340	11.67×	11.21×	11.09×

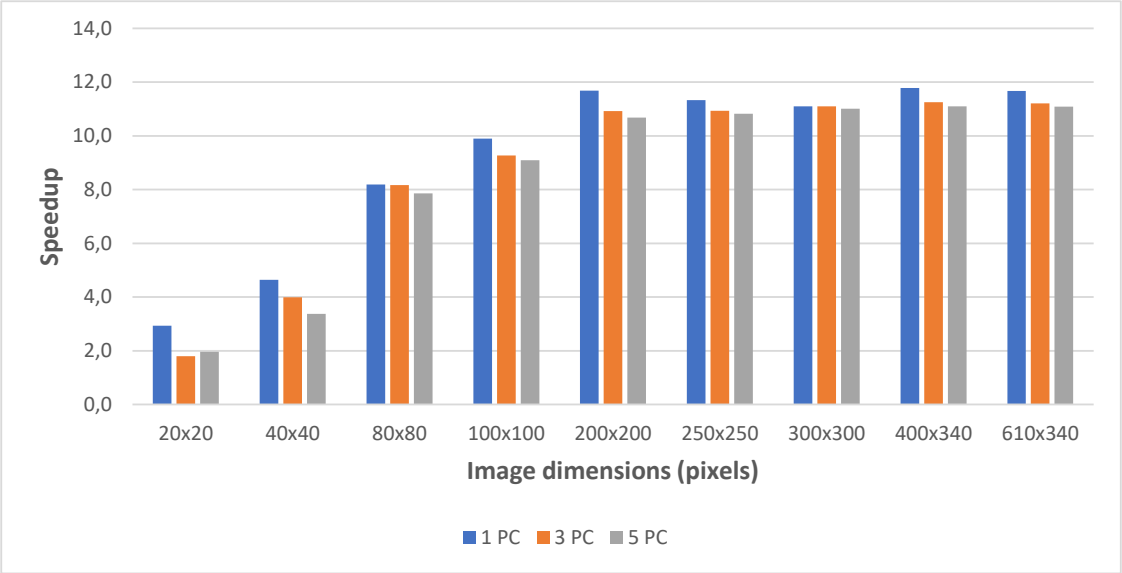


Figure 5.5: CPU vs. GPU speedup plot for computing 1, 3 and 5 principal components for the Pavia University dataset on the Intel Xeon W3670 - GTX 1050Ti platform.

(with the Matlab implementation taken as a baseline for comparison) for all the test cases of the Indian Pines and the Pavia University, respectively.

For the first test case (Indian Pines), the Matlab implementation outperforms its Python equivalent, being with up to 4.58× for the 40×40 image crops, while for the other image sizes (80×80, 100×100 and 145×145) the Python implementation is faster. For the second test case (Pavia University), we can notice that the Matlab implementation is faster than the Python one with approximately 20% on average (ranging from 4% for the 200× 200 and 5 PCs dataset to 42% for the 100× 100 and 1 PC dataset).

Crop size	No. of PCs	Matlab	Python	PyCUDA
40x40	1	0.275	1.260	0.153
40x40	3	0.832	2.802	0.449
40x40	5	1.448	3.519	0.756
80x80	1	8.326	6.797	0.769
80x80	3	24.678	18.265	2.319
80x80	5	40.990	29.333	3.884
100x100	1	22.090	14.531	1.592
100x100	3	66.377	41.929	4.843
100x100	5	110.449	68.647	8.004
145x145	1	104.134	64.498	5.843
145x145	3	313.070	181.152	18.036
145x145	5	521.057	298.212	30.137

Table 5.6: Indian Pines Matlab vs.Python vs. PyCUDA execution times (s).

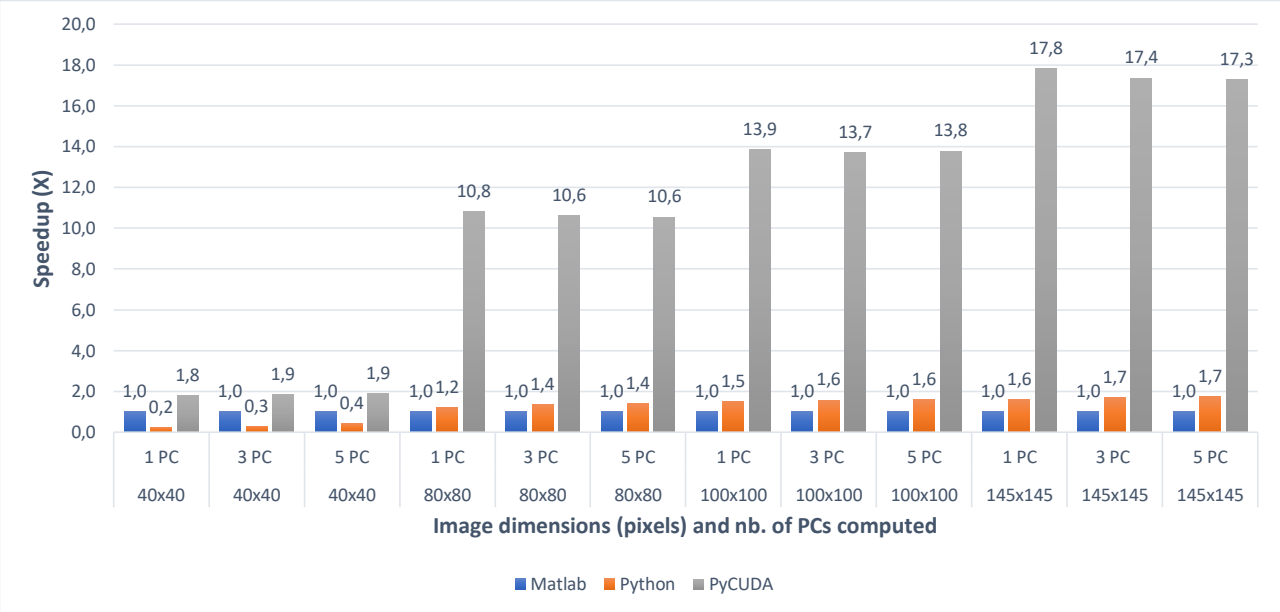


Figure 5.6: Indian Pines Matlab vs.Python vs. PyCUDA speedup for 1 PC (a) 3 PCs (b) and 5 PC (c) for various image dimensions.

Crop size	No. of PCs	Matlab	Python	PyCUDA
100x100	1	9.507	13.476	0.866
100x100	3	28.439	39.126	2.745
100x100	5	47.580	64.131	4.497
200x200	1	195.801	204.855	10.391
200x200	3	575.083	601.477	31.884
200x200	5	957.494	992.193	53.496
300x300	1	883.342	1027.397	50.905
300x300	3	2653.649	3036.512	155.068
300x300	5	4432.107	5030.831	260.203
610x340	1	4501.181	5453.752	267.402
610x340	3	13588.632	16035.242	806.866
610x340	5	22675.191	26702.160	1347.312

Table 5.7: Pavia University Matlab vs.Python vs. PyCUDA execution times (s).

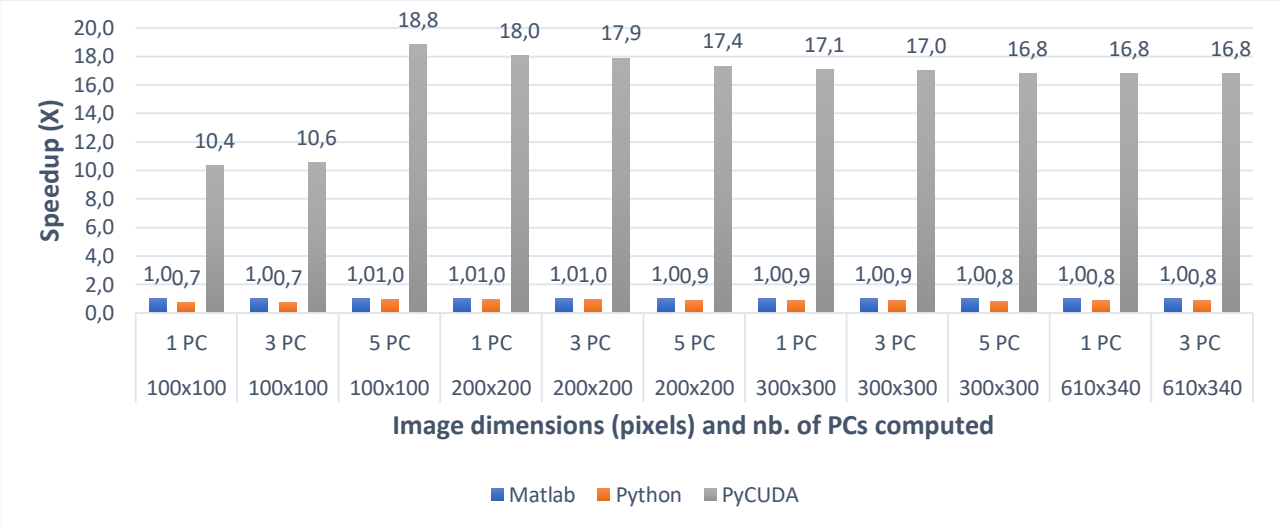


Figure 5.7: Pavia University Matlab vs.Python vs. PyCUDA speedup for 1 PC (a) 3 PCs (b) and 5 PC (c) for various image dimensions.

5.3.3 C++ single core vs. multicore

The second comparative evaluation was performed between the execution times of several C++ implementations: single core (SC), single core with AVX2 intrinsics (SC AVX2), multi-core (MC) and multi-core using AVX2 intrinsics (MC AVX2). Table 5.8 shows the timing results for the four implementations, for each test case of the Indian Pines dataset. Similarly, Table 5.9 shows the timing results for the test cases of the Pavia University dataset.

The speedups between the four implementations (with the single core implementation taken as a baseline for comparison) are shown in Figures 5.8 and 5.9, for all the test cases of the

Indian Pines and the Pavia University, respectively.

Crop size	No. of PCs	C++ SC	C++ SC AVX2	C++ MC	C++ MC AVX2
40x40	1	0.947	0.206	0.169	0.047
40x40	3	2.858	0.620	0.480	0.120
40x40	5	4.749	1.035	0.796	0.211
80x80	1	15.147	3.317	2.502	0.663
80x80	3	45.274	9.942	7.534	1.674
80x80	5	75.451	16.389	12.457	3.012
100x100	1	36.834	8.070	6.108	1.605
100x100	3	110.570	23.834	18.452	4.560
100x100	5	184.189	40.409	30.627	7.871
145x145	1	162.853	35.185	27.017	6.797
145x145	3	491.379	105.084	81.027	18.816
145x145	5	814.208	175.127	135.510	32.400

Table 5.8: Indian Pines C++ Single Core (SC) vs. Single Core AVX2 (SC AVX2) vs. Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) execution times (s).

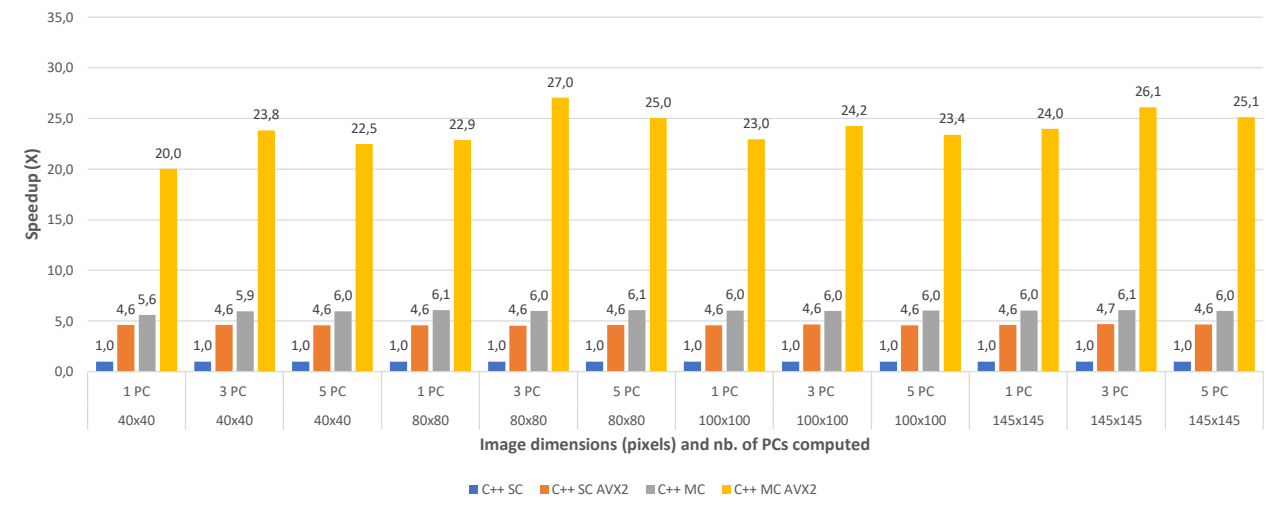


Figure 5.8: Indian Pines C++ Single Core (SC) vs. Single Core AVX2 (SC AVX2) vs. Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) speedup for 1 PC (a) 3 PCs (b) and 5 PC (c) for various image dimensions.

5.3.4 C++ multi core vs. CUDA

The final comparative evaluation was performed between the execution times of the three parallel C++ implementations: multi-core (MC), multi-core using AVX2 intrinsics (MC AVX2) and multi-core with CUDA programming (MC CUDA). Table 5.10 shows the timing results for the three implementations, for each test case of the Indian Pines dataset. Similarly, Table 5.11 shows the timing results for the test cases of the Pavia University dataset.

Crop size	No. of PCs	C++ SC	C++ SC AVX2	C++ MC	C++ MC AVX2
100x100	1	18.373	4.63084	3.030	0.805
100x100	3	55.185	13.9219	9.025	2.814
100x100	5	91.666	23.1688	15.277	4.565
200x200	1	293.311	74.8406	48.650	12.652
200x200	3	880.324	222.279	144.703	40.199
200x200	5	1472.080	371.005	243.616	68.642
300x300	1	1488.370	387.629	247.666	67.894
300x300	3	4489.640	1165.61	741.890	211.110
300x300	5	7438.200	1933.44	1240.140	363.606
610x340	1	7956.360	2053.12	1322.530	393.734
610x340	3	23962.794	6202.35	3975.840	1144.730
610x340	5	40190.385	10408.8	6605.060	1905.980

Table 5.9: Pavia University C++ Single Core (SC) vs. Single Core AVX2 (SC AVX2) vs. Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) execution times (s).

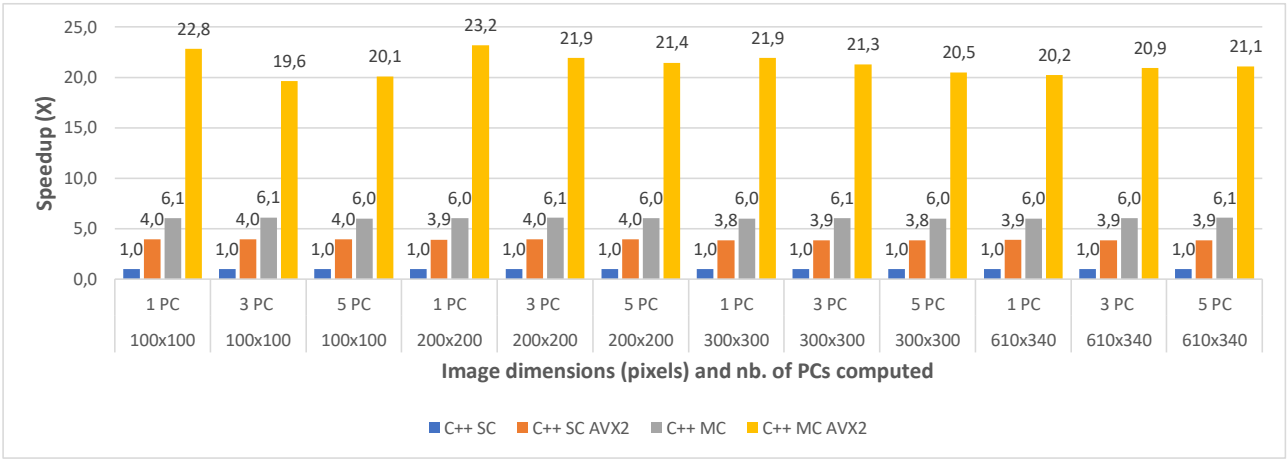


Figure 5.9: Pavia University C++ Single Core (SC) vs. Single Core AVX2 (SC AVX2) vs. Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) speedup for 1 PC (a) 3 PCs (b) and 5 PC (c) for various image dimensions.

The speedups between the three implementations (with the standard multi-core implementation taken as a baseline for comparison) are shown in Figures 5.10 and 5.11, for all the test cases of the Indian Pines and the Pavia University, respectively.

Crop size	No. of PCs	C++ MC	C++ MC AVX2	C++ MC CUDA
40x40	1	0.169	0.047	0.113
40x40	3	0.480	0.120	0.202
40x40	5	0.796	0.211	0.239
80x80	1	2.502	0.663	0.585
80x80	3	7.534	1.674	1.619
80x80	5	12.457	3.012	2.654
100x100	1	6.108	1.605	1.324
100x100	3	18.452	4.560	3.835
100x100	5	30.627	7.871	6.343
145x145	1	27.017	6.797	5.609
145x145	3	81.027	18.816	16.690
145x145	5	135.510	32.400	27.770

Table 5.10: Indian Pines C++ Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) vs. Multi Core CUDA (MC CUDA) execution times (s).

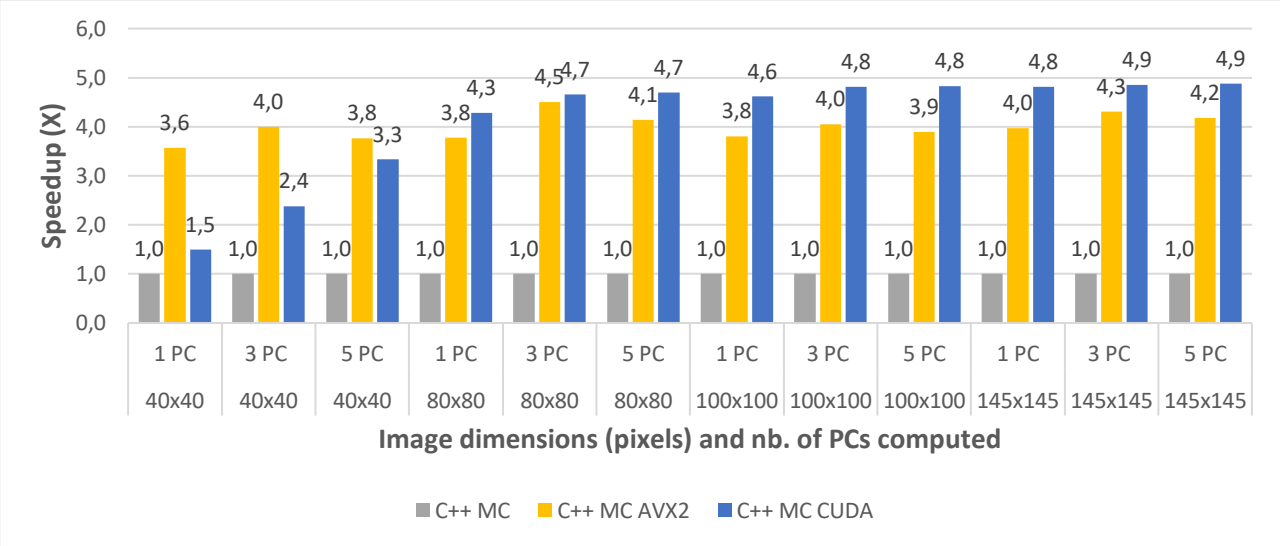


Figure 5.10: Indian Pines C++ Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) vs. Multi Core CUDA (MC CUDA) speedup for 1 PC (a) 3 PCs (b) and 5 PC (c) for various image dimensions.

Figure 5.12 shows a comparison of all implementations in terms of speed-up compared to the C++ single-core version taken as baseline. This comparative perspective confirms that the two CUDA-based implementations (C++ MC CUDA and PyCUDA) yield the highest speed-ups, followed closely by the C++ MC AVX2 version. The C++ MC CUDA is faster than the PyCUDA version on average with 9.3% for Pavia and with aprrox. 30% for Indian Pines.

Crop size	No. of PCs	C++ MC	C++ MC AVX2	C++ MC CUDA
100x100	1	3.030	0.805	0.689
100x100	3	9.025	2.814	1.916
100x100	5	15.277	4.565	3.143
200x200	1	48.650	12.652	9.989
200x200	3	144.703	40.199	29.699
200x200	5	243.616	68.642	49.068
300x300	1	247.666	67.894	50.700
300x300	3	741.890	211.110	151.082
300x300	5	1240.140	363.606	251.730
610x340	1	1322.530	393.734	267.495
610x340	3	3975.840	1144.730	801.950
610x340	5	6605.060	1905.980	1336.010

Table 5.11: Pavia University C++ Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) vs. Multi Core CUDA (MC CUDA) execution times (s).

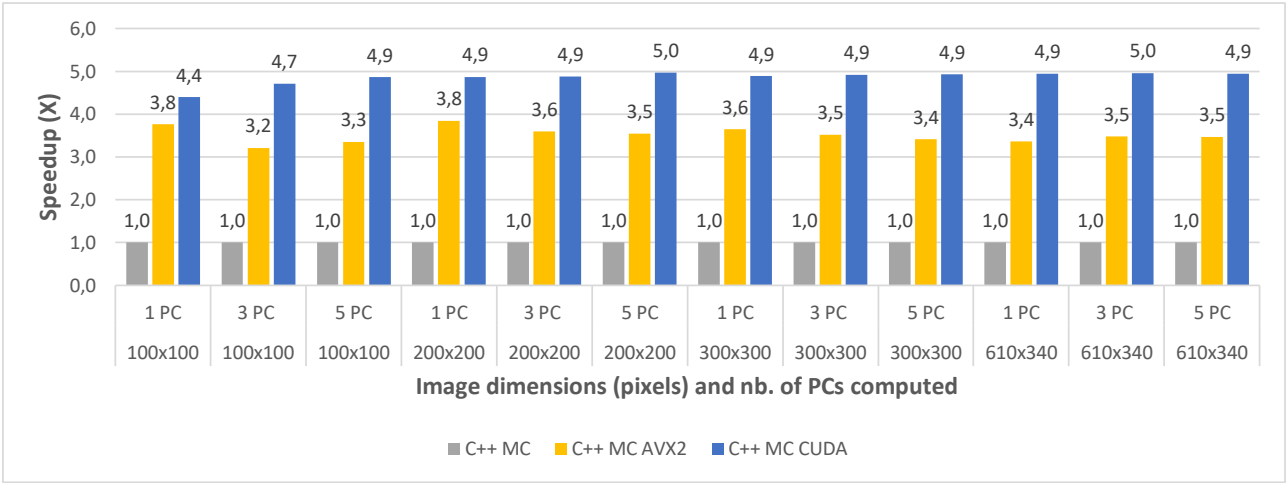


Figure 5.11: Pavia University C++ Multi Core (MC) vs. Multi Core AVX2 (MC AVX2) vs. Multi Core CUDA (MC CUDA) speedup for 1 PC (a) 3 PCs (b) and 5 PC (c) for various image dimensions.

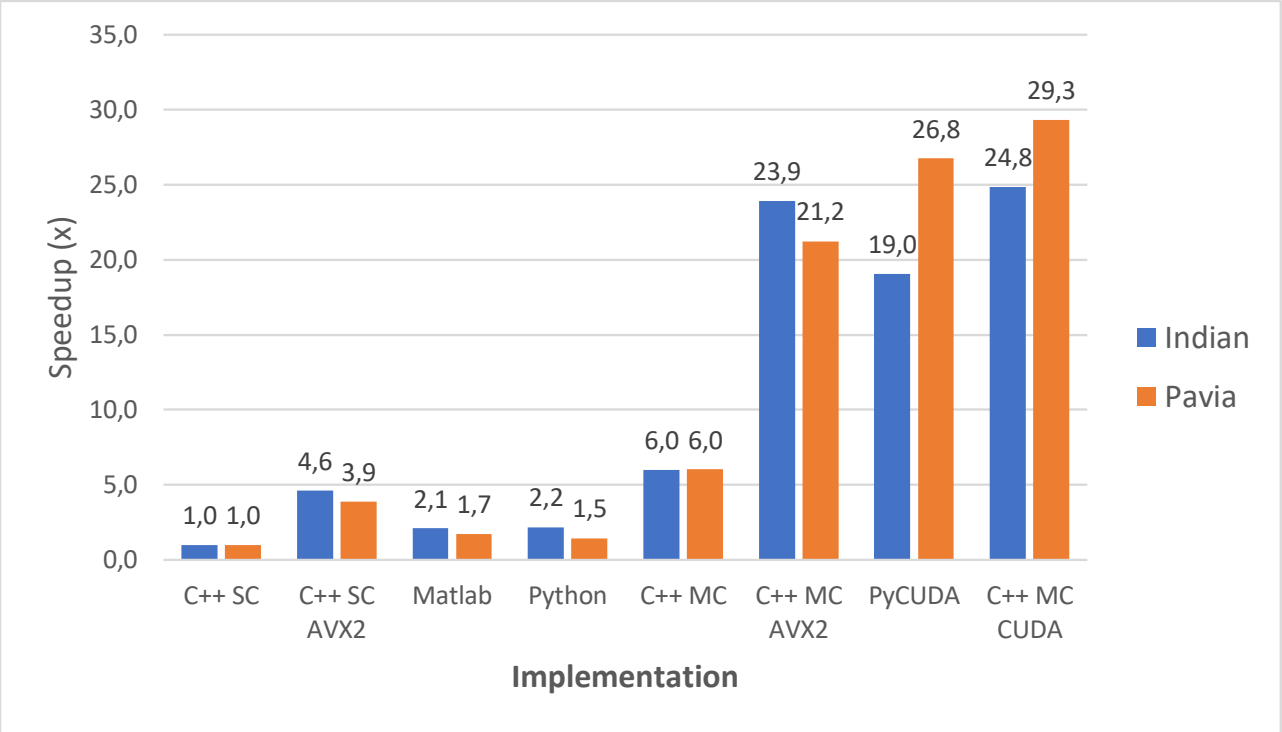


Figure 5.12: gaPCA overall speedup comparison

5.3.5 Energy efficiency

For evaluating the energy consumption we measured the system power consumption during the execution of each algorithm implementation for two test-cases: Indian Pines 100×100 and Pavia University 200×200; in all cases 5 PCs were computed. The system's power consumption was measured using a PeakTech 1660 Digital Power Clamp Meter [10], which provides a USB connection and dedicated software for data acquisition, thus increasing the accuracy and reliability of the results.

Table 5.12 shows the energy consumption for the Matlab, Python and PyCUDA implementations versus the total execution time; the energy results are also displayed in Figure 5.13. Similarly, the energy consumption for the C++ implementations are illustrated in Table 5.13 and Figure 5.14.

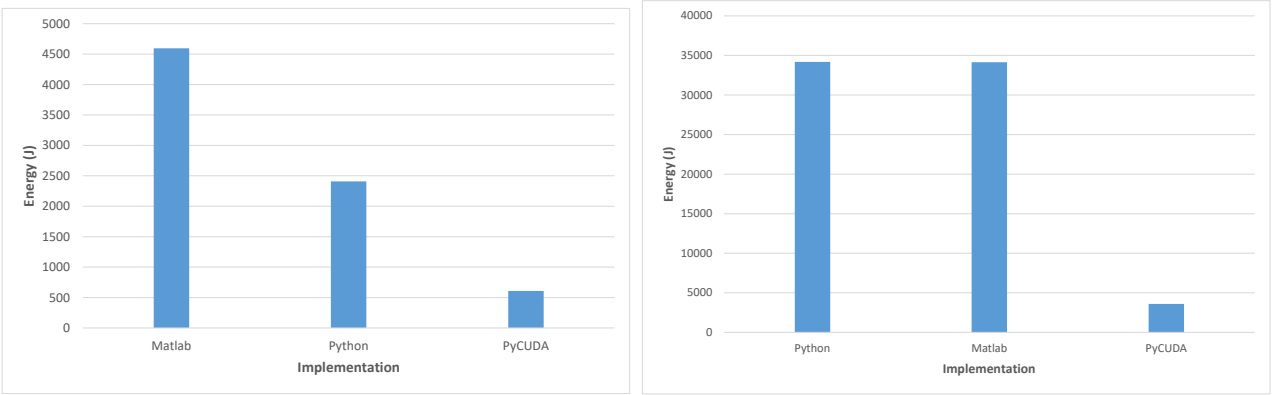
C++ MC CUDA implementation is shown to be slightly less energy efficient than the C++ MC AVX2 implementation (consuming 24.74% more energy in the Indian Pines test-case and 4.89% more energy in the Pavia University test-case) but it outperforms the C++ MC AVX2 implementation in terms of execution speed (being 24.07% faster for the Indian Pines test-case and 39.89% faster for Pavia University). The difference in the increased energy consumption for the two test-cases (24.74% Indian Pines vs. 4.89% Pavia University) can be explained by the difference in the number of spectral bands (200 for Indian Pines vs. 103 for Pavia University), which leads to the number of threads per block being used by the CUDA kernel (256 for Indian Pines vs. 128 for Pavia University). These results are confirmed also in the case of the PyCUDA implementation, which uses the same CUDA kernel.

Dataset	Size	No. of PCs	Implementation	Energy (J)	Time (s)
Indian	100x100	5	Matlab	4595.29	110.449
			Python	2409.06	68.647
			PyCUDA	609.23	8.004
Pavia U	200x200	5	Matlab	34139.77	957.494
			Python	34192.04	992.193
			PyCUDA	3589.8	53.496

Table 5.12: Energy measurement results for the Matlab, Python and PyCUDA implementations.

Dataset	Size	No. of PCs	Implementation	Energy (J)	Time (s)
Indian	100x100	5	C++ SC	3672	184.189
			C++ MC	1108.75	30.627
			C++ SC AVX2	792	40.409
			C++ MC CUDA	471.43	6.343
			C++ MC AVX2	378	7.871
Pavia U	200x200	5	C++ SC	27512.87	1472.080
			C++ MC	9242.40	243.616
			C++ SC AVX2	7431.87	371.005
			C++ MC CUDA	3491.43	49.068
			C++ MC AVX2	3328.63	68.642

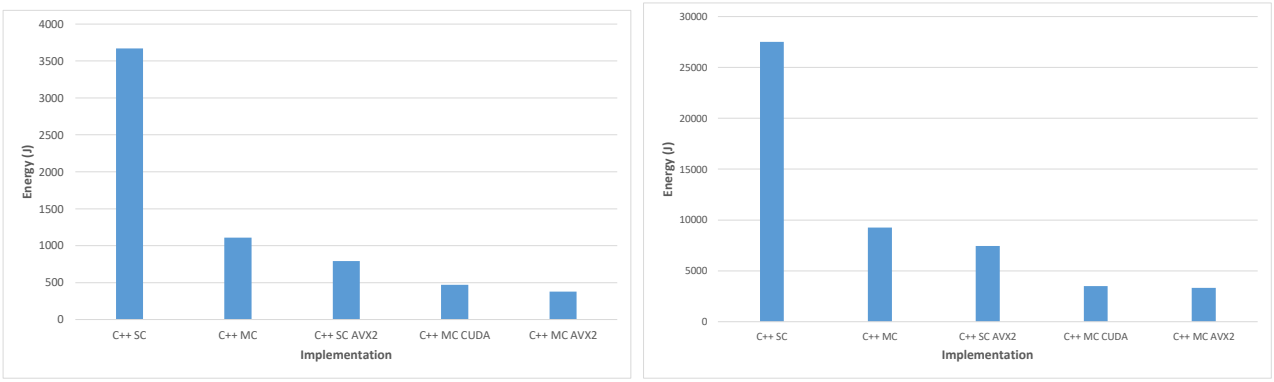
Table 5.13: Energy measurement results for the C++ implementations.



(a) Indian Pines 100×100 5 PCs

(b) Pavia University 200×200 5 PCs

Figure 5.13: Energy consumption for the Indian Pines (a) and Pavia University (b) datasets for the Matlab, Python and PyCUDA implementations.



(a) Indian Pines 100×100 5 PCs

(b) Pavia University 200×200 5 PCs

Figure 5.14: Energy consumption for the Indian Pines (a) and Pavia University (b) datasets for the C++ implementations.

5.4 Conclusions

The recent trends and advances in remote sensing and Earth observation in recent years have led to the continuous acquisition of massive geospatial data of various formats. This raises scientific challenges related to the processing, analysis, and visualization of remote sensing data, with focus on algorithms and computing paradigms able to extract knowledge and meaningful information both offline and in real time. Hence, the latest High Performance Computing devices and techniques, like parallel computing, GPUs and enhanced CPU instructions sets like AVX2 represent solutions able to provide significant gains in term of performance improvements of various data intensive applications and algorithms.

We have presented the implementation of a PP-based geometrical Principal Component Analysis approximation algorithm (gaPCA) for hyperspectral image analysis on multi-core Central Processing Units (CPU), Graphics Processing Units (GPU) and multi-core CPU using AVX2 intrinsics, together with a comparative evaluation of the implementations in terms of execution time and energy consumption. The experimental evaluation has shown that all parallel implementations have consistent speed-ups over the single core version: the C++ CUDA was on average 29.3× faster on Pavia and 24.8× faster on Indian Pines, while the C++ MC AVX2 version had an average speed-up of 21.2× for Pavia and 23.9× for Indian Pines compared to the baseline C++ SC version. These timing results show not only the benefits of using CUDA programming in implementing the gaPCA algorithm on a GPU in terms of performance and energy consumption, but also considerable advantages in implementing it on the multi-core CPU using AVX2 intrinsics. These two implementations were shown to be much faster than the standard multi-core implementation, and also the most efficient with regard to energy consumption. The C++ MC AVX2 version was shown to be the most efficient, requiring on average 8.26× less energy when running the Pavia dataset and 9.71× less energy when running the Indian Pines dataset compared to the baseline C++ SC implementation. The C++ MC CUDA version had just slighlty lower results, being 7.88× more efficient on Pavia and 7.78× more efficient on Indian Pines than the single-core implementation.

Consequently, this chapter highlights the benefits of using parallel computing, AVX2 in-

trinsics and CUDA parallel programming paradigms for accelerating dimensionality reduction algorithms like gaPCA in order to obtain significant speed-ups and improved energy efficiency over the traditional CPU single-core or multi-core implementations.

The research and experiments presented in this chapter have been validated and disseminated in the following publication:

- [44] A. L. Machidon, C. B. Ciobanu, O. M. Machidon, and P. L. Ogrutan. On Parallelizing Geometrical PCA Approximation. In *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pages 1–6. IEEE, 2019 also indexed in IEEE Xplore Digital Library
- [48] A. L. Machidon, O. M. Machidon, C. B. Ciobanu, and P. L. Ogrutan. Accelerating a Geometrical Approximated PCA Algorithm Using AVX2 and CUDA. *Remote Sensing*, 12(12):1918, 2020

CHAPTER 6

Final conclusions and original contributions

6.1 Final conclusions

This doctoral research was focused on designing, implementing, testing, validating and optimizing a novel Projection Pursuit method, namely the gaPCA method.

Following the research studies and experiments performed, presented in this thesis, the following conclusions can be drawn:

- The novel gaPCA method is characterized by several specific advantages compared to other similar methods in the Projection Pursuit family. gaPCA was confirmed to have an improved ability to discriminate smaller signals or objects from the background of the scene, and also given its algorithmic design, it is natively easily parallelizable which allows it to be accelerated on the latest High Performance Computing architectures. The most time-consuming subroutine in the gaPCA method is the Euclidean distance computation, which was shown to be seamlessly executed using parallel computing platforms [44].

- The validation of the novel gaPCA method using quality metrics on remote sensing data showed that when compared to the canonical PCA, both methods provide similar results with regard to the contrast and entropy scores, while in the case of the energy metric the gaPCA principal components are shown to have a superior image spatial quality, which could potentially lead to better classification results.

- With regard to land classification accuracy, the gaPCA method had, on average, higher results than the canonical PCA. gaPCA clearly outperformed its more well-established counterpart for the preponderantly spectral classes, small objects or classes; in these situations, the canonical PCA disregards the information with small contributions to the overall signal variance, labeling it as redundant or "unimportant", which diminishes its capacity to distinguish small objects or classes with fine similarities. Subsequently,

gaPCA proved to be more appropriate for hyperspectral images with small structures or objects that need to be identified or where generally spectral classes or spectrally similar classes are encountered.

- In the face recognition experiment, when looking at the recognition accuracy numbers, gaPCA performed equally or slightly lower (under 10%) with the canonical PCA. For the case where face recognition was performed using a neural network classifier, gaPCA scored again very close to its counterpart, with just under 2% in average compared to the canonical PCA, with superior results for specific classes. Additionally, gaPCA was shown to decreases the number of training iterations required for the neural network classifier.

- The gaPCA algorithm was implemented using multi-core Central Processing Units (CPU), Graphics Processing Units (GPU) and multi-core CPU using AVX2 intrinsics, and a comparative evaluation of the implementations in terms of execution time and energy consumption was made. The experimental evaluation has shown that all parallel implementations have consistent speed-ups over the single core version: the C++ CUDA was on average 29.3× faster on Pavia and 24.8× faster on Indian Pines, while the C++ MC AVX2 version had an average speed-up of 21.2× for Pavia and 23.9× for Indian Pines compared to the baseline C++ SC version. These results highlight that the CUDA-based GPU implementation of the gaPCA method has significant advantages with regard to computing performance (execution time) and energy consumption; in addition, the multi-core CPU using AVX2 intrinsics implementation was shown to provide very similar performances as the CUDA implementation. Consequently, the benefits of using parallel computing, AVX2 intrinsics and CUDA parallel programming paradigms have been highlighted for accelerating dimensionality reduction algorithms like gaPCA in order to obtain significant speed-ups and improved energy efficiency over the traditional CPU single-core or multi-core implementations.

- The energy consumption measurement results for the various single-core, multi-core and CUDA-based gaPCA implementations showed on one hand that all parallel solutions consume much less total energy than the single-core implementation and on the other, that among all parallel implementation, the most efficient was the C++ MC AVX2 implementation, followed closely by the C++ MC CUDA implementation and the PyCUDA version.

6.2 Original contributions

Among the original contributions of this doctoral research, which conducted to the achievement of the objectives, we enumerate:

- *A study on the current state of the algorithms and methods for multidimensional data analysis*

- *The design and implementation of a new dimensionality reduction method [46]*

- *The validation of the novel gaPCA method on synthetic data [45]*

- *The validation of the novel gaPCA method on remote sensing data visualization [45]*
- *The validation of the novel gaPCA method using quality metrics on remote sensing data [46]*
- *The validation of the novel gaPCA method on remote sensing data classification [46][47]*
- *The validation of the novel gaPCA method in face recognition [49]*
- *The parallelization of the novel gaPCA method using multi-core Central Processing Units (CPU), Graphics Processing Units (GPU) and multi-core CPU using AVX2 intrinsics [48][44]*
- *A cross-platform and cross-language assesment of the of the gaPCA algorithm and its multi-core and GPU implementations was performed [48][44]*
- *Energy consumption analysis of the various gaPCA single-, multi-core and CUDA implementations [48]*

6.3 Future research directions

Regarding the future research directions that can capitalize on the obtained results, the following can be mentioned:

- Extending the application range of the gaPCA method to the target detection field.
- Extending the application range of the gaPCA method to other types of multidimensional data, like biomedical data, data recordings from sensors, traffic data, etc.
- Extending the comparison of the gaPCA method's performances with other dimensionality reduction methods like Linear Discriminant Analysis, Minimum Noise Fraction, etc.

6.4 Dissemination and validation of the research results

The results of the doctoral research obtained and presented in this thesis were validated by the international scientific community and capitalized by publishing in journals and specialized journals, as well as and in the volumes of prestigious international conferences.
Scientific articles published in Web of Science-indexed Journals:

- [46] A. L. Machidon, F. Del Frate, M. Picchiani, O. M. Machidon, and P. L. Ogrutan. Geometrical Approximated Principal Component Analysis for Hyperspectral Image Analysis. *Remote Sensing*, 12(11), 2020 (Impact Factor = 4.509)
- [48] A. L. Machidon, O. M. Machidon, C. B. Ciobanu, and P. L. Ogrutan. Accelerating a Geometrical Approximated PCA Algorithm Using AVX2 and CUDA. *Remote Sensing*, 12(12):1918, 2020 (Impact Factor = 4.509)

- [26] L. Fasano, D. Latini, A. L. Machidon, C. Clementini, G. Schiavon, and F. Del Frate. SAR Data Fusion Using Nonlinear Principal Component Analysis. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2019 (Impact Factor = 3.833) also indexed in IEEE Xplore Digital Library

Scientific articles presented at peer-reviewed international conferences and indexed in Web of Science - proceedings paper:

- [44] A. L. Machidon, C. B. Ciobanu, O. M. Machidon, and P. L. Ogrutan. On Parallelizing Geometrical PCA Approximation. In *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pages 1–6. IEEE, 2019 also indexed in IEEE Xplore Digital Library

- [49] A. L. Machidon, O. M. Machidon, and P. L. Ogrutan. Face Recognition Using Eigenfaces, Geometrical PCA Approximation and Neural Networks. In *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pages 80–83, 2019 also indexed in IEEE Xplore Digital Library

- [45] A. L. Machidon, R. Coliban, O. Machidon, and M. Ivanovici. Maximum Distance-based PCA Approximation for Hyperspectral Image Analysis and Visualization. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–4, 2018 also indexed in IEEE Xplore Digital Library

Scientific articles and abstracts presented at other peer-reviewed conferences and symposiums:

- [25] L. Fasano, F. Del Frate, D. Latini, A. L. Machidon, and C. Clementini. Classification of Urban areas by Means of Multiband SAR Data Fusion. In *European Space Agency 2nd Mapping Urban Areas from Space 2018 - MUAS 2018*. ESA, 2018

- [47] A. L. Machidon, M. Ivanovici, R. Coliban, and F. Del Frate. A Geometrical Approximation of PCA for Hyperspectral Data Dimensionality Reduction. In *The ESA Earth Observation Phi-week EO Open Science and FutureEO*. ESA, 2018

Contributions to the teaching activity during the PhD studies:

- [54] P. L. Ogrutan, A. L. Machidon, and A. Dinu. Is There a Link Between Creativity and Multiculturalism in Education? *TEM Journal*, 8(2):577, 2019

Bibliography

[1] AMD Ryzen 5 3600 Processor Specifications. https://www.amd.com/en/products/cpu/amd-ryzen-5-3600. Accessed: 2019-09-17.

[2] AVX2 Overview. https://software.intel.com/en-us/cpp-compiler-developer-guide-and-reference-overview-intrinsics-for-intel-advanced-v Accessed: 2019-01-22.

[3] Cambridge Database of Faces. https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. Accessed: January 28, 2019.

[4] ENVI online documentation. https://www.harrisgeospatial.com/Portals/0/pdfs/HG_ENVI_brochure_WEB.pdf. Accessed: December 07, 2018.

[5] FEI Face Database. http://fei.edu.br/cet/facedatabase.html. Accessed: January 08, 2019.

[6] Gen-Z: A New Approach. Gen-Z Consortium. https://www.youtube.com/watch?v=OTsk8B-EnmM. Accessed: February 14, 2019.

[7] Mathworks. Principal Component Analysis (PCA). Online documentation. https://www.mathworks.com/help/stats/principal-component-analysis-pca.html. Accessed: February 5, 2018.

[8] Multidimensional - In Merriam-Webster's Dictionary. https://www.merriam-webster.com/dictionary/multidimensional. Accessed: November 27, 2018.

[9] OpenMP. https://www.openmp.org/. Accessed: 2019-12-19.

[10] Peaktech Power Meter. https://www.peaktech.de/productdetail/kategorie/digital-leistungszangenmessgeraet/produkt/peaktech-1660.html/. Accessed: 2020-01-24.

[11] PyCUDA. https://mathema.tician.de/software/pycuda/. Accessed: 2019-08-19.

[12] Yale Face Database. `http://vision.ucsd.edu/content/yale-face-database`. Accessed: January 20, 2019.

[13] Y. Aït-Sahalia and D. Xiu. Principal component analysis of high-frequency data. *Journal of the American Statistical Association*, pages 1–17, 2018.

[14] I. S. Bajwa, M. Naweed, M. N. Asif, and S. I. Hyder. Feature based image classification by using principal component analysis. *ICGST Int. J. Graph. Vis. Image Process. GVIP*, 9:11–17, 2009.

[15] A. Barcaru. Supervised projection pursuit – A dimensionality reduction technique optimized for probabilistic classification. *Chemometrics and Intelligent Laboratory Systems*, 194:103867, 2019.

[16] D. Báscones, C. González, and D. Mozos. Hyperspectral Image Compression Using Vector Quantization, PCA and JPEG2000. *Remote Sensing*, 10(6):907, 2018.

[17] J. B. Campbell and R. H. Wynne. *Introduction to remote sensing*. Guilford Press, 2011.

[18] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[19] G. Cheng, J. Han, and X. Lu. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *Proceedings of the IEEE*, 105(10):1865–1883, Oct 2017.

[20] M. Chi, A. Plaza, J. A. Benediktsson, Z. Sun, J. Shen, and Y. Zhu. Big Data for Remote Sensing: Challenges and opportunities. *Proceedings of the IEEE*, 104(11):2207–2219, 2016.

[21] A. Davies, K. Lahiri, et al. A new framework for analyzing survey forecasts using three-dimensional panel data. *Journal of Econometrics*, 68(1):205–228, 1995.

[22] S. Ding, L. Chen, and J. Li. Dimension reduction of local manifold learning algorithm for hyperspectral image classification. In *Intelligent Image and Video Interpretation: Algorithms and Applications*, pages 217–231. IGI Global, 2013.

[23] Q. Du and J. E. Fowler. Hyperspectral image compression using JPEG2000 and principal component analysis. *IEEE Geoscience and Remote sensing letters*, 4(2):201–205, 2007.

[24] D. Eni, A. Iwara, and R. Offiong. Analysis of soil-vegetation interrelationships in a south-southern secondary forest of Nigeria. *International Journal of Forestry Research*, 2012, 2012.

[25] L. Fasano, F. Del Frate, D. Latini, A. L. Machidon, and C. Clementini. Classification of Urban areas by Means of Multiband SAR Data Fusion. In *European Space Agency 2nd Mapping Urban Areas from Space 2018 - MUAS 2018*. ESA, 2018.

[26] L. Fasano, D. Latini, A. L. Machidon, C. Clementini, G. Schiavon, and F. Del Frate. SAR Data Fusion Using Nonlinear Principal Component Analysis. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2019.

[27] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on computers*, 100(9):881–890, 1974.

[28] V. Hlaváč. Principal Component Analysis (PCA) Application to images. `http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/11ImageProc/15PCA.pdf`. Lecture notes, Czech Technical University in Prague. Accessed: December 20, 2018.

[29] E. Homenauth, D. Kajeguka, and M. A. Kulkarni. Principal component analysis of socioeconomic factors and their association with malaria and arbovirus risk in Tanzania: a sensitivity analysis. *J Epidemiol Community Health*, pages jech–2017, 2017.

[30] G. B. Huang and E. Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep*, pages 14–003, 2014.

[31] L. O. Jimenez and D. Landgrebe. Projection pursuit for high dimensional feature reduction: Parallel and sequential approaches. In *1995 International Geoscience and Remote Sensing Symposium, IGARSS'95. Quantitative Remote Sensing for Science and Applications*, volume 1, pages 148–150. IEEE, 1995.

[32] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[33] X. Kang, B. Zhuo, and P. Duan. Semi-supervised deep learning for hyperspectral image classification. *Remote Sensing Letters*, 10(4):353–362, 2019.

[34] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, and A. Hooman. An overview of principal component analysis. *Journal of Signal and Information Processing*, 4(03):173, 2013.

[35] Q. Ke and T. Kanade. Robust l/sub 1/norm factorization in the presence of outliers and missing data by alternative convex programming. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 739–746. IEEE, 2005.

[36] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern analysis and Machine intelligence*, 12(1):103–108, 1990.

[37] P. V. Krishna, M. R. Babu, and E. Ariwa. *Global Trends in Information Systems and Software Applications: 4th International Conference, ObCom 2011, Vellore, TN, India, December 9-11, 2011, Part II. Proceedings*, volume 270. Springer, 2012.

[38] T.-W. Lee. Independent component analysis. In *Independent component analysis*, pages 27–66. Springer, 1998.

[39] M. Lennon, G. Mercier, M. Mouchot, and L. Hubert-Moy. Independent component analysis as a tool for the dimensionality reduction and the representation of hyperspectral images. In *Geoscience and Remote Sensing Symposium, 2001. IGARSS'01. IEEE 2001 International*, volume 6, pages 2893–2895. IEEE, 2001.

[40] H. Lin and A. Zhang. Summarization of hyperspectral image visualization methods. In *2014 IEEE Int. Conf. Prog. Info. Comp.*, pages 355–358, May 2014.

[41] C. Lomont. Introduction to Intel Advanced Vector Extensions. *Intel white paper*, 23, 2011.

[42] D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *Int. J. Remote Sens.*, 28(5):823–870, 2007.

[43] Y. Luo, L. K. John, and L. Eeckhout. Self-monitored adaptive cache warm-up for microprocessor simulation. In *16th Symposium on Computer Architecture and High Performance Computing*, pages 10–17, Oct 2004.

[44] A. L. Machidon, C. B. Ciobanu, O. M. Machidon, and P. L. Ogrutan. On Parallelizing Geometrical PCA Approximation. In *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pages 1–6. IEEE, 2019.

[45] A. L. Machidon, R. Coliban, O. Machidon, and M. Ivanovici. Maximum Distance-based PCA Approximation for Hyperspectral Image Analysis and Visualization. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–4, 2018.

[46] A. L. Machidon, F. Del Frate, M. Picchiani, O. M. Machidon, and P. L. Ogrutan. Geometrical Approximated Principal Component Analysis for Hyperspectral Image Analysis. *Remote Sensing*, 12(11), 2020.

[47] A. L. Machidon, M. Ivanovici, R. Coliban, and F. Del Frate. A Geometrical Approximation of PCA for Hyperspectral Data Dimensionality Reduction. In *The ESA Earth Observation Phi-week EO Open Science and FutureEO*. ESA, 2018.

[48] A. L. Machidon, O. M. Machidon, C. B. Ciobanu, and P. L. Ogrutan. Accelerating a Geometrical Approximated PCA Algorithm Using AVX2 and CUDA. *Remote Sensing*, 12(12):1918, 2020.

[49] A. L. Machidon, O. M. Machidon, and P. L. Ogrutan. Face Recognition Using Eigenfaces, Geometrical PCA Approximation and Neural Networks. In *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pages 80–83, 2019.

[50] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[51] F. Melgani and L. Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on geoscience and remote sensing*, 42(8):1778–1790, 2004.

[52] Y. Mori, M. Kuroda, and N. Makino. *Nonlinear principal component analysis and its applications*. Springer, 2016.

[53] A. Norko. Simple image classification using principal component analysis (PCA). *GMU Volgenau School of Engineering, Fairfax, VA, USA*, 9, 2015.

[54] P. L. Ogrutan, A. L. Machidon, and A. Dinu. Is There a Link Between Creativity and Multiculturalism in Education? *TEM Journal*, 8(2):577, 2019.

[55] A. Peleg and U. Weiser. MMX technology extension to the Intel architecture. *IEEE micro*, 16(4):42–50, 1996.

[56] D. Peña, F. J. Prieto, and J. Viladomat. Eigenvectors of a kurtosis matrix as interesting directions to reveal cluster structure. *Journal of Multivariate Analysis*, 101(9):1995–2007, 2010.

[57] S. A. Priyanka, Y.-K. Wang, and S.-Y. Huang. Low-light Image Enhancement by Principal Component Analysis. *IEEE Access*, 2018.

[58] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang. A PCA-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944. ACM, 2015.

[59] S. K. Raman, V. Pentkovski, and J. Keshava. Implementing streaming SIMD extensions on the Pentium III processor. *IEEE micro*, 20(4):47–57, 2000.

[60] C. Rodarmel and J. Shan. Principal component analysis for hyperspectral image classification. *Surveying and Land Information Science*, 62(2):115–122, 2002.

[61] T. Roughgarden and G. Valiant. CS168: The Modern Algorithmic Toolbox Lecture# 7: Understanding and Using Principal Component Analysis (PCA). 2016.

[62] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig. Non-linear PCA: a missing data approach. *Bioinformatics*, 21(20):3887–3895, 2005.

[63] J. Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[64] C. O. S. Sorzano, J. Vargas, and A. P. Montano. A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*, 2014.

[65] L. van der Maaten, E. O. Postma, and J. van den Herik. Dimensionality reduction: A comparative review. 2009.

[66] L. Wang, Z.-H. You, X. Yan, S.-X. Xia, F. Liu, L.-P. Li, W. Zhang, and Y. Zhou. Using two-dimensional principal component analysis and rotation forest for prediction of protein-protein interactions. *Scientific reports*, 8(1):1–10, 2018.

[67] M.-H. Yang, N. Ahuja, and D. Kriegman. Face recognition using kernel eigenfaces. In *Image processing, 2000. proceedings. 2000 international conference on*, volume 1, pages 37–40. IEEE, 2000.

[68] S. Zhang and M. Turk. Eigenfaces. *Scholarpedia*, 3(9):4244, 2008.

ANNEX A

Abstract

The PhD thesis entitled "Algorithms and statistical methods for multidimensional data analysis" presents the design, implementation, testing, validation and optimization of a novel dimensionality reduction method, related to the Principal Component Analysis (PCA) technique, namely the gaPCA method (Geometrical Approximated Principal Component Analysis), and its applications in several domains like Remote Sensing and Face Recognition. The thesis is structured in six chapters, in addition to the Introduction. In the first chapter, the current state of the art on dimensionality reduction is presented, with focus on the Principal Component Analysis method: advantages and disadvantages, applications in various domains and adaptations. Chapter 2 introduces the gaPCA novel method based on a geometrical construction as an alternative to the canonical Principal Component Analysis. Chapter 3 presents the validation of gaPCA in the field of hyperspectral images for the purposes of image analysis and classification. The performance of the gaPCA method was evaluated using several metrics, for both image quality assessment, quality of the reconstruction, redundancy of the information, and accuracy of the classification and the results were compared with the ones from the canonical PCA. Chapter 4 describes the results of the gaPCA method in the field of face recognition and its performance in terms of accuracy of the recognition, with the canonical PCA as a benchmark. Chapter 5 presents the gaPCA implementations using parallel computing principles on different hardware architectures, for accelerating the computation time to obtain speed-ups and improved energy efficiency. Finally, Chapter 6 summarizes the conclusions of the previous chapters and reviews the original contributions made in this doctoral thesis research. It also presents several future research directions for extending the existing contributions.

Rezumat

Teza de doctorat intitulată "Algoritmi și metode statistice pentru analiza datelor multidimensionale" prezintă conceperea, proiectarea, implementarea, testarea, validarea și optimizarea unei noi metode de reducere a dimensionalității, inspirată de Principal Component Analysis (PCA), și anume metoda gaPCA (Geometrical Approximated Principal Component Analysis) și aplicațiile sale în diverse domenii precum Remote Sensing și recunoașterea facială. Teza este structurată pe șase capitole, pe lângă Introducere. În primul capitol este prezentat stadiul actual privitor la reducerea dimensionalității, cu accent pe PCA: avantaje și dezavantaje, aplicații în diferite domenii și diverse implementări. Capitolul 2 introduce metoda inovativă bazată pe o construcție geometrică gaPCA, ca alternativă la PCA. Capitolul 3 prezintă validarea gaPCA în domeniul imaginilor hiperspectrale cu accent pe analiza imaginilor și clasificare. Performanța metodei gaPCA a fost studiată prin evaluarea calității imaginii, calității reconstrucției, redundanța informației și acuratețea clasificării, iar rezultatele au fost comparate cu cele ale PCA. Capitolul 4 descrie rezultatele metodei gaPCA în domeniul recunoașterii faciale în ceea ce privește acuratețea recunoașterii, având ca reper rezultatele obținute cu metoda PCA. Capitolul 5 prezintă implementările gaPCA utilizând principii de calcul paralel pe arhitecturi hardware diferite, pentru accelerarea timpului de calcul și îmbunătățirea eficienței energetice. În cele din urmă, capitolul 6 rezumă concluziile capitolelor anterioare și trece în revistă contribuțiile originale realizate în această cercetare doctorală. Sunt prezentate de asemenea mai multe direcții viitoare de cercetare pentru extinderea contribuțiilor existente.