DOI: 10.5604/20830157.1159324

OPTIMIZATION OF 3D LOCAL ORIENTATION MAP CALCULATION IN THE MATLAB FRAMEWORK

Ranya Al Darwich, Laurent Babout
Lodz University of Technology, Institute of Applied Computer Science

Abstract. *This paper presents the development and evaluation of a new approach toward the optimization of 3D local orientation map calculation in the Matlab framework. This new approach can be detailed as: optimize eigenvector calculation for PCA analysis of X-ray micro tomography images of lamellar Titanium alloys image. We use two different methods to find the eigenvector of the largest eigenvalue and compare them with the Matlab built-in function (eigs). The results show a steep decrease of the calculation time using the authors' method compared to the Matlab built-in function.*

Keywords: local orientation, PCA analysis, eigenvalue, eigenvector, matrix of inertia

OPTYMALIZACJA OBLICZEŃ TRÓJWYMIAROWYCH LOKALNYCH MAP KIERUNKOWYCH Z UŻYCIEM ŚRODOWISKA OBLICZENIOWEGO MATLAB

Streszczenie. *W artykule przedstawiono rozwój i ocenę nowego podejścia dotyczącego optymalizacji obliczeń 3D lokalnych orientacji map w środowiska Matlab. Zastosowano dwie różne metody wyznaczania wektora własnego największej wartości własnej. Wyniki są porównywane z wynikami otrzymanymi przy pomocy wbudowanych w pakiecie Matlab funkcji wyznaczające wektory i wartości własne. Wyniki porównania pokazują redukcję czasu obliczeń przy użyciu autorskiej metody w stosunku do funkcji wbudowanej w Matlab.*

Słowa kluczowe: orientacja lokalna gradientu, analiza PCA, wartość własna, wektor własny, macierz bezwładności

Introduction

There is a non-negligible amount of structured materials that show a local orientation in their microstructures. For instance the microstructure of titanium alloys can appear with a 3dimensional lamellar texture [1], or fiber composite materials can be designed with a woven pattern like textile fabrics (see fig. 1). They can be classified as texture materials and one may need to locally estimate the orientation of the features for further quality or property analysis. These orientation can be extracted from 2D/3D images and popular image processing method, which is based on gradient estimation and its matrix representation using matrix of inertia [4] or Hessian matrix [2] are frequently used. Next the local orientation can be calculated using principal component analysis (or matrix diagonalization), where the eigenvector of the largest/smallest eigenvalue represents the local direction. The main problem that arises from such methodology is the very time consuming process of matrix reduction as it is done with pixel/voxelwise operations. For instance, if one assume a 3D image of size 500^3, the Matlab built-in function (eigs), which uses the Arnoldi iteration method [6] to obtain the eigenvalues/eigenvectors, will have to be run sequentially 125 millions times, which is computationally very slow.

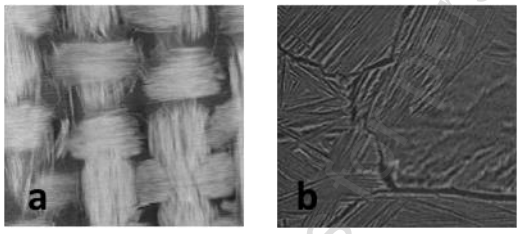

Fig. 1. Examples of microstructure, a) fibrous microstructure of glass fiber reinforced polymers (GFRP) b) lamellar microstructure of Titanium alloys (Ti)

Therefore, this paper aims at proposing alternative methods to speed up the calculation of eigenvector. The main objective is to use matrix operations and avoid loops, which are known as the main slowing elements in algorithms. The two proposed approaches are compared with the standard method based on the Matlab built-in function (eigs). Both eigenvector maps and computation times are compared to testify the usefulness of the new approaches. The aim of this paper is introducing methods to optimization of 3D local orientation. These methods are tested in 2D and 3D image of X-ray micro tomographic images of Titanium lamellar alloys (Ti) [1], and glass fiber reinforced polymers (GFRP) [7].

1. Algorithms

PCA, or Principal Component Analysis**,** is the most important 3-dimensionality reduction technique. This technique was initially employed by statisticians to reduce the variables into a lower number of orthogonal variables (factors), which are also called eigenvectors. In this paper we will calculate the eigenvectors and eigenvalues of the data covariance matrix using three methods. The eigenvector corresponding to the largest eigenvalue is the direction of greatest variation. The covariance matrix is based on the average gradient (first derivative) defined in the neighbourhood $W(p)$ of each pixel/voxel p composing the 2D/3D image I. In a more formal way, let consider the Definition domain $\Delta(N_1, N_2, N_3)= \{1,2,\dots N_1\}x\{1,2,\dots N_2\}x\{1,2,\dots N_3\}$ of an input volumetric image I: $\Delta (N_1, N_2, N_3)\rightarrow\{0,1,2,\dots,255\}$

To find the principle orientation of an image we use the matrix of inertia, we consider a neighbouring window W of size s^n (in n-dimensional space R^n) located around every point $(i,j,k) \in \Delta$, inside this window the matrix of inertia $\mathbf{J}_{ijk}$ is given by (in the 3D case):

$$J_{ijk}=\begin{pmatrix} \left\langle \frac{\partial I}{\partial x}\right\rangle^2 & \left\langle \frac{\partial I}{\partial x}\right\rangle\left\langle \frac{\partial I}{\partial y}\right\rangle & \left\langle \frac{\partial I}{\partial x}\right\rangle\left\langle \frac{\partial I}{\partial z}\right\rangle \\ \left\langle \frac{\partial I}{\partial x}\right\rangle\left\langle \frac{\partial I}{\partial y}\right\rangle & \left\langle \frac{\partial I}{\partial y}\right\rangle^2 & \left\langle \frac{\partial I}{\partial y}\right\rangle\left\langle \frac{\partial I}{\partial z}\right\rangle \\ \left\langle \frac{\partial I}{\partial x}\right\rangle\left\langle \frac{\partial I}{\partial z}\right\rangle & \left\langle \frac{\partial I}{\partial y}\right\rangle\left\langle \frac{\partial I}{\partial z}\right\rangle & \left\langle \frac{\partial I}{\partial z}\right\rangle^2 \end{pmatrix}=\begin{pmatrix} a_{ijk} & d_{ijk} & e_{ijk} \\ d_{ijk} & b_{ijk} & f_{ijk} \\ e_{ijk} & f_{ijk} & c_{ijk} \end{pmatrix} \quad (1)$$

where $\langle . \rangle$ denotes the averaging operation in the neighbourhood W (usually performed using convolution with a matrix of 1 in the Fourier domain for decrease processing time, especially in the 3D space).

At that stage we obtain a n-D map of matrix of inertia $\mathbf{J}$. Since $\mathbf{J_{ijk}}$ is symmetric, each point in the n-D space is represented by a vector of size $n(n+1)/2$. For instance, in the 3D case, this vector is of length 6 and can be represented as follows for a point $(i,j,k) \in \Delta$:

$$\mathbf{v}_{ijk}^{J}=\left(a_{ijk}\, d_{ijk}\, e_{ijk}\, b_{ijk}\, f_{ijk}\, c_{ijk}\right) \quad (2)$$

and the map of matrix of inertia takes the following form:

$$\mathbf{J}=\left(\mathbf{A}\,\mathbf{D}\,\mathbf{E}\,\mathbf{B}\,\mathbf{F}\,\mathbf{C}\right) \quad (3)$$

with $\mathbf{A}=\left\{\mathbf{a}_{ijk}\right\}$ where $\mathbf{a}_{ijk}=a_{ijk}$, $\mathbf{B}=\left\{\mathbf{b}_{ijk}\right\}$ where $\mathbf{b}_{ijk}=b_{ijk}$ and so on.

In what follows, we will present the 2 proposed methods that perform the eigenvector calculation on **J**.

1.1. Method 1: analytical method

This method relies on the *determinant solution*. Let **M** be an *n*-by-*n* matrix. So λ is an eigenvalue if:

$$\mathrm{Det}[\mathbf{M}-\lambda\mathbf{I}]=0 \tag{4}$$

where **I** is the *n*-by-*n* identity matrix.

This equation gives us a characteristic polynomial in λ of degree n, and the roots of this equation are the eigenvalues. In three-dimensional space we will get a cubic equation ($a\,\lambda^3+b\,\lambda^2+c\,\lambda+d=0$), that is solvable analytically using the well-known Cardano method published in 1545 [3]:

$$\mathbf{x}=\sqrt[3]{\mathbf{q}+\sqrt{\mathbf{q}^2+\left(\mathbf{r}-\mathbf{p}^2\right)^3}}+\sqrt[3]{\mathbf{q}-\sqrt{\mathbf{q}^2+\left(\mathbf{r}-\mathbf{p}^2\right)^3}}+\mathbf{p} \tag{5}$$

where:

$$\mathbf{p}=-\mathbf{b}/3a$$

$$\mathbf{q}=\mathbf{p}^3+\left(\mathbf{bc}-3a\mathbf{d}\right)/6a^2$$

$$\mathbf{r}=\mathbf{c}/3a$$

In the case of **J** (eq. 3), the set of variables defining the cubic equation are:

$$a=-1$$

$$\mathbf{b}=\mathbf{A}+\mathbf{B}+\mathbf{C}$$

$$\mathbf{c}=-\left(\mathbf{A}\circ\mathbf{C}+\mathbf{A}\circ\mathbf{B}+\mathbf{B}\circ\mathbf{C}-\mathbf{D}^2-\mathbf{E}^2-\mathbf{F}^2\right)$$

$$\mathbf{d}=\mathbf{A}\circ\mathbf{B}\circ\mathbf{C}+2\mathbf{E}\circ\mathbf{D}\circ\mathbf{F}-\mathbf{A}\circ\mathbf{F}^2-\mathbf{C}\circ\mathbf{D}^2-\mathbf{B}\circ\mathbf{E}^2$$

where $\circ$ denotes the Hadamart product (i.e. an element-wise multiplication). Knowing one root of the cubic equation representing the diagonalization of the tensor of inertia, it is straightforward to estimate the 2 other roots by reducing the polynomial by one degree.

What is important to notice here is that in the previous equations of the eigenvalue, each parameter can be considered as a 3D matrix, where each point corresponds to the local eigenvalue from the input image, following the PCA reduction of the tensor of inertia. This approach is well adapted for $n\leq 3$. For larger dimension, the next method is more appropriate.

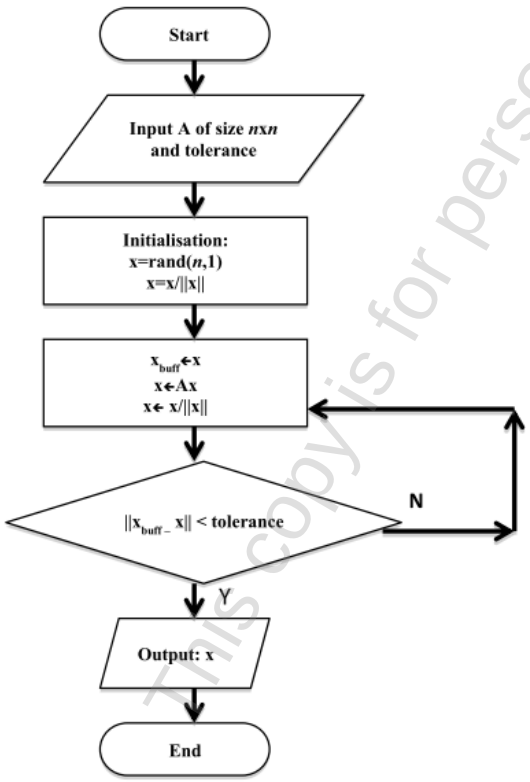

Fig. 2. Flow chart of the power iteration method

1.2. Method 2: power iteration

We implement this method to find iteratively the largest eigenvalue and its corresponding eigenvector from an initial eigenvector guess [5] (see Fig. 2 for the general power iteration flow chart). The power iteration algorithm is frequently used in the cases when the first couple of eigenvalues need to be computed, as in the case of searching the main spatial feature orientation. As in the previous method, the input is the map of matrix of inertia **J**. The iteration step is initially performed in parallel on each voxel yielding a vector corresponding to the local matrix of inertia $\mathbf{J}_{ijk}$. However, the number of points to be considered in the next iteration is decreased as the Euclidean distance between the eigenvectors at 2 successive steps is smaller than a given tolerance (typically 10^{-3}). Obviously, the number of updates of local eigenvector decreases as the number of iteration increases. This approach speeds up the iteration procedure and therefore the overall execution time of the algorithm.

1.3. Method 3: Matlab built-in function

In this method we calculate the eigenvalue and the eigenvector by using the Matlab built-in function *eigs*.

Eigs(**A**) solves the eigenvalue problem using the Arnoldi iteration [6]. While the approach is well suited for large sparse matrix, it is not designed to solve in parallel a large set of eigenvalue/vector decompositions, as in the case of the two previous methods.

Let consider, as previously, every point $(i,j,k) \in \Delta$ of an input image *I* and the corresponding matrix of inertia $\mathbf{J}_{ijk}$. The pseudo code is the following:

For $i \leftarrow 1$ *to* N_1
  For $j \leftarrow 1$ *to* N_2
    For $k \leftarrow 1$ *to* N_3
      $[\mathbf{x}_{ijk}, \lambda_{ijk}] \leftarrow \mathrm{eigs}(\mathbf{J}_{ijk})$

where $\mathbf{x}_{ijk}$ and λ_{ijk} are the eigenvectors and eigenvalues of $\mathbf{J}_{ijk}$.

2. Results

The comparison between the 3 methods is done in the following way. The results are verified qualitatively by comparing the map of eigenvector projection with respect to the main axis [001] (i.e. z-axis of the 3D image). Then comparison about the CPU time (i.e. execution time) is performed to estimate which method is the most appropriate to estimate local orientation in 3D.

Fig. 3 shows the result about the projection map for the 3 tested methods. Once can clearly see that they give very similar results. This has been done in the case of an input image of size 100^3.

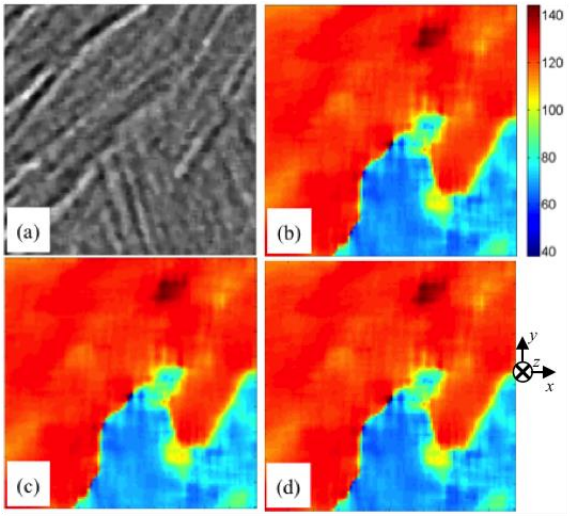

Fig. 3. a) the input image; b) the projection map calculated using method #1 (analytical method); c) the projection map calculated using method #2 (power iteration); d) the projection map calculated using method #3 (Matlab built-in function). The colour bar correspond to the projection angle (in degree)

One can also judge the accuracy of the calculation looking at the mean square error (MSE) of the 3D projection maps between method #3 (taken here as the reference method) and methods #1 and #2. In the case of the comparison between method #1 and #3, MSE is ~10^{-11}, while it is 0.09 between method #2 and #3. In the latter case, this MSE result is obtained for a tolerance value of 10^{-3}. This is considered as a good compromise between the execution time and the MSE, as shown in Fig. 4. Again, this calculation has been done in the case of an input image of size 100^3.

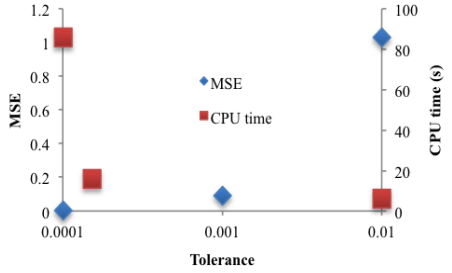

Fig. 4. Evolution of MSE and CPU time with respect to the tolerance, for an input image of size 100^3

Fig. 5 shows the evolution of the computation time as a function of the number of voxels of an input image, for the 3 tested methods. One can see that the three evolutions are linear (at least for the given range of image sizes that are lower than 100^3), but the fastest is clearly the method #1, which takes advantages of the analytical approach for the Eigen decomposition based on Hadamart product. Method #3 is particularly slow because of the algorithm that estimates voxel-by-voxel the eigenvector/eigenvalue using the Matlab built-in function *eigs*.

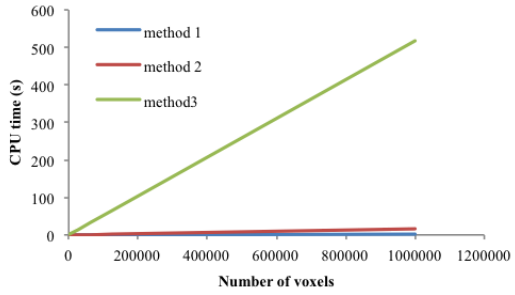

Fig. 5. Evolution of the CPU time with respect to the number of voxels for the 3 tested methods

Table 1 presents the results of our study, which once again shows that the calculation time of the determinant solution (i.e. method #1) is the smallest, which justifies the use of such method to accurately calculate main orientation of features in 3D images in the smallest time. This choice is also justified by the corresponding MSE value presented above. Also the table reveals that in average, method #1 is faster than method #2 by about 10 times, while it is ~500 times faster than method #3. Note that in this table, we disregard the calculation time for the Matlab built-in function (method #3) when the number of element is more than 100^3 because of its extremely long processing time for larger data set.

Table 1. Summary of execution time for the 3 tested methods and different 3D images sizes

Number of elements	CPU time for method #1 (analytical method) (s)	CPU time for method #2 (power iteration) (s)	CPU time for method #3 (Matlab built-in function) (s)
50^3	0.13	1.13	66.2
100^3	1.6	16	517.3
200^3	10.6	166	-
400^3	231.2	2621.5	-

Table 2 summarises the results and the properties of the 3 methods. In the case of matrix of inertia of dimension n≤3, method #1 is the best choice, considering both its computational time and accuracy. However, for largest dimensions, the power iteration should be considered, even if the accuracy depends on the chosen tolerance level. However, perspective work will aim at generalizing the Arnoldi/Lanczos method [5,6], which is used in the Eigs function (i.e. incorporating Hadamart product for matrix operations) to circumvent the main execution time drawback.

Table 2. summary of the advantages and disavantages of the 3 tested methods

Methods	Advantages	Disadvantages
Method1: The determinant solution	• very fast • analytical method • exact solution	• limited to 2D/3D case • need important RAM
Method2: power iteration	• locate the dominant eigenvalue • n-D (even quite complex to program for n>3)	• iterative method convergence speed depends on tolerance • need important RAM
Method3: Matlab built-in functions	• easy programming • n-D • exact solution	• very large CPU time

In all cases, the computation has been done on a server equipped with 2 Intel®Xeon® processors (12M Cache, 2.53 GHz, 4 cores, 8 logical threads) and 24 GB of RAM.

3. Conclusion

The current paper has presented a comparison between three methods to calculate the largest eigenvalue and corresponding eigenvector for large set of matrix of inertia calculated for 3D images. The comparison shows that the first authors' method based on analytical approach is the fastest and most accurate method compared to the two other methods (power iteration, and the Matlab built-in functions) with similar accuracy. This is of great importance when dealing with large data set as the one in 3D tomography images.

References

[1] Babout L., Jopek L., Janaszewski M.: A New Directional Filter Bank for 3D Texture Segmentation: Application to Lamellar Microstructure in titanium Alloys, MVA2013 IAPR International Conference on Machine Vision Applications, May 20-23, 2013, Kyoto, JAPAN.
[2] Eriksen E.: Principal Minors and the Hessian, BI Norwegian School of Management Department of Economics, October 01, 2010.
[3] Wituła R., Słota D.: Cardano's formula, square roots, Chebyshev polynomials and radicals, Journal of Mathematical Analysis and Applications, vol 363, Issue 2, 15 March 2010 , 639–647.
[4] Jeulin D., Moreaud M.: Segmentation of 2D and 3D textures from estimates of the local orientation, Image Anal Stereol 27, 2008, 183-192.
[5] Lanczos C.: An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators, Journal of Research of the National Bureau of Standard, vol 45, No. 4, October 1950.
[6] Lehoucq R.B., Sorensen D.C.: Deflation Techniques for an Implicitly Re-Started Arnoldi Iteration, SIAMJ. Matrix Analysis and Applications, SIAM Journal on Matrix Analysis and Applications, Vol 17, Number 4, 1996, 789-821.
[7] Schell J.S.U., Renggli M., van Lenthe G.H., Mu¨ller R., Ermanni P.: Microcomputed tomography determination of glass fibre reinforced polymer mesostructure, Composites Science and Technologie,Vol66, Issue 13 October, 2006, 2016–2022.

M.Sc. Ranya Al Drwich
e-mail: raldarwich@kis.p.lodz.pl

Ranya Al Darwich is a Ph.D. student at Lodz University of Technology (TUL), Institute of Applied Computer Science. Her scientific interest covers image processing and analysis.

Prof. Laurent Babout
e-mail: Laurent.babout@p.lodz.pl

Prof. L. Babout obtained his Ph.D. degree in 2002 from INSA-Lyon (France) and D.Sc. degree in 2011 from the Lodz University of Technology (TUL). Currently he holds a position of associate professor at TUL. His scientific interest covers X-ray tomography and 3Dimage processing, with main focus on materials science applications. Prof. L .Babout is a SIAM member and author or co-author of more than 100 scientific papers,scientific papers, books and chapters.

otrzymano/received: 27.11.2014 *przyjęto do druku/accepted: 15.04.2015*