ISSN 2278-3091
Volume 9 No.2, March - April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse161922020.pdf
https://doi.org/10.30534/ijatcse/2020/161922020

From digital images to barcodes

Ismail MAMOUNI[1], Hajar CHAWQI[2]
[1]CRMEF Rabat, Morocco, mamouni.myismail@gmail.com
[2]ENSIAS Rabat, Morocco, h.chawqi@gmail.com

ABSTRACT

The recognition of objects in digital images is a key problem in the computer vision. It has received the attention of many researchers in order to evaluate and improve the performance of descriptors, especially that of the shape descriptor.

In this paper, we will consider the persistent homology as an algebraic tool measuring the topological features of shapes. To digital images, one can associate cubical complexes instead of triangulation because it reduces significantly the size of complexes. We implement this algebraic characterization by an algorithm that represents any digital image as bar codes in the form of finite union of intervals. The benefits of this algorithm is not only to be the shape descriptor, but also a shape comparator. Indeed, by using the the Bottleneck distance of barcodes, the algorithm allows us to answer how much close or far two shapes are.

Key words : Image analysis, topological data analysis, persistent homology, cubical homology, digital images, shape descriptor.

I. INTRODUCTION

Actually, a huge amount of images is produced and analyzed in different domains such as biology, industry, astronomy, medicine, security, etc. This images contains some interesting objects that have to be recognized, classified and then identified. To represent an image or a part of it and describe the pertinent information, we use the features extracted there-from [1, 9].

These extracted features are represented in different ways (vector, signature, barcode, ...). This representation is called a shape descriptor. The shape descriptors are a powerful tool used in wide spectrum of computer vision and image processing tasks such as object matching, classification, recognition and identification.

It is worth to point out that there exists a huge variety of object recognition approaches, but the general concept remains the same : An object recognition system uses training data sets containing images with known and labeled objects and extracts different types of information (color, edges, shapes and so on) based on the chosen algorithm. The first step of recognition system is to detect interest locations (objects) in the images and describe them. Once the descriptors are computed, they are compared to the objects presented in an image to recognize and identify them.

The shape descriptor rely on two types : The global features and the local ones. In the global one, the image is represented by one multidimensional feature vector, describing the information in the whole image. In the other hand, local features permit to detect interest regions in an image and represent them as n vectors where each one describes a certain feature, like color, texture, shape, orientation, ... It is well know that local features are very successful, powerful and faster than global ones. This is due to the lack of accuracy for global features which generally cannot distinguish foreground from background of an image.

However, recent mathematical developments are shedding new light on such traditional ideas and techniques from the relatively new discipline of Topological Data Analysis (TDA). By using homology, a tool in algebraic topology, one can measure several features of the shape including the numbers of component, holes, and voids (higher-dimensional versions of holes). By using persistent homology, one can then represent the lifetime of such features using a finite collection of intervals known as a barcodes. The barcode, considered here as a shape descriptor, will help us detecting an object in a certain image and identify it.

We develop here a shape descriptor, based in persistent homology ideas. It is an algorithm, of input is a digital image, and that issues as signature some barecodes. The strength of our algorithm is that it combines the two classic approaches mentioned above : the global one by involving all the pixels, and the local one by associating to each pixel 8 new neighbors thanks to the Low Star Process (LSP) given in Algorithm 2.

LSP is based on two priority queues, the first one associates to each pixel 8 neighbors and assign each new neighbors some values following a well stated order that will be discussed in Algorithm 2. The second queue builds from this new values a filtration that enable us to compute the

persistent homology of our data. Here, the specialist will well recognize the catching similarity between LSP, and the folkloric Local Binary Patterns (LBP, see [HPS]) which encodes the ordering relationship by comparing neighboring pixels with the center pixel, that is, it creates an order based feature for each pixel by comparing each pixel's intensity value with that of its pixels. The added value of our algorithm based on LSP is that it still encoding the ordering relationship but by adding new neighboring "pixels" and form a new matrix. All the data information are stored in the new coordinates, and one can easily come back to the initial matrix if necessary. The algorithm is smooth and does not lose any information.

Moreover, we will define a metric between the inputs (i.e., between the digital images) and another one between the outputs (i.e., between their associated barcodes). Thanks to Theorem 1, we can claim that our algorithm is stable (i.e., two similar digital images will generate two similar barcodes). Thus, we can conclude that our algorithm is not only a performing and smooth shape descriptor, but also a stable shape comparator.

The rest of the paper is organized as follows. Section 2 covers the relevant algebraic topology background material, like homotopy, simplicial homology or persistent homology. Section 3 presents our algorithm for building barcodes from a digital image. In section 4 we valid our algorithm is some well known images databases. It also contains a discussion of the computational complexity of our approach.

2. ALGEBRAIC TOPOLOGY PRELIMINARIES

In this section, we will provide the necessary algebraic topology background that will be used in the rest of the paper. It is an accessible introduction to the relevant notions, but it will be somewhat brief. The interested reader is referred to [7] for further details in general algebraic topology, to [5] for simplicial homology and to [3] for persistent homology.

2.1 Homotopy

Homotopy is an equivalence relation, that deforms continuously a shape X, viewed as a topological space, to another more simple shape Y . The two shape are said to be the same homotopy type. Indeed : Two continuous maps $f_0, f_1 : X \to Y$ are said to be homotopic if there exists a homotopy deforming continuously f_0 to f_1. That is a continuous family $(f_t)_{0 \le t \le 1}$ of continuous maps $f_t : X \to Y$, starting from f_0 and arriving to f_1. Then, we write $f_0 \sim f_1$.

Homotopy Equivalence. X and Y are said to be homotopic or of the same homotopy type, if there exists some continuous maps $f_0 : X \to Y$ and $g : Y \to X$ such that $g \circ f \sim id_X$ and $f \circ g \sim id_Y$. Then we write $X \sim Y$.

2.2 Simplicial homology

Holes of dimension n, are all shapes that bound a variety of dimension n + 1. For example, a circle, an empty triangle or an empty square are holes of dimension 1. One of the strengths of the homology theory is to count, thanks to the Betti numbers, how many holes a given shape contains. Homology can therefore be viewed as a descriptor of the topological structure of this given shape.

Simplex. Let $(a_0,..,a_p)$ be some geometrically independent points in IR^n . The convex hull of these points, denoted here $[a_0,...,a_p]$ is called a p-simplex. In other words,
$[a_0,...,a_p] := \{\sigma = \Sigma \lambda_i a_i$, with $0 \le \lambda_i \le 1$, $\Sigma \lambda_i = 1\}$. Faces of $[a_0,...,a_p]$ are all the (p-1)-simplicies $[a_0,..\hat{a}_i ..,a_p]$ where $\hat{a}_i$ means omitted.

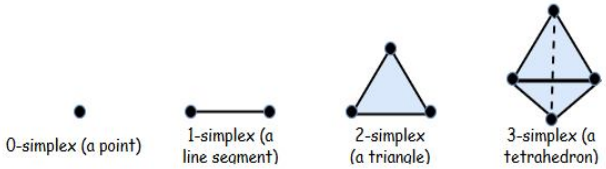


Figure 1: Simplicies with 0,1,2 dimensions

Thus one can define a simplicial complex, to be any collection K of simplicies in IR^n such that:
- Any face of a simplex from K is in K,
- Any non empty intersection of any two simplicies from K is a common face of the both two simplicies.

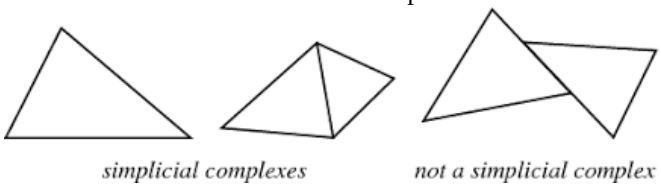


Figure 2: Example of simplicial and not simplicial complexes

Let K a given simplicial complex, a p-chain of K is any formal and finite sum $\sigma = \Sigma (-1)^i n_i \sigma_i$, where $n_i \in Z$ and σ_i a p-simplex. On the Z-module of such chains, denoted C_p (K), we define the boundary operator $\partial_p : C_p (K) \to C_{p-1} (K)$ to be formal sum $\partial_p \sigma = \Sigma (-1)^i n_i \sigma_i$. It is easy to prove that $\partial_{p-1} \circ \partial_p = 0$. Thus, we obtain a chain complex

$$0 \to C_p (K) \to C_{p-1} (K) \to ... \to C_0 (K) \to 0,$$

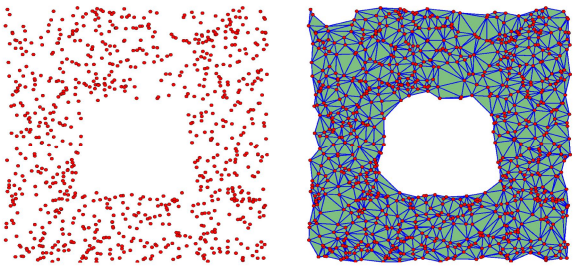
with $\mathrm{Im}\partial_p \subset \ker \partial_{p-1}$ at each connection. Elements of $\mathrm{Im}\partial_p$ are called boundaries, those of $\ker \partial_{p-1}$ are called cycles.

Homology groups. The quotient group

$$H_p (K) = \ker \partial_{p-1} / \mathrm{Im} \partial_p$$

is the p-th simplicial homology group of K, and its rank, denoted β_p (K), is the k-th Betti number of K. Since homology is invariant up to homotopy, then in the case of simplicial homology, one can replace any shape by a simplicial complex, that is a triangulation of the same type. This is feasible for applications in real life situations such as in image analysis process. It is an algebraic tool for measuring topological features of shapes and functions. It

has widespread applications in computer vision and image analysis.



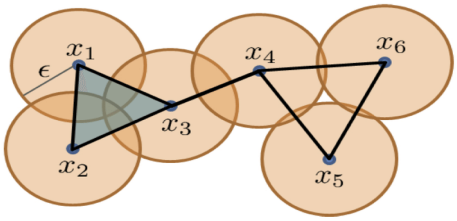
(**a**) (**b**)

Figure 3: From a topological space to a simplicial complex

2.3 Persistent homology

This section remains the theory of persistent homology and its concepts. Persistent homology, as a topological data analysis tool, is a young and quickly-developing research area at the intersection of mathematics, statistics, and computer science. It seeks to use topology to discern structure in a complex data by studying its shape. Topology takes on two main tasks : the measurement of shape and the representation of shape. Both tasks are meaningful in the context of large, complex, and high dimensional data-sets. They permit to measure shape related properties within the data, such as the presence of holes. Given a point cloud, we want to use homology to describe data, but our data is a point cloud and homology operates on simplicial complexes.

For any real number $\varepsilon > 0$, the first step is to associate to any points cloud $X = (x_i)$, the Čech complex $C(\varepsilon)$ whose p-simplicies are the $[x_0, \ldots, x_p]$, whenever the balls



B(x i,ε/2) have a common point.

Figure 4: An example of a points cloud (left) and the corresponding Česh complex

While ε is growing, other Čech complexes will appear, and this yields to a filtration of complexes

$$\square = K^0 \subseteq K^1 \subseteq \cdots \subseteq K^m = K.$$

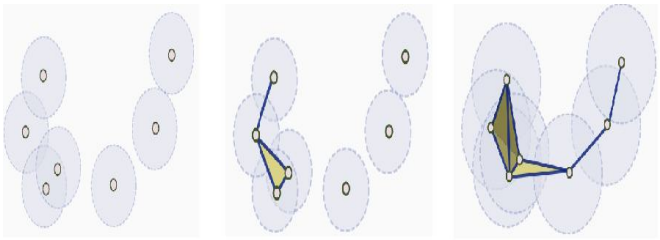


Figure 5: Example of filtered Česh complex

The associated homology of that kind of chain of complexes, is what we call the persistent homology of the initial cloud of points $X = (a_i)_i$. The key idea of persistent homology is to look for features that persist for some range

of parameter values. Typically a feature, such as a hole, will initially not be observed, then will appear, and after a range of values of the parameter it will disappear again. The resulting persistent Betti numbers are usually given as barcodes, and represent how the homology changes through the filtration.

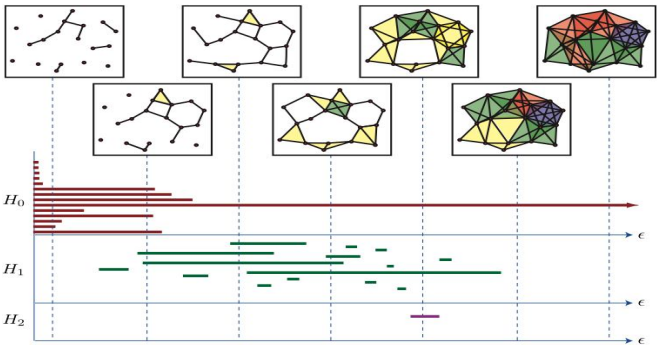


Figure 6: From points cloud to barcodes

The beginning of a barcode can be thought of as a birth time of a persistent homology class and its end as the death time. The significance of a homological feature is given by the length of the corresponding barcode. It is a way to encode the persistent homology of a data set in the form of a parametric version of a Betti number and represents each persistent generator with a horizontal line beginning at the first filtration level where it appears, and ending at the filtration level where it disappears. Persistence diagram is another equivalent way to visualize the evolution of the topological features in the filtration, by summarizing the filtration as two dimensional point sets with multiplicities. A point (x, y) with multiplicity m represents m features that all appear for the first time at scale x and disappear at scale y. Features appear before they disappear, So the points lie above the diagonal x = y. The difference $y - x$ is called the persistent of a feature. A class of homology that appears at times i and disappears in another value j will be represented by the point of coordinates (i, j). The persistent of a class will be the real value of $j - i$, this diagram is called persistent diagram because it will encode the persistent of the homology groups of the simplex. An interesting case of persistent diagrams is that associated to R-valued functions $f : X \to R$, where the filtration is given by the sub level sets $X_\varepsilon := f^{-1}(-\infty, \varepsilon]$.

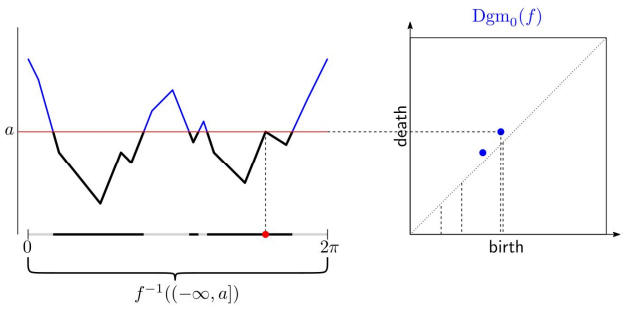


Figure 7: Example of persistent diagram

One way to compare how two persistent diagrams are closed or not, is the Bottleneck distance used to calculate the distance between two persistence diagrams D_1 and D_2 by setting $d_B(D_1, D_2) := \inf_{\mu} \sup_{x \in D1} \| x - \mu(x) \|_{\infty}$, where $\mu : D_1 \rightarrow D_2$ range all the bijections from D_1 to D_2 .

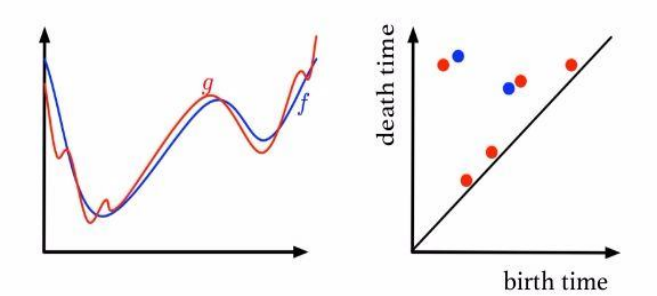


Figure 8: Example of Bottleneck distance

Stability theorem. Let f, g : X → IR two tame functions, then $d_B(Dgm(f), Dgm(g)) \leq \| f - g \|_{\infty}$.
The scientific recognition that persistent homology enjoys now is due to this fundamental result which states that a small perturbation in the input filtration leads to a small perturbation of its persistence diagram.

3. THE ALGORITHM

Our algorithm aims to represent the objects of an image (classes) as barcodes to get a shape signature (descriptor). This descriptor will be an efficient tool to recognize an object and identify it, this will be satisfied when we will build a knowledge database that contains the objects of our interest with their barcodes and their label (identification). The output of the algorithm will be compared to all the objects present in the database until we find the most similar barcode.

To the best of our knowledge, the only algorithm that encodes digital images into barcodes is that implemented in [5]. The lack in this paper is not only that the its algorithm compute just the spatiotemporal 0-barcode encoding lifetime of connected components, but unfortunately the output 0-barcode may not coincide with the one provided by the 0-barcode encoding the 0-persistent homology. In [8], the authors use Morse theory to implement an algorithm for determining the Morse complex of a 2- or 3-dimensional gray scale digital image, but they do not go further to the crucial stage: that of drawing barcodes. In [10], the authors present an efficient framework for computation of persistent homology of cubical data in arbitrary dimensions. Based on their ideas, we go further by issuing for each image a barcodes. This enables us to compare how two images are by calculating the Bottleneck distance of their respective barcodes. Many figures used is this paper are copied from this manuscript.

For our purpose, we opt for cubical persistent homology as an efficient application of persistent homology in domains where the data is naturally given in a cubical form, like the case of digital images. By avoiding triangulation of the data, we significantly reduce the size of the complex. The approach is to use n-cubes [0, 1] n instead of n-simplicies, the remainder (like boundary operator or boundary matrix) still roughly speaking the same. Simplicial complex is now called cubical homology, its persistent homology is called cubical persistent homology. In this new context, the non-null pixels play the point cloud.

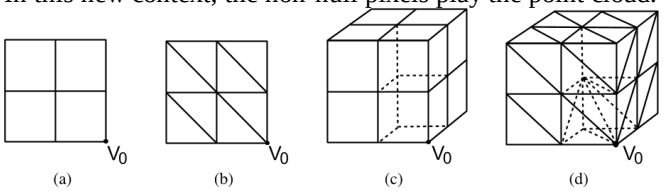


Figure 8: Cubical complex triangulation vs. complex triangulations

The cubical complex is kindly built as follows: To any pixel B(i, j) (considered as 0-cube and colored in yellow in Figure 9), we associate 8 neighbors (four 1-cubes colored in blue and four 2-cubes colored in red).

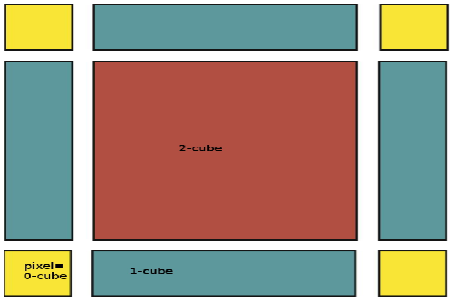


Figure 9: From pixels to cubical complex

This representation of cubical complexes from images permit us to have an idea about the relationship between cells by reading their coordinates. For each pixel, we can store the necessary information in 3x3 array. The coordinates of any cell gives immediately its dimension :

- it is an 0-cube when its coordinates are (even,even);
- it is a 1-cube when its coordinates are (even,odd) or (odd,even);
- it is a 2-cube when its coordinates are (odd, odd).

The benefit is that we can track down the past coordinates of any pixel by dividing by two the new ones. For purpose of simplification, we implement our approach on gray scale images (2D image array). The image is converted to a matrix where each element (pixel) represent the intensity or any other feature that we will precise after.

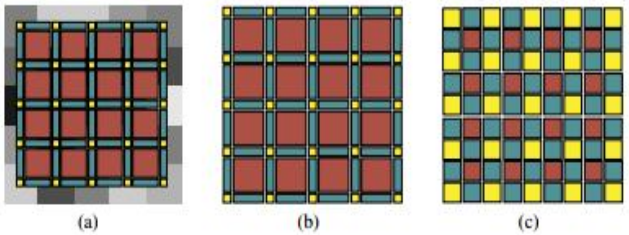


Figure 10: Cubical complex built over a gray-scale 2D image with 4x4 pixels

Before building the cubical complex. The first idea is to apply the function

$$f : B[i, j] \rightarrow B[i, j] + (i + I \,.\, j)/(\alpha \,.\, I \,.\, J)$$

where α equals the minimum non null difference between all the values of the matrix, and I, J its length-width sizes. Sometimes, when applying this function, the new values may grow hugely. To avoid this ”bad” situation, we normalize all the values by dividing it by their maximum

$$B[i, j] \leftarrow B[i, j] / \max B[i, j]$$

The goal of changing the values of the matrix is to get distinct values, and so get the ”best” filtration, by avoiding any confusion that can be induced by two similar values. As f is linear, the the effect of this perturbation can be removed later by applying the inverse-map. In all the rest of this paper, B := (B[i, j], i = 1..I, j = 1..J) denotes the initial matrix that contains the pixels-values, while
C := (C[i, j], i = 1..2 . (I − 1), j = 1..2 . (J − 1)) denotes its associated complex.

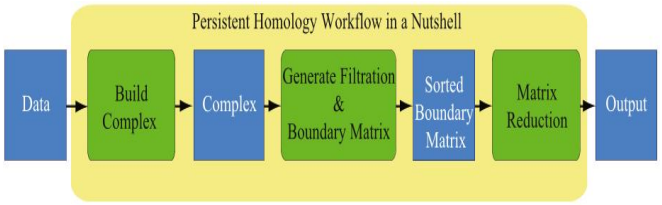


Figure 11: Persistent homology workflow in a Nutshell

3.1 Step 1 : Build the cubical complex.

Firstly, it is worth to point out, that if (i, j) are the coordinates of a pixel in the original binary matrix, those in the cubical complex should be (2i, 2j). This is another strong point to count in favor of our algorithm; that to switch smoothly between the initial and the final coordinates by dividing or multiplying by 2. Thus we define a function ”neighbors”, whose aim is to associate to each pixel B[i, j] its 8 neighbors C[2i + ε, 2j + ε’], where ε,ε’∈{−1,0,1}, with some excluded values, especially when the pixel is on the border. For example, ε ‘= −1, whenever the pixel locates in west border. Our function (see Algorithm 1) take in account this technical but useful detail.

```
Input: A matrix and the position of an element
Output: Position of the neighbors of the element
for X − 1 < x1 < X + 2 do
   for Y − 1 < y1 < Y + 2 do
      if −1 < x < X
         and −1 < y < Y
         and (X! = x2||Y ! = y2)
         and 0 < x1 < X
         and 0 < y1 < X
         then return(x1,y1)
      end if
   end for
end for
```

Algorithm 1: Function neighbors((x,y),X):(x1,y1)

Once the empty cubical complex is defined, which should be a (2I−1)×(2J−1)-matrix, the values of a pixel in this cubical complex should be the same than those in the original matrix. The next algorithm respects this rule. To fill the other values, we use the low-star rule, which say that any n-cube and all its faces should have the same values. We firstly start by the pixel of high value, and assign this value to all its neighbors (see Algorithm 2).
We move on to the next one and assign the not yet assigned neighbors and so on.

```
Input: Matrix filled with the pixels form
Output: Matrix representing the cubical complex
lp[],flag=-1
List L: elements of the matrix sorted in descending way
for i ← 0 to len(L) do
   posM ax ← P ositionOf L[i]inB
   Lneighbors ← neighbors(posM ax[0], posM ax[1], B)
   Lp ← add(Lneighbors)
   f lag ← f lag + 1
   for k ← 0 to len(lp) do
      for p ← 0 to len(lp[k]) do
         if b[Lp[k][p][0], Lp[k][p][1]] == 0
            then [Lp[k][p][0],Lp[k][p][1]]=neighbors[f lag]
         end if
      end for
   end for
end for
```

Algorithm 2: Function BuildComplex(B)

Maybe some values in the cubical complex will be equals or else extremely huge, in this case on can re-apply the transformations

$$B[i, j] \leftarrow B[i, j] + (i+I.j) / (\alpha.I.J)$$
$$B[i, j] \leftarrow B[i, j] / \max B[i, j]$$

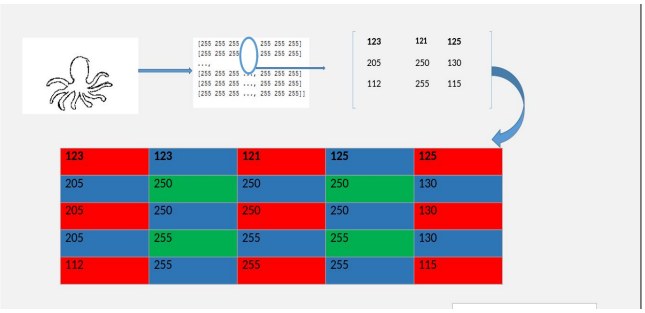


Figure 12: An example of a cubical complex computed from a digital image

3.2 Step 2: Build the boundary matrix.

This is the crucial and important stage. The first step is to build a filtration. We will follow this natural ordering :

- K 0 := □;
- K 1 := the 2-cube of high value and all its faces;
- K 2 := the next 2-cube and all its faces;
- Once all the 2-cubes are swept, we pass to the 1-cubes.

The boundary matrix should be a (N × N)-array, where N is the number of all different values that appear in the cubical complex. It should encode the evolution of the filtration here above. Thus in the 0-th stage all values will be null. In the first stage, we put boundary matrix(i,j)=1, once the j-th cube is a neighbor of the i-th cube, where the filtration indices are assigned from higher to lower

```
Input: Cubical matrix C (each cube has faces named Bj
Output: Boundary matrix
for each cube C i of K do
   Column ←filtration index of C i
   for each cube B j in boundary of C i do
      row ← filtration index of B j
      boundaryMatrix(row,column) ← 1
   end for
end for
```

Algorithm 3: Function boundaryMatrix(Cubical complex C: matrix)

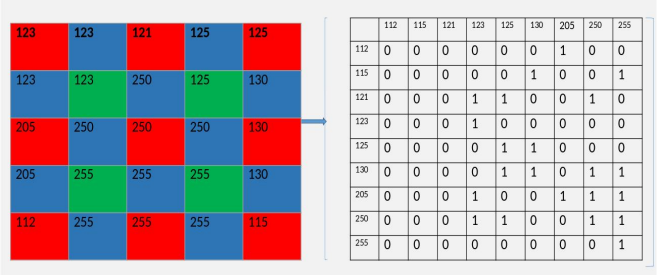


Figure 13: An example of a boundary matrix computed from a cubical complex.

3.3 Step 3: Reduce the boundary matrix

To get the reduced boundary matrix, we follow the [EH] approach, where the Smith normal form of the boundary matrix is used to record the face relationship between simplicies of dimension p and p − 1. Let low(j) be the row number of the lowest non-zero entry in column j, where we set low(j) = 0 if the entire column is zero. A matrix is called reduced when each row has at most one entry that is the lowest 1 for a column. To reduce our boundary matrix, we proceed from left to right by using only column additions (see Algorithm 4).

```
Input: Boundary matrix
Output: Reduced boundary matrix
for j ← 1 to n do
   while ∃j 0 < j with low(j 0 ) 6 = low(j) do
      add column j 0 to column j
   end while
end for
```

Algorithm 4: Smith reduction algorithm

4. CONCLUSION

Fast and robust feature extraction is crucial for many computer vision applications. The performance of shape descriptor or the efficiency of shape features is related to some essential properties, for example the location, the rotation and the scaling changing of the shape. Since our algorithm is based on the the pixel coordinates, which it never loses thanks to its ability to track it at any time in the process, then such essential properties are not affect the extracted features which still as robust as possible against noise.

The classical hurdle is that when we are in a real situation with huge amount of data, not only, algorithms can not be run on devices with limited computational capabilities but also we lose a lot of information hidden in details and they are not invariant to significant transformations and sensitive to clutter and occlusion. That is what create a growing need for local descriptors that even if they are not faster in computing but more efficient, and yet exhibiting good accuracy. Based on these features we get the shape descriptor whose it is performance is judged from the accuracy of the result and the computation time. Most of the time either we have a great accuracy but very slow computing time or the contrary. This because of the complexity of the data which includes situations in which it is noisy, high-dimensional, and/or incomplete. The use of clustering techniques and other ideas from areas such as computer science, machine learning, and uncertainty quantification along with mathematical and statistical models are often very useful for data analysis. One way to perform our algorithm may be the distributed computation of persistent homology (as stated in [BKR]). Once this done, a direct application may be :

- Consider an universal database of images;
- Consider two almost similar images, produce their codebare, compute their Bottleneck distance, which should be roughly small;
- Consider two images clearly very different, produce their codebare

REFERENCES

1. Alvin S. Alon, Rufo I. Marasigan, Jr. , Jennalyn G. Nicolas- Mindoro and Cherry D. Casuat. **An Image Processing Approach of Multiple Eggs' Quality Inspection.** *International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, Vol. 8, No.6, pp. 2794–2799, November–December 2019. https://doi.org/10.30534/ijatcse/2019/18862019
2. U. Bauer, M. Kerber, and J. Reininghaus. **Distributed Computation of Persistent Homology**, in *Proc. The Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX14)*, 2014, pp. 31-38. https://doi.org/10.1137/1.9781611973198.4
3. H. Edelsbrunner, J. Harer. **Persistent homology: a survey**. *Contemporary mathematics*, Vol. 453, pp. 257-282, 2008. https://doi.org/10.1090/conm/453/08802
4. R. Gonzalez-Diaz, M.-J. Jimenez, B. Medrano. **Spatiotemporal Barcodes for Image**. *Sequence Analysis, Combinatorial Image Analysis*, pp 61-70, 2015. https://doi.org/10.1007/978-3-319-26145-4_5
5. A. Hatcher. ***Algebraic topology.*** Cambridge University Press, 1993.
6. M. Heikkilä, M. Pietikäinen, C. Schmid, **Description of interest regions with local binary patterns**, *Pattern Recognition*, Vol. 42, pp. 425-436, 2009. https://doi.org/10.1016/j.patcog.2008.08.014
7. J. R. Munkres. **Elements of algebraic topology.** Persues Books Publishing, 1984.
8. V. Robins, A. P. Sheppard, P. J. Wood. **Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images.** *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, pp. 1-14, 2011. https://doi.org/10.1109/TPAMI.2011.95
9. Sumit Dhariwal and Sellappan Palaniappan. **An Efficient Approach for Semantic Image Classification using Normalization Method**. *International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, Vol. 8, No.4, pp. 1268 –1274, July –August 2019. https://doi.org/10.30534/ijatcse/2019/37842019
10. H. Wagner, C. Chen, E. Vucini. **Efficient computation of persistent homology for cubical data**. *Topological Methods in Data Analysis and Visualization*, pp. 91-106, 2012.