

National Graduate School of Engineering

Report

TOPOLOGICAL EVALUATION OF IMAGE TRANSFORMATIONS

February 4th of 2022

Sarah BROOD

sarah.brood@ecole.ensicaen.fr

Heithem DRIDI

heithem.dridi@ecole.ensicaen.fr

Tutor : Yukiko KENMOCHI

Tutor : Sébastien FOUREY



www.ensicaen.fr

Acknowledgement

We want to thank Yukiko Kenmochi and Sebastien Fourey for their help and support during all the project. It was a wonderful experience, and we spend really good times working on it. We learned so much thanks to them, whether it be notions for the project or parallel notions that we found very useful and fun.

Contents

Acknowledgement	2
Project presentation	5
1. Motivation & Problematic	5
2. Background	6
2.1. Transformation	6
2.2. Interpolation	7
3. Goal & Approach	8
Methodology	9
4. Transformation and interpolation	9
5. Discrete topology	9
6. Persistent homology	11
Management & tools	13
7. Project Management	13
7.1. Team	13
7.2. Organisation	13
8. Tools	14
8.1. DGtal	15
8.2. Gudhi	15
Project progress	17
9. Transformation and interpolation	17
9.1. Transformation	18
9.2. Interpolation	18
10. Discrete topology	20
11. Persistent homology	21
Conclusion	23
12. Project	23
13. Personal	23
14. Perspective	23

List of Figures

1	Example of transformation without interpolation Source : Combinatorial structure of rigid transformations in 2D digital images [2]	5
2	Example of transformation without interpolation	6
3	Square image rotated by $\frac{\pi}{4}$ with bicubic interpolation	7
4	2D and 3D examples for Betti numbers	9
5	Illustration for the 2D connectivity - Source : Wikipédia	10
6	Illustration for the 3D connectivity - Source [3]	10
7	Persistent homology - Source [6]	11
8	final timeline of the project	13
9	Gitlab board	14
10	Transformation & Interpolation Diagram	17
11	Base image	19
12	Nearest Neighbour, Bilinear and Bicubic interpolation of the base image	19
13	Example of computed topology	20
14	Example of computed topology	22
15	Example of computed topology	22

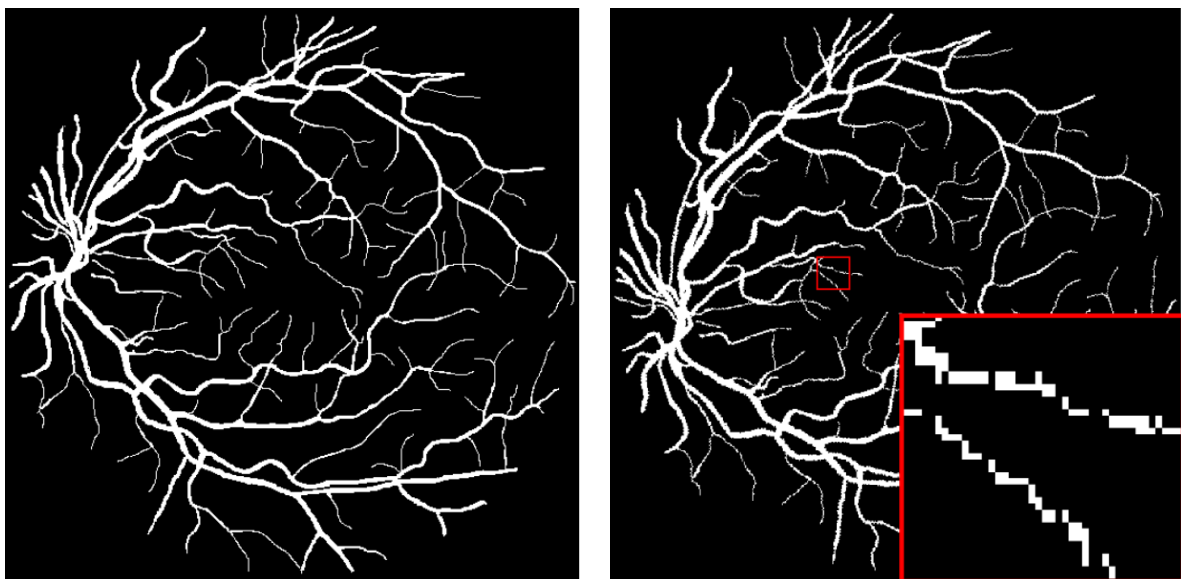
List of Tables

1	Existing transformations - Source :Wikipédia	6
2	Examples of interpolations	7

PROJECT PRESENTATION

1. Motivation & Problematic

Transformation errors on any object is disruptive because it produces a loss of data or meaning for it. For instance in the medical field, 2D or 3D representation are produced. These objects are intended to be transformed to be seen in many ways as possible. In the following example one can see in the first image a greyscale retina. The second represents the same image rotated with a small angle.



(a) Original image

(b) Image rotated with a small angle

Figure 1. Example of transformation without interpolation

Source : Combinatorial structure of rigid transformations in 2D digital images [2]

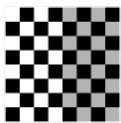
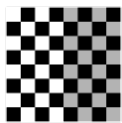
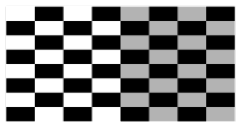

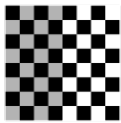

It is possible to see, within the red squares in the second image, parts that have been missing or deleted. This is problematic because it could lead to a medical misinterpretation while the issue is created by the previous transformation.

2. Background

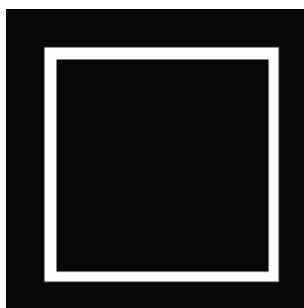
2.1. Transformation

There are many transformations available. These are represented in the following table.

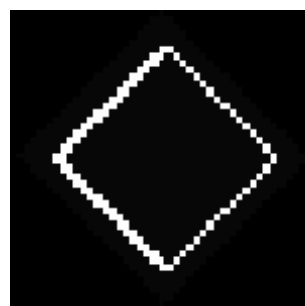
Table 1. Existing transformations - Source :Wikipédia

Transformation	Illustration	Transformation	Illustration
Identity		Translation	
Scale		Rotate	
Reflection		Shear	

To properly transform an object today, 2 components are used : a transformation and an interpolation. The transformation is used to change globally the object. But used alone, a "transformed only" image would look like the following example.



(a) Original image



(b) Image rotated by $\frac{\pi}{4}$





Figure 2. Example of transformation without interpolation

It can be seen that a transformation without interpolation, deteriorates a lot the shape. Indeed here, there are only as many white pixel on the result as the original one. Therefore in an attempt to counter this problem, we use interpolation. Just like the transformations, there are many interpolation available.

2.2. Interpolation

Interpolate consist in estimating a value from one or more other values, it is an approximation. The following table presents some of the available interpolations we discovered in the article [1] with its illustrations.

Table 2. Examples of interpolations

Interpolation	Illustration	Interpolation	Illustration
Nearest Neighbor		Bicubic	
Bilinear		Sinc	

Thanks to interpolation, it is now possible to create transformed images looking much smoother. The figure 3 represents the same manipulation as the figure 2 but with a bicubic interpolation.

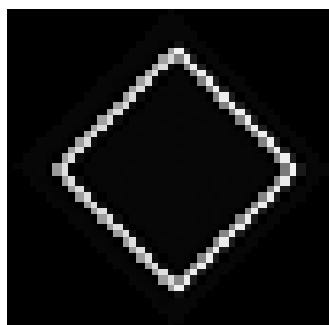


Figure 3. Square image rotated by $\frac{\pi}{4}$ with bicubic interpolation

The result is much more smoother and the original shape is better preserved. But as one can see it is not perfect, and with more finer-grained images like the retina one. And even with the interpolation, the transformation doesn't always keep all the useful data.

3. Goal & Approach

This why this project aims to create tools to quantify the errors that transformation on image (2D) or objects (3D) could produce. As 2D and 3D objects are represented with pixels and voxels, by transforming them we make deformations. Here we want to know if the object has kept its information, and its geometric characteristics.

To do so, we will use topology. This mathematical concept aims to observe the geometric properties of objects under continuous deformations.

The project is developed in C++ with chosen libraries to help and a appropriate architecture. Each part of the project correspond to several implemented tools.

METHODOLOGY

The project is divided in 3 main parts. All parts should work on 2 and 3 dimensions.

4. Transformation and interpolation

Is it natural that to quantify the error of an operator, it is inevitable to implement it. This first part, is a good introduction to the subject and a good way to understand the problem created by the operation (transformation+interpolation).

The objective is to implement tools to apply a chosen transformation with a chosen interpolation, those being totally independent.

5. Discrete topology

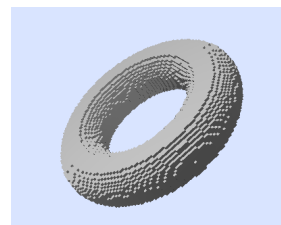
To compute the difference between the original and the transformed image, a first good verification is to compare the Betti numbers of the two. These numbers represent number of holes in a shape.

The k th Betti number represent the k -dimensional holes on a topological surface. Here we work in the 2nd and 3rd dimension so we have to look for the :

- b_0 : number of connected components
- b_1 : number of one-dimensional or "circular" holes
- b_2 : number of two-dimensional "voids" or cavities



(a) 2D image



(b) 3D Torus

Figure 4. 2D and 3D examples for Betti numbers

The sub-figure *a* has for Betti-numbers :

- $b_0 = 8$
- $b_1 = 2$

The sub-figure *b* has for Betti-numbers :

- $b_0 = 1$
- $b_1 = 2$
- $b_2 = 1$

To compute the Betti numbers, one has to choose the adjacency used. The adjacency correspond to the pixel connectivity for the foreground and the background. In 2D and 3D there 3 different connectivity : 4, 6 and 8 for the first one and 6, 18, 26 for the second one. In 2D : 4 connected means that the 4 pixels around the pixel will be its neighbour, 8 connected means the 8 around an so one. These two examples are illustrated in the following figure.

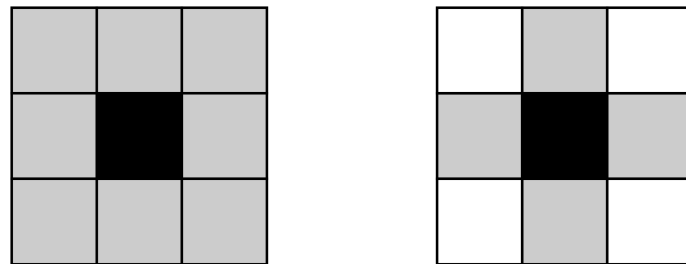


Figure 5. Illustration for the 2D connectivity - Source : Wikipédia

In 3D, the same logic is applied :

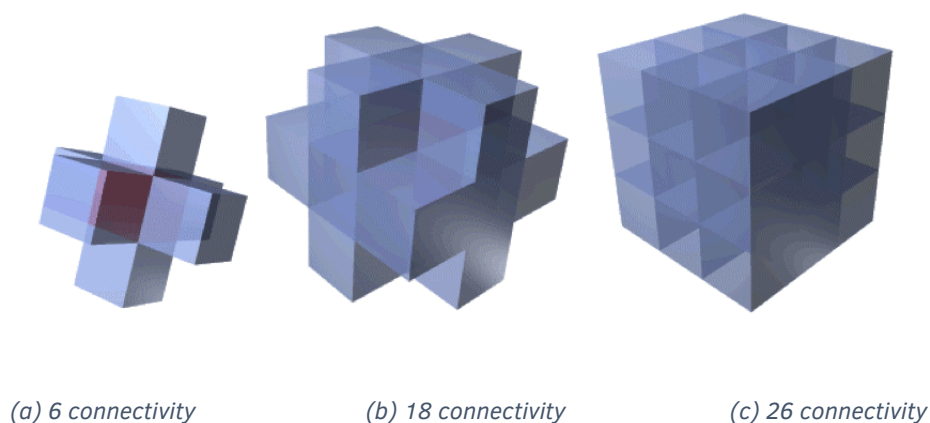


Figure 6. Illustration for the 3D connectivity - Source [3]

6. Persistent homology

Persistent homology introduces the scale notion between two objects. Indeed topology alone can't make the difference between a torus and a torus homeomorphic object, both have the same topological characteristics ($b_0 = 1, b_1 = 2, b_2 = 1$).

This notion is strongly related to the Morse Theory [4]: from a tame function $f : \mathcal{M} \rightarrow \mathbb{R}$ and its critical points t that are associated to the object, one can slice the object \mathcal{M} in a sub-object $\mathcal{M}_t = f^{-1} = (]-\infty, t])$. Depending on the object, the tame function can be a function running through the grey level (in 2D for example) or a function running through the height. (this time, for 3D) Consequently for every \mathcal{M}_t , it is possible to compute the Betti-numbers, enabling the creation of a persistence diagram. In the diagram, every point is a persistence interval, which means a birth-death pair, of a characteristic. The more a point is closer to the diagonal, the less it persists. On the other hand, a point far from the diagonal is a point which the persistence is bigger. As a result, point being too close to the diagonal can be considered as noise.

Persistent homology is stable if two objects are close [5]. Therefore, for \mathcal{M} and \mathcal{M}' its transformed version, the two objects will have similar diagrams if the transformation applied to \mathcal{M} doesn't deform too much the object. To define the similarity, we introduce the Bottleneck distance as : $d_B(\mathcal{M}, \mathcal{M}') = \inf_{\gamma} \sup_{m \in \mathcal{M}} \|m - \gamma(m)\|_{\infty}$ where $\gamma : \mathcal{M} \rightarrow \mathcal{M}'$ is a bijection. The more d_B is little, the more the objects are similar and there are few errors because of the transformation.

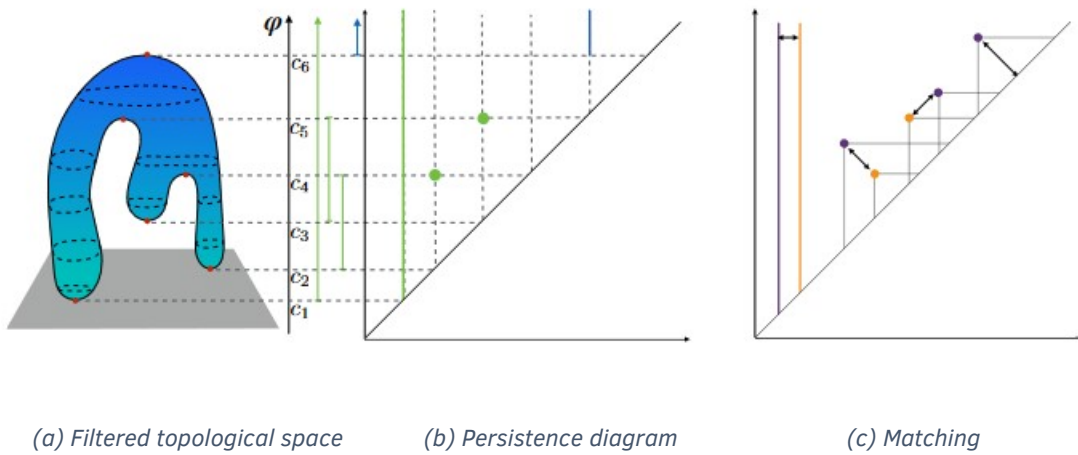


Figure 7. Persistent homology - Source [6]

MANAGEMENT & TOOLS

7. Project Management

7.1. Team

Our team is a pair so we kept the project management simple. We divided the work in function of our preferences. But for the biggest part of the project we worked together, to be sure we had understood all the concepts and then implement it. We tried to create an architecture as clean as possible and reviewed it many times.

In general we both implement the code, try to understand the concepts, make debugging and write documentation.

7.2. Organisation

7.2.1. Timeline

We didn't create a timeline for our project. As it is divided in 3 parts we could easily know if there was a problem or not. The first part is the smaller part, but for each part a time to learn and understand all the new notions has to be taken. We have reached the last part of the project at the end of December. In general, once we understood the notions and had a first implementation one of the two of us started the researches for the next part.

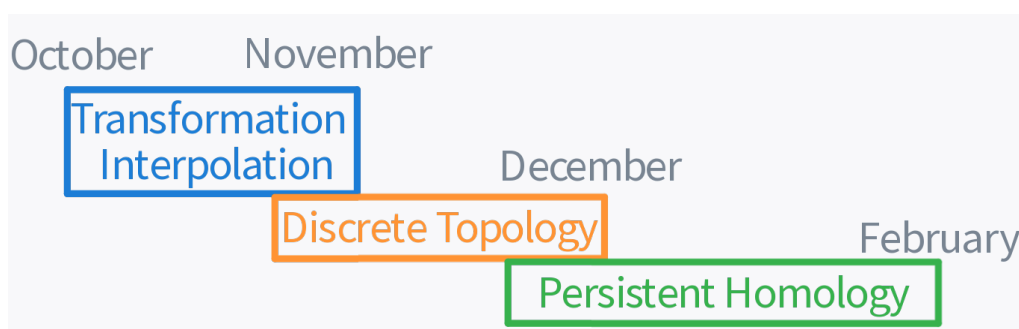


Figure 8. final timeline of the project

7.2.2. Meetings

Each week a meeting is organized with our tutors Mrs Kenmochi and Mr Fourrey. For the most part of the project we met at the GREYC laboratory. Sometimes we use Zoom or BigBlueButton to meet online.

These meetings are divided in 2 times. Firstly, we make a point on the project progress. There we ask question to understand better some notions or get further explanations. We present what we have made during the day or the last day. In a second time, we adjust the future objectives, and the task for the day and/or the following one.

7.2.3. Reports

At the end of each meeting, we make a report in Markdown. This report allows everyone to be sure we all understand the same things. The parts of the reports are the following ones :

- Issues discussed
- Tasks done
- Next goals

7.2.4. Gitlab

The project can be found on the school gitlab, here. For each task we create an issue. We use the issue board to manage our progress.

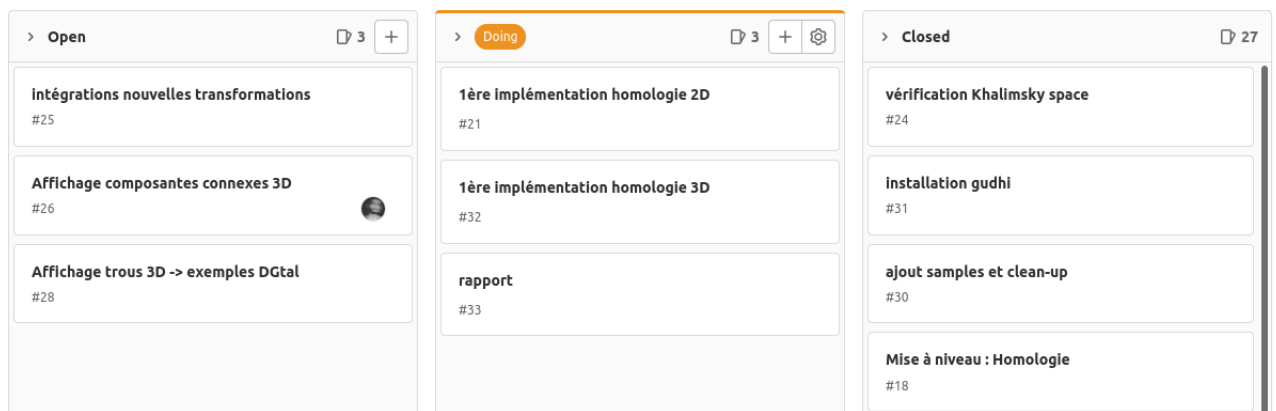


Figure 9. Gitlab board

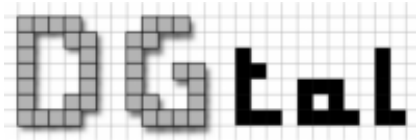
From each issue is created a branch that is merged after a pair review.

8. Tools

Our project is developed in C++ and uses two main libraries : DGtal and Gudhi. We chose those the libraries by looking for tools that could help us to answer each part of the project.

We chose to develop the project in C++ because it is one of the language we learned at school and because we found easier to create clean architecture with it.

8.1. DGtal



DGtal¹ is a collaborative project that aims at providing tools for digital geometry programming. It is a open-source library that we chose to use in C++.

It provides adapted data structures and algorithms for digital topology that is what we are looking for. This library is used during the entire project. It also offers many tools and examples to output or display the results.

8.2. Gudhi



Gudhi² Understanding for Higher Dimensions. This open-source library is made for Topological Data Analysis and Higher Dimensional Geometry Understanding. We tried for a long moment to use it, but unfortunately the bound to create between DGtal and Gudhi was too complicated to create as the two libraries use completely different types of data representation and doesn't read the same files.

Finally, in the last part of the project, after putting aside Gudhi, and developping our own tool for persistent homology. We used the feature to compute the Bottleneck distance of two persistent diagrams.

¹DGtal documentation can be found here and its sources on github

²Gudhi documentation can be found here, and its sources on github

PROJECT PROGRESS

Even if our code works for 2D and 3D, we decided to only demonstrate 2D examples in the report, as it is much easier to describe the problems, and to show the computed characteristics.

9. Transformation and interpolation

The first part of the project is the creation of the basic tools : load an image, make a transformation and an interpolation. For this part we used DGtal. The following figure shows the final architecture for the transformation and interpolation part. The two parts are used within the ImageTransformation class, that can use any affine transformation (2D or 3D) and any interpolation.

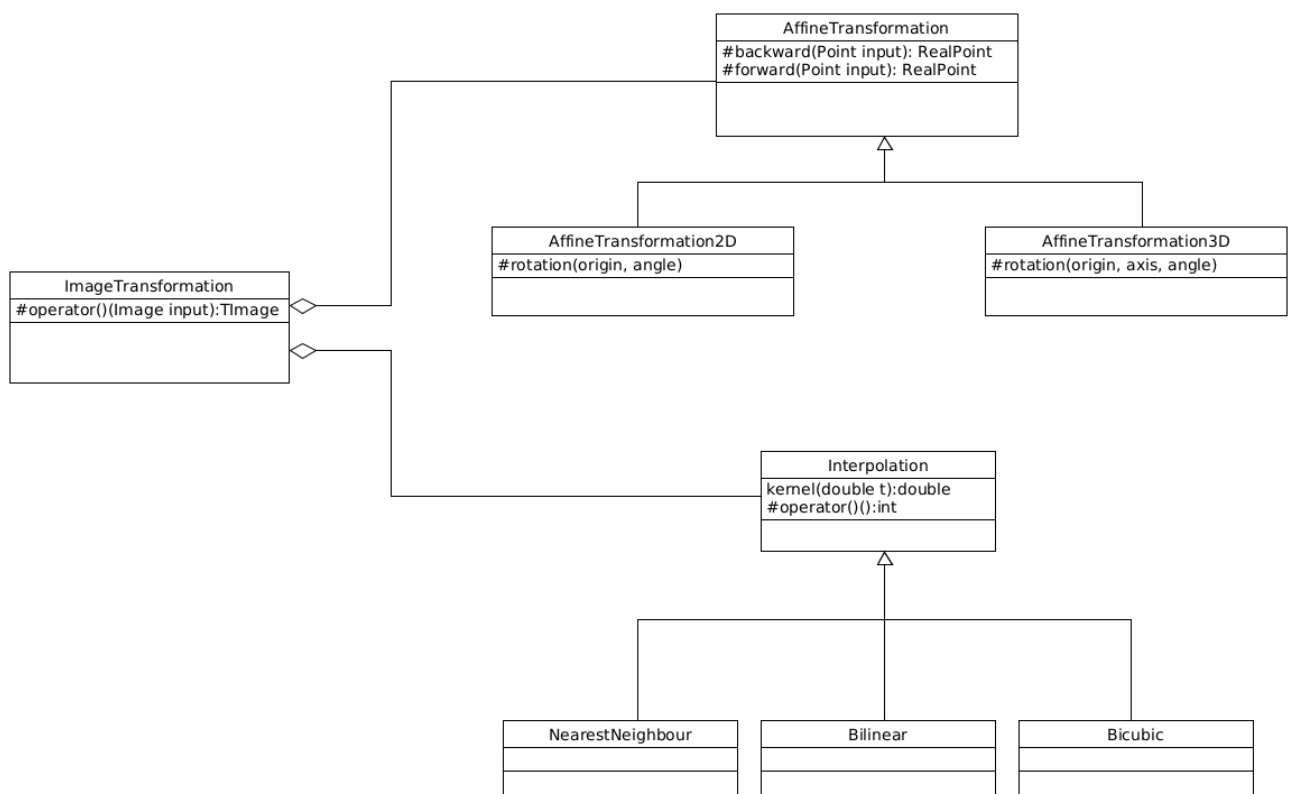


Figure 10. Transformation & Interpolation Diagram

9.1. Transformation

The transformation used is the backward transformation. Backward transformation iterates over each pixel of the output image and uses the inverse transformation to determine the position in the input image. This way to compute transformation permits to not create holes, while forward transformation does. Forward transformation iterates over each pixel of the input image, computes new coordinates for it, and copies its value to the new location.

We implemented both of the transformation to have the opportunity to play with both and see their differences. The implementation wasn't so difficult but we had to understand `DGtal` and all its related types. Backward transformation was new to us, so we took some time to understand the notion.

Each transformation is implemented thanks to its transformation matrix that we found in the OpenCV documentation³ for the 2D transformations and on Wikipedia⁴ for the 3D ones. For example a 2D rotation matrix is represented as follow :

$$\begin{bmatrix} \alpha & \beta & (1 - \alpha) \cdot \text{center.x} - \beta \cdot \text{center.y} \\ -\beta & \alpha & \beta \cdot \text{center.x} + (1 - \alpha) \cdot \text{center.y} \end{bmatrix}$$

We only implemented one transformation : the rotation. At first we chose this one because it is the most visual one but finally we didn't take the time to implement another one even if thanks to the architecture it is really easy to add one.

9.2. Interpolation

To understand interpolation we mainly used the article *Linear Methods for Image Interpolation* by Pascal Getreuer.[1] This article presents the use of interpolations and the different interpolations that exist.

For each interpolation we implemented we created a function that returned the pixel value computed by the chosen interpolation kernel. An interpolation kernel is only a function that smooth the pixel value. For example the bicubic interpolation can be represented as :

$$K_1(t) = \begin{cases} (\alpha + 2)|t|^3 - (\alpha + 3)|t|^2 + 1 & \text{if } |t| \leq 1 \\ \alpha|t|^3 - 5\alpha|t|^2 + 8\alpha|t| - 4\alpha & \text{if } 1 < |t| < 2 \\ 0 & \text{otherwise} \end{cases}$$

We implemented the following interpolations :

³Geometric Image Transformations in the OpenCV documentation

⁴Transformation Matrix on Wikipedia

- Nearest Neighbour
- Bilinear
- Bicubic

We had problems on the long term with the bicubic interpolation because there were values under and over threshold values (negative values or over 255 for grayscale for example) that were computed.

With a simple shape as a plain circle, we can already see differences between interpolations in 2D:

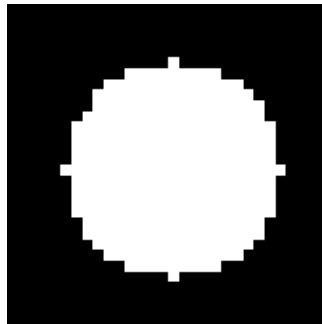


Figure 11. Base image

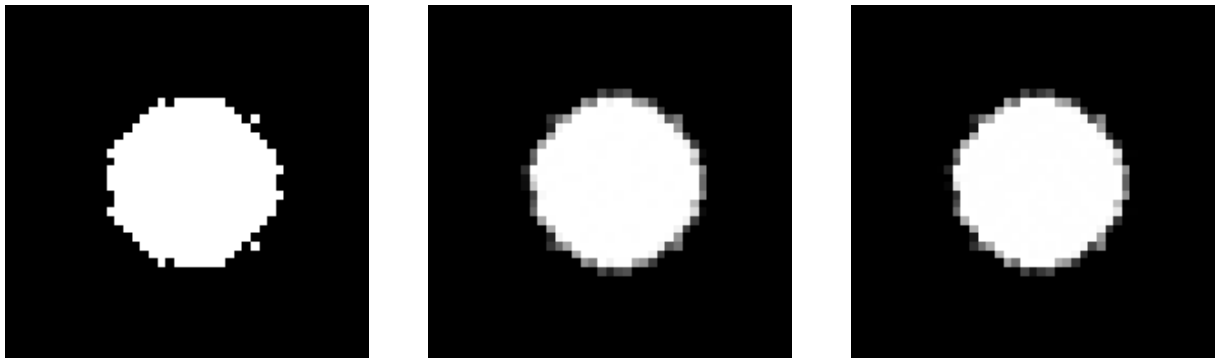


Figure 12. Nearest Neighbour, Bilinear and Bicubic interpolation of the base image

This is why by quantifying the errors between two transformed images, we could choose the most adapted interpolation to the situation.

For this part we also implemented a 3D viewer : to control the results obtained, we needed to visualize them. However, there are very few 3D viewers available and even fewer for the format we use. So we created our own viewer from DGtal.

This first part has given us the knowledge to understand the problem of errors in transformations. Indeed one can already see that the result examples provide a first sight of image errors that would be more frequent on bigger images.

10. Discrete topology

The Discrete topology part uses DGtal to compute the Betti numbers.

We created a Topology class that can take in parameter a DGtal representation of the image or the 3D object and can return its Betti numbers. Its architecture is represented in the following figure.

Betti-numbers are computed as follow : In 2D : $\{b_0 = \text{number of components of the foreground}, b_1 = \text{number of components of the background} - 1\}$. In 3D : we first compute the Euler Characteristic thanks to DGtal, then we compute the numbers as follow : $\{b_0 = \text{number of components of the foreground}, b_1 = \chi + b_0 + b_2\}$ and $\{b_2 = \text{number of components of the background} - 1\}$

To illustrate the change between an original image and its transformation, to use boards, a feature from DGtal. The board allows to bring to the fore the new holes or the new shape of the connected components.

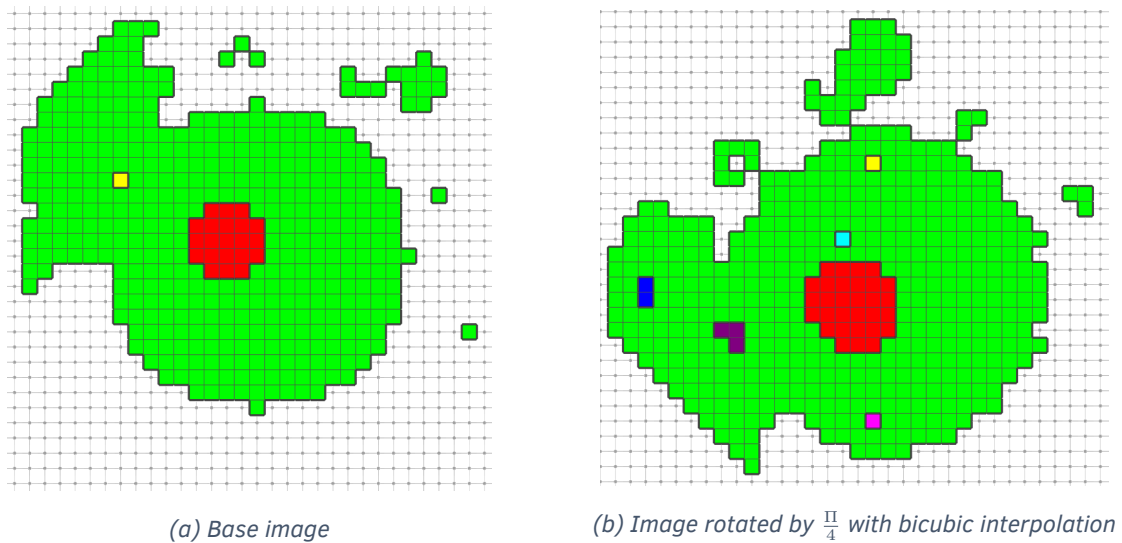


Figure 13. Example of computed topology

The program also returns the Betti numbers for each image: the sub-figure *a* has for Betti-numbers: $b_0 = 8$ and $b_1 = 2$, and the sub-figure *b* has for Betti-numbers: $b_0 = 5, b_1 = 6$. Here in green is represented the connected components. The other colors represent the holes. So we can see thanks to the numbers and visually that the transformation has damaged the image.

11. Persistent homology

We took some time to understand what was persistent homology. We made a good use of the video from the GT GDMM days by Alexandra Bac [7] which presents algebraic topology and its applications.

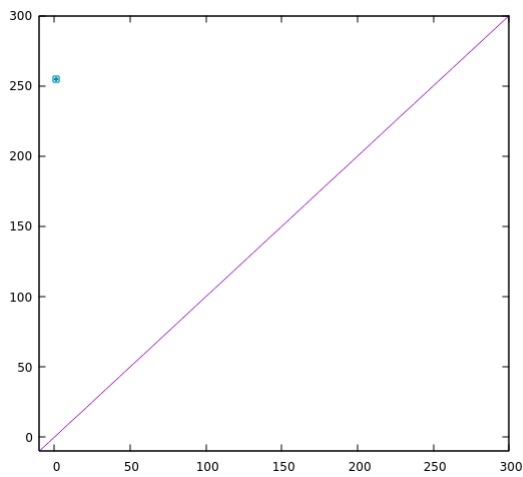
The first idea we had to create tools for persistent homology was to use Gudhi and its tools. Unfortunately, as we said before, the link between the library and DGtal which our project is based on, was too complicated. Furthermore the installation of Gudhi sources was pretty heavy. After a long time wasted on trying to find ways to use the library, and trying to understand its features, we stopped the idea to use it.

We had the discussion about creating our own tool to compute persistent diagrams, so we gave it a try. And it worked ! The computation itself was not too complicated as we think we well understood the notions.

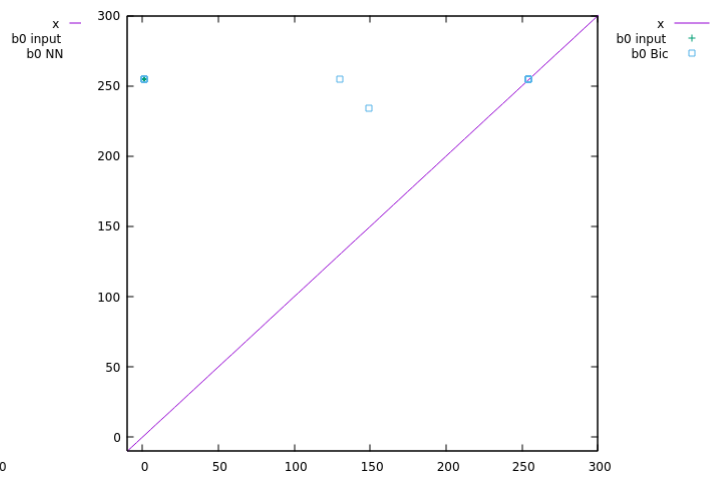
Firstly we take a tame function, we had the choice to either take a height function or a grayscale function. Given that two persistent diagrams have to have the same axis to be compared, we decided to implement in the first instance the grayscale function. To avoid to iterate over all the range of values (between 1 and 254), we retrieve only the image range values and iterate over it. The points within this range will be the critical points. To this extent for every iteration, we can compute the topological characteristics for every critical point and finally make a monitoring of their persistence.

To this point, we didn't use any new library. We wanted to compute the Bottleneck distance by coding ourselves, but due to lack of time we chose to look for off-the-shelf solution. That's why, we finally chose to use Gudhi. We only use it to compute the Bottleneck distance. But in a future, it would nice to compute it within our project. However Gudhi didn't provide a perfect match with our solution to draw a Bottleneck diagram. This is why, we only use 2 persistent diagrams and their Bottleneck distance.

As a result we can compare the Nearest Neighbor and the Bicubic interpolation we have implemented with the same transformation. We used the same picture as the figure 13 and the same transformation :



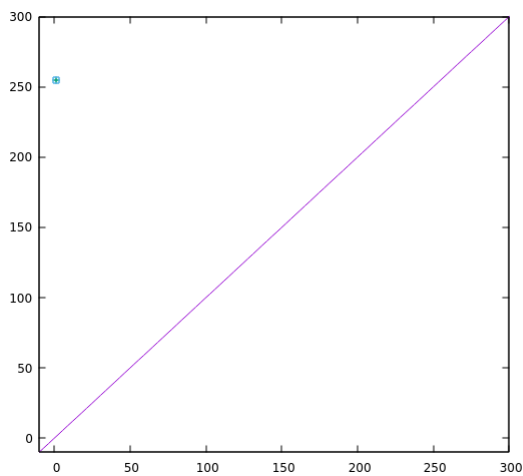
(a) NN interpolation persistent diagram for b_0



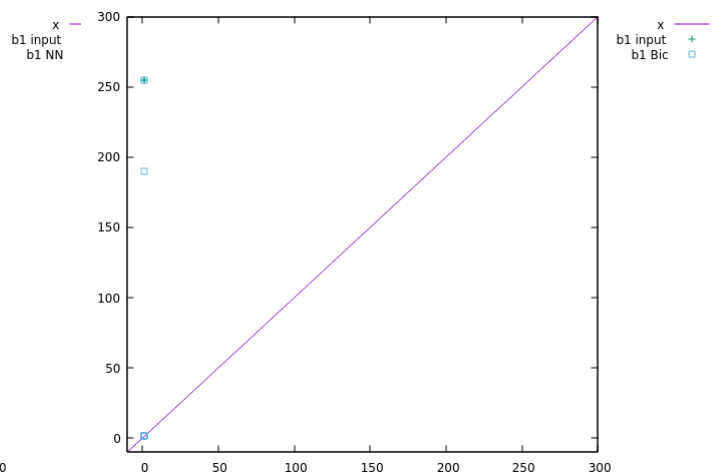
(b) Bicubic interpolation persistent diagram for b_0

Figure 14. Example of computed topology

Here the Bottleneck distance for b_0 for the nearest neighbor interpolation = 127 and for the bicubic interpolation = 127. So we can say the number connected component doesn't change with any interpolation.



(a) NN interpolation persistent diagram for b_1



(b) Bicubic interpolation persistent diagram for b_1

Figure 15. Example of computed topology

But the two distances are different for b_1 indeed, for the nearest neighbor the distance is $= 2.2e-308$ and for the bicubic interpolation the distance is $= 65$. As a result for this example, the nearest neighbor interpolation seems to be the interpolation that loses the less data on the shape of the object.

CONCLUSION

12. Project

To conclude the project we are grateful to our tutors for letting us do this project. On the three parts of the project, the first two seems to work really fine. The first part is adaptable and usable for many purpose. The topology part is clean and has many tools to display the results which is a problematic we encountered during the project as we better discovered topology. The last part was pretty problematic, and hard to understand and we have been stuck for a long time on the library problem. But we are happy to have results and to have understand the concepts !

We think that choosing C++ may have been a choice that made us spend to much time on programming instead of developing more the features and the project. The compromise between understand notions and create well implemented tools was complicated because of the time we had. But we think that providing source code is pretty good for the future of the project. The project seems to have a great start, we find that we created good foundations to allow other people to continue it.

13. Personal

In a more personal way, we think we learned a lot during this project. We are happy of what we created even with a project done by 2 persons, we made a good team. The knowledge we gathered during this year cannot be forgotten, and we have find really entertaining trying to answer to a real and visual problematic.

14. Perspective

This project has to be finished and polished. More over we know that there are many others topological characteristics that could be compute on 2D and 3D objects. The 3D part could be further developed, as the concepts were new to us, we only tried to adapted as much as we could the 2D part to 3D. It would also be interesting to create tools to see topological features for 3D, like the board for 2D.

BIBLIOGRAPHY

- [1] Pascal Getreuer (2011), Linear Methods for Image Interpolation, Image Processing On Line, 1 (2011), pp. 238–259. https://doi.org/10.5201/ipol.2011.g_lmii
- [2] Phuc Ngo, Yukiko Kenmochi, Nicolas Passat, Hugues Talbot, Combinatorial structure of rigid transformations in 2D digital images, Computer Vision and Image Understanding, Volume 117, Issue 4, 2013, pp. 393–408, ISSN 1077-3142, <https://doi.org/10.1016/j.cviu.2012.08.014>.
- [3] Tankyevych, Olena. (2010). Filtering of thin objects : applications to vascular image analysis.
- [4] Edelsbrunner, Herbert Harer, John. (2008). Persistent homology—a survey. Discrete Computational Geometry - DCG. 453. 10.1090/conm/453/08802.
- [5] Cohen-Steiner, D., Edelsbrunner, H. Harer, J. Stability of Persistence Diagrams. Discrete Comput Geom 37, 103–120 (2007). <https://doi.org/10.1007/s00454-006-1276-5>
- [6] Mattia G. Bergomi, Patrizio Frosini, Daniela Giorgi, Nicola Quercioli. Towards a topological-geometrical theory of group equivariant non-expansive operators for data analysis and machine learning. CoRR abs/1812.11832 (2018). <http://arxiv.org/abs/1812.11832>
- [7] GDMM 2021 : Un rapide panorama de la topologie algébrique et ses applications - Alexandra Bac - on Youtube : <https://www.youtube.com/watch?v=DZVbGs9NIRU>



National Graduate School of Engineering

6 boulevard Maréchal Juin, CS 4505

14050 CAEN cedex 04

