UCLouvain

École polytechnique de Louvain

Characterization of violins : a digital tool at the service of organology

Author: **Renaud Lothaire**
Supervisors: **Anne-Emmanuelle Ceulemans**, **Paul Fisette**, **François Glineur**
Reader: **Vincent Wertz**
Academic year 2018–2019
Master [120] in Mathematical Engineering

Abstract

The instruments of the violin family differ in size and morphology, which vary according to epoch and region. This diversity is difficult to study because most of the old instruments that have come down to us have been recut to match the dimensions of the new models in use. Anomalies can therefore appear in the vault, but they are difficult to detect with the naked eye. This project aims to develop a digital tool in the programming language *Python* to objectively quantify these singularities and geometric anomalies of ancient violins and violas. All this is based on a set of digital representations of the three-dimensional geometry of these instruments, which come essentially from the MIM (Musical Instruments Museum, Brussels). The challenges are to enable organologists to better understand the original morphology of members of the violin family under the Ancient Regime, and to evaluate the authenticity of ancient instruments. This research will also help to guide luthiers and musicians who are often involved in historically informed interpretation practices.

Acknowledgements

I would first like to thank my thesis advisers Pr. Anne-Emmanuelle Ceulemans of the SSH/FIAL at UCLouvain and Pr. Paul Fisette of the SST/EPL at UCLouvain. The door to their offices was always open whenever I ran into a trouble spot or had a question about my research or writing. They consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge Pr. François Glineur of the SST/EPL at UCLouvain as the third advisor of this thesis, and I am grateful indebted to his for his very valuable comments on this thesis.

I would also like to thank the MIM (Musical Instruments Museum, Brussels) for allowing us to scan instruments from their museum. Without their help and participation, this project would never have been able to achieve such promising results.

I would also like to acknowledge the experts who were involved in the validation survey for this research project: Pr. Emmanuel Coche (Cliniques universitaires Saint-Luc), Pr. Etienne Danse (Cliniques universitaires Saint-Luc) and their team, who agreed to scan the violins. Mr. Xavier Bollen of the SST/EPL at UCLouvain who kindly agreed to make the support to scan the violins. Without their passionate participation and input, the validation survey could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents, to my brother and to my girlfriend for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Lothaire Renaud

Contents

Introduction

At the end of the 18th and during the 19th century, many ancient instruments of the violin family were recut to match new standard shapes used in lutherie.
Initially, these instruments were larger and had been recut a few centimeters or millimeters to reach the adequate dimension. When a violin is recut, its soundboard, neck, ribs and back are unglued. Usually, the neck will be replaced, and therefore will not be studied in this master's proof. The upper and lower parts of the soundboard and back are recut (see figure 1), but not their sides, or at least to a much lesser extent. The C-bouts do not allow any recutting, as this would enlarge them instead of reducing them. The art of the violin maker is to provide a smooth transition from the recut area to the original contour of the instrument.

Figure 1: Recutting to be characterized

Over time violin makers and musicologists tried to identify some specific characteristics of these recut violins. Thanks to their knowledge of those instruments, some scholars succeeded in putting forward anomalies that might be due to the reduced size of the soundbox. Their observations, however, are problematic from a scientific point of view, as they do not rely on mathematical deduction. Unfortunately, no computer tool currently exists that would allow us to formalize these anomalies from a quantitative point of view. And yet the stakes are high : lots of ancient instruments of the violin family, sometimes attributed to well-known luthiers, are recut. These instruments may reach staggering prices on the market, but they cannot be considered as faithful witnesses of past building practices. A computer tool that analyzes violin shapes could be useful to give a rational approach to tackle that problem.

This Master's thesis is focuses on this problem. The aim is to create a digital tool that classifies and characterizes violins. These analyses must be achieved based on purely individualized criteria, i.e. taking into account to the own form of each instrument, because violin, viola, violoncello vaults can be widely different.
This dissertation follows the research done by Laetitia Motte [10] who used a photography technique to reconstruct the shape of the instruments and analyze them. Her work offers valuable research results, but the method used was somewhat restrictive. Since then, a lot of precious violins (Musical Instruments Museum, Brussels (MIM)) have undergone a CT scan with high precision at Cliniques Universitaires Saint-Luc. Those scans were available into *DICOM* format that could be converted into *Python* data.

This Master's thesis presents a digital tool that was realized in the *Python* language during the year to characterize the external shape of those instruments. Using mathematical, numerical, computer and musicological tools, we manage to highlight a lot of geometrical attributes for a given violin/viola/violoncello.
We also intended to develop an IT tool that permits us to sort violins into categories. The idea was to give a violin CT scan to the program and to predict if the violin is of "non-recut" or "recut" type. Due to the limited number of available instruments, the artificial intelligence aspect was not realized in the frame of this dissertation.

The aim of this research is twofold. On the one hand, it will help organologists to better understand the original morphology of recut instruments of the violin family. On the other hand, it will enable them to assess the authenticity and shape of ancient instruments using quantitative criteria. Considering that the violin forms the basis of the occidental orchestra since the 17th century, we note that the scope of this research is extensive.

The text will be structured as follows.
In chapter 1 we will describe the historical context and make some links with the engineering approach. We will also present more information about recut violins and their musicological context around.
Chapter 2 will explain the procedures that we followed to obtain and treat DICOM data. That consists of a description of the data, an explanation of the functions used and a discussion about the robustness of the program.
Chapter 3 tackles the critical aspect of the violin positioning into the scanner. We will see that the manner used to submit the violin to a CT scan has significant consequences on the quality of the program. We will thus talk about problems encountered and solutions found.
Chapter 4 will present all graphical and numerical results we obtained through our program. Geometrical aspects that were analyzed here are : the culmination points and lowest points of the table, the layout formed by the inflection points of the soundboard and eventually some results about the parallelism of the spikes.
Chapter 5 will show the many results we have achieved with our Python program. We will also highlight the differences that may appear between different instruments.
At the end we will finish with a short conclusion.

Chapter 1

State of the question and positioning in relation to the state of the art

1.1 Historical context

During the 16th century, the violin was conceived as an instrument family of different sizes.
Nowadays, this family consists of three types of string instruments : violin, viola and violoncello [1] (The double bass is usually considered as a member of this family but does not intervene in this research).
Under the Ancient Regime, north of the Alps, the violin family differed both by the number of its members and by their dimensions that are variables depending on the regions and epochs. At this time no kind of standardization existed, as standardization appears mainly during the 19th century through the conservatories. Hence, a lot of diversity was present in the violin family. To illustrate this diversity, we could refer to [9] which argues that no French violins built under the Ancient Regime have been preserved in their original state. Some have been preserved for the Low Countries. But for both countries, most violins that have come to us have been recut.

From the end of the 18th century, a certain number of instruments have been recut to fit the standard dimensions that are still in use. The instruments' historical diversity is then particularly challenging to study. For example, a "taille" instrument from the 17th century could be recut to fit the dimensions of a viola and took this name attribution. It is the case for a lot of string instruments conserved at MIM [2] most of which were built in the Low Countries. It is possible that some of them were recut a few centimeters or

[1] In order to get more information about those three types of instruments we could consult [1] for violoncello [2] for viola and [3] for violin.

[2] Musical Instruments Museum, Brussels

millimeters and their original shape is therefore very difficult to retrace and to detect without computer help.

The main problem nowadays in order to check if an instrument has been recut or not, is that written sources commenting on this practice are particularly scarce. The diversity in violins in France has been better documented in the last few years but still, all kinds of erroneous information, myths and inaccuracies still circulate on that subject to this day. In addition, no existing computer tool is able to characterize the geometrical shape of the instruments. The only tool available so far to study such instruments is the human eye.
K. Moens, a musicologist who worked mainly at MIM, has written extensively on the topic of violins and especially on authenticity issues which arise frequently when dealing with instruments kept in public and private collections.
He is one of the scholars who relied on his experience to determine whether an instrument presented any anomaly. He studied a great number of violas through a careful visual examination to try and assess their initial dimensions [9].
For example, figure below, issued from [9], illustrates presumably recut violin and the hypothesis of its original dimensions by K.Moens :

Illustration of a presumably recut violin [9] : Height of the actual case : 35,4 cm (left) Height of the original case : 38 cm (right)

Figure 1.1: Recut violin, France, first half of 18th century (Antwerpen, Vleeshuismuseum, inv. 1967.001.204)

Karel Moens' hypotheses are not widely accepted, on the one hand because they

indirectly affect difficult financial issues. On the other hand because K.Moens struggles to demonstrate their validity with receivable scientific arguments. Across all his publications, he keeps making many assumptions but fails to demonstrate them convincingly.
All the mathematical and computing features developed in this text then take on their full meaning.

In order to better visualize the problem of recut violins we must look at figure 1.2 showing a cross section of the violin :

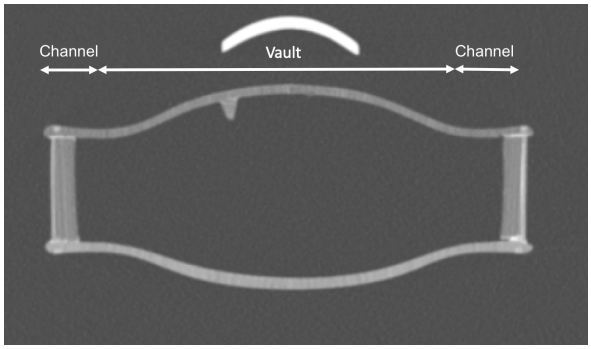


Figure 1.2: Information on a violin cross section

This illustration allows to understand the problem concretely.
We observe that the vaults on the soundboard and the back of the violin are circled by a channel. This is in fact a depression circling the vault itself.
When an instrument is recut, this channel recedes or disappears in the upper or lower part of the sound box. It is difficult to modify the central sides of the instruments and thus this channel is usually still present in these places. It could be that a luthier recut slightly on the sides, but it would make the instrument very misshapen, something that can be easily spotted. This conclusion is based on the characteristics of the construction of a violin [3].
As a result, it is important to study these anomalies to detect if a violin has been recut or not.

Looking at the channel is not the only way to detect anomalies.
Analyzing other geometrical characteristics can also be revealing, namely, the shape of the vault, the maximum zone of the table, the zone formed by the inflection points, the symmetry of the soundboard, etc. In the literature we saw that a certain number of papers have been written on those topics.
For example, an article from the Strad [11] addresses points of interest to violin and bow makers. Steven Sirr and John Waddle published a study on the central section of an

[3] In order to better visualize how a violin is made we advice [5]

instrument's arching : the longitudinal arch structure.
Other examples include a research done by the Chimei Museum (Taiwan). They attempted to develop a software for the optical recognition of violin soundboards by means of a good photography. This software was created in order to allow the classification of the soundboards into families, e.g. Stradivari model, Guarneri model, etc. But they ultimately gave up as the software could only be used to recognize the features of an individual violin but did not succeed in identifying the style of specific violin makers. So, ultimately the museum did not use nor sell the software.
A last example is the article published by Geerten Verberkmoes [12] that presents how medical imaging techniques could be exploited in order to compare violins. This publication provides a general idea of what is done currently. They used a MATLAB program in order to detect the superior and inferior surfaces of the top and back plates. By way of these surfaces, the plate elevation and thickness, as well as the local density could be highlighted.
Figure 1.3, as published in [12] p.162, illustrates a map-like image that is constructed with a color scale that defines the numerical value ranges.

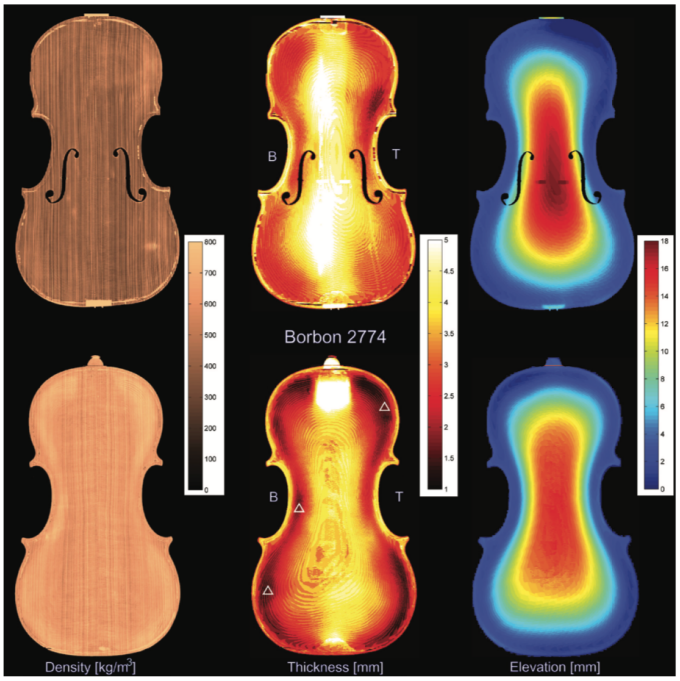


Figure 1.3: Density, thickness and elevation maps for the violin by Gaspar Borbon, MIM inv. no 2774. The symbol △ in the thickness map of the back indicates the places where confirmation measurements were performed with a magnetic thickness gauge (Hacklinger, type B, Germany) at areas of local minimum thickness. At the locations indicated by symbol △, a thickness of $1.7mm$ was measured.

1.2 Engineering approach

This dissertation aims to help luthiers, musicologists and museum curators to base their judgement on scientific criteria that are more objective than the mere visual appreciation of instruments. The added value of an engineering approach boils down to the fact that violin-making and expertise are mainly qualitative : starting from the conception of the violin, to the evaluation of its anomalies. Luthiers are craftsmen who manipulate their tools with great precision but seldom use equations. As engineers, we could quantify the problem and more specifically the channel, inflection points and maximum points. This is exactly what is presented in this master's thesis. We used *Python*, a computer language, in order to get some information about the instruments retrieved from available CT scan. We used some mathematical tools such as interpolation, approximation, splines, matrices, etc in order to reach the objective.

The program created to retrieve all the characteristics of a given instrument is made up of about 2800 lines of code. Starting from a DICOM CT scan to the point where it can provide all the desired characteristics, the program runs for approximately 17 minutes. All the details are presented in the appendix in the table B.1.
The methodology used in order to reach the objective may be summed up as shown in figure 1.4.

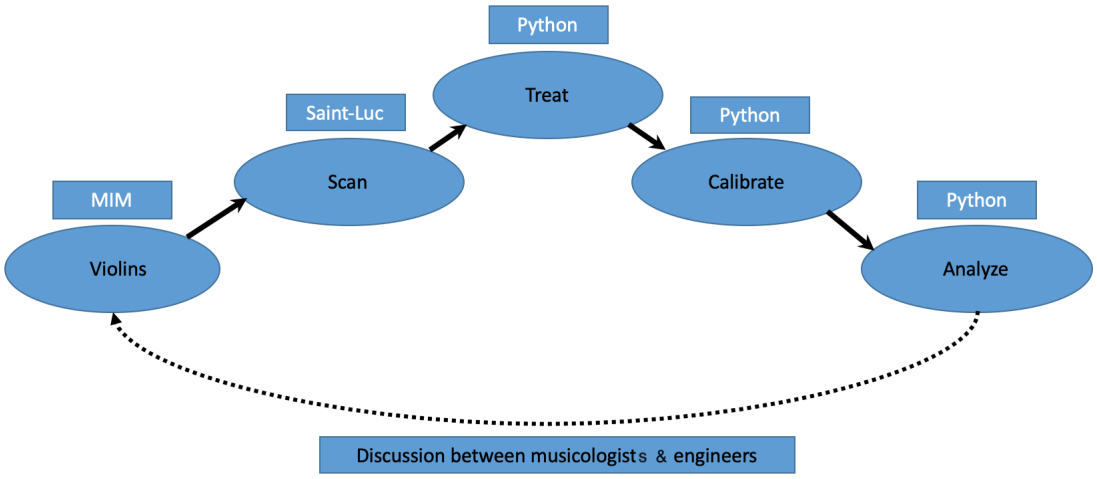


Figure 1.4: Diagram of the general methodology used during this master's thesis

The starting point was to find database of several violins to base the research on. A collaboration with the MIM made this possible.
Once a set of violins were made available, they were scanned thanks to the collaboration of Hopital Erasme (2013-2014) and the Cliniques universitaires Saint-Luc (2016-2019).

This is where the whole computer process began. First of all, it was necessary to process the files and convert them into a form that could be used in Python. Then we were able to calibrate the position of the instrument as explained in chapter 3, to finally analyze the instruments and extract their main characteristics as explained in chapter 4.
It is important to note that this whole computer process would not have been possible without the various discussions with musicologists like Mrs Ceulemans. These discussions made it possible, among other things, to better target expectations and to obtain essential feedback on IT results.

Chapter 2

Acquisition and processing of data

2.1 DICOM data description

The very first step of the master's thesis was to collect and process the violins' data.
The photography technique used by Laetitia Motte for her research [10] was somewhat restricted. The procedure was indeed quite difficult to implement, and the results were not very accurate for multiple reasons.
The quality of the pictures was not optimal due to the camera used and to the fact that violins are varnished. Professional lighting could solve the problems to some extents, but the right conditions were not reunited to obtain a reliable result.
The results, although probably the best we could draw from photographs, were not significant as seen in figure 2.1 issued from [10] :

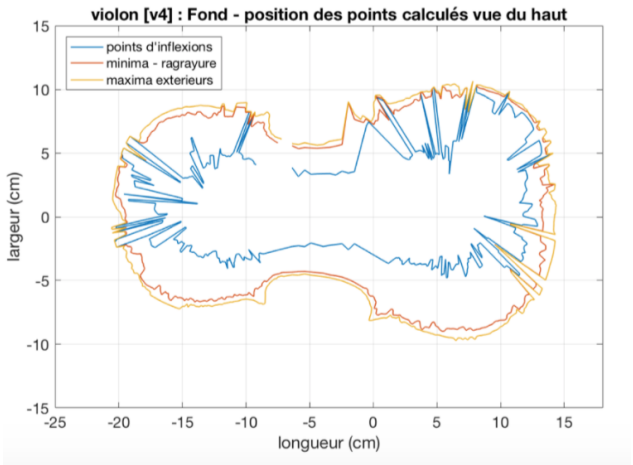


Figure 2.1: Some characteristics points of a given violin

Results obtained with that technique were probably the best we could draw from this

photography technique with restricted budget.
However, for luthiers and musicologists, precision is one of the most, if not the most important element to take into consideration when analyzing a violin. As a consequence, these results did not satisfy those who wished for more accurate graphs.
As a result, another technique to retrieve data was put forward. Since then a lot of valuable violins had been scanned in Cliniques Universitaires Saint-Luc with high degrees of precision. A few violins had been scanned previously at the Hôpital Erasme.
The Philips IQon Spectral CT scanner, used in Saint-Luc, is based on a technology that allows it to recognize high and low energy photons. This technique is based on the decomposition of the x-ray into different energies, which therefore allows each element of matter to be identified according to its response to a given energy.
The scanner has the following characteristics :

- Device UID : 1.3.46.670589.33.1.2200303521616
- Device Name : IQON-860010
- Device Manufacturer : Philips
- Device Model Name : IQon - Spectral CT
- Device Physical Location : UCL St Luc

The starting point of this research was then to collect all this data. Those scans were available under *DICOM (Digital Imaging and Communications in Medicine)* format. This format is very widespread in medicine. It is indeed the standard to manage and communicate medical imaging information. A DICOM scan can be converted into a file containing a 3D grid of pixels and will therefore be easily usable with our program.
The available dataset was formed by the following instruments [1]:

Number of instruments	Saint-Luc	Erasme	Saint-Luc II (explained in section 3.3)
Violin	2	5	7
Viola	4	0	4
Violoncello	2	2	0

Table 2.1: Number of instruments available under DICOM format

Before using those data sets, it was important to visualize them. That was made possible by using Radiant program [8] which is a PACS-DICOM viewer for medical images.

[1] In order to get information about the instruments, we will specify the inventory number from `www.carmentis.be`
For more information about luthiers we advice [6]

This application allowed us to visualize cross sections of the instruments in an easy way as can be seen on figure 2.2

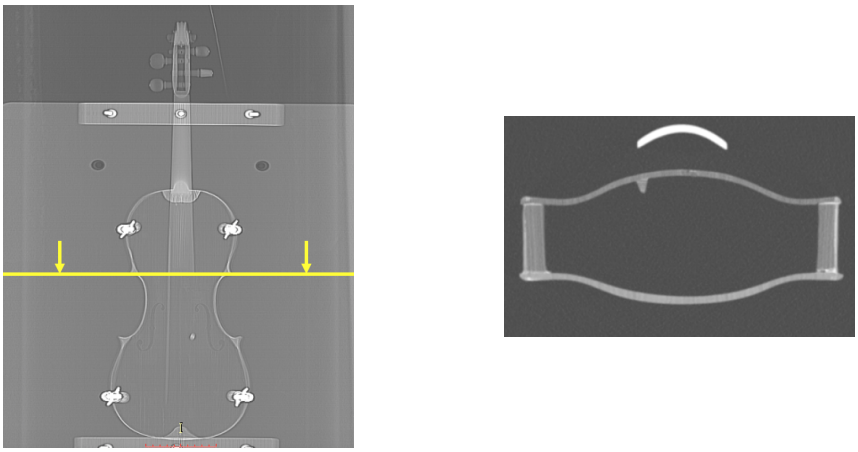


Figure 2.2: Longitudinal and transverse cut of a given instrument with Radiant viewer

This kind of image has a very high resolution and is more accurate than a photographical technique. For example, a scan consists of approximately 2000 transverse slices as shown on right figure 2.2 for $\approx 701mm$ of scanning length [2] which results in a cross-section every $0.35mm$.
A given cross section consists of a 512×512 grid of pixels with the degree of intensity indicated by a grey-scale. The size of such cross-section is $350mm$ by $350mm$ which results in a precision of one pixel every $0.68mm$.

The whole data set for a given violin is made of $2000 \times 512 \times 512$ values that define the intensity of each pixel. A solid work should then be done in order to treat and manipulate these data in the most efficient way. If it were not solid, the program would be unusable because of the time it would take to run.

All the pixel data under DICOM format were then converted to A 3D grid of pixels in *Python* in order to be used with our program. All these ingredients are detailed in section 2.3. The choice of the programming language was motivated by the possibility to allow institutions like MIM to use the program for free.

[2] The fact that a viola is larger than a violin is actually noticeable in the number of significant cuts we have in a given CT scan (i.e. those with an image representing the material). Indeed for a viola there are about 1800 images containing material, and for a violin about 1600 : with advanced precision this would mean that a violin measures about 57cm and an viola 63cm, which is the reality.

For all the analyses that follow, we fixed our conventions about the axes to be used from then onward as shown on figure 2.3 :

Figure 2.3: Cartesian directions chosen

2.2 Robustness and limitations of the program

The most important and difficult aspect of this thesis was to create a program that is robust. The ideal program is one to which we could give an input that is variable and that returns us an output without modifying anything in the code. That is nearly the case for the program that is presented in this thesis. The only thing is that a pre-process phase was needed in order to provide some characteristics of the violin to our program as shown on figure 2.4.

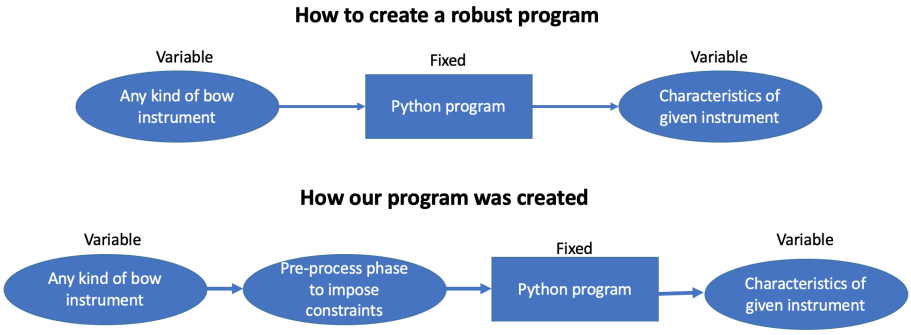


Figure 2.4: Scheme of a robust program

This pre-process phase is mainly due to the fact that violins and more generally

bowed instruments are formed by organic material that has gone through the times and has undergone some shape modifications.
This prior phase is also linked to the positioning of the violin in the scanner. Indeed, we encounter a lot of problems when dealing with points at the top and the bottom of the violin sound box. This will be discussed in further details in chapter 3.

In should be added that the scanning technique that we used cannot be repeated for a large quantity of instruments. We were indeed confronted with certain constraints to scan these instruments. It was necessary to have several Saint-Luc staff members present who agreed to work outside their working hours and free of charge, it was mandatory to analyze the instruments in advance and to pay insurance for the instruments,... This technique is therefore obviously not reproducible for a larger number of violins. For all these reasons, the program presented in this master proof cannot be applied as is to large collections of musical instruments, but its methodological principles could be adapted to 3D images available through scanning techniques differing from computed tomography.

2.3 Image treatment

After the presentation of the available DICOM data, the following step was to explain the transformation of those data into Python format. Once this task was completed, we needed to treat the images by denoising them. Indeed, they obviously contained a lot of noise around the violin body. All steps followed in order to transform a DICOM image into a denoised Python image are summarized in figure 2.5 and detailed hereafter.
From now on, if nothing is mentioned when detailing a function, it means that the corresponding function was created from scratch and it does not rely on pre-existing functions from the Python library or elsewhere. This was the case for approximately 95% of the functions presented in this thesis. As we mentioned in chapter 2, resources and works tackling this subject are indeed very scarce. It was thus very difficult to rely on previous works using this technique. Consequently nearly all functions and algorithms have been created from scratch.
We used two kind of existing resources

- The first available resource allowed us to convert DICOM data into Python data [7] and [13]. This kind of algorithm had already been created for medical imaging. We used this while creating the *GotScan* function which for that reason will not be developed into more details.

- The second resource we used is about image denoising : algorithms to detect and delete noise on images are widespread in many domains. As explained hereafter we used one existing function named *threshold otsu* available in the *skimage* Python library.

Just below we display the figure 2.5 which details the general morphology of our code. After that we will explain more precisely the role of each function.

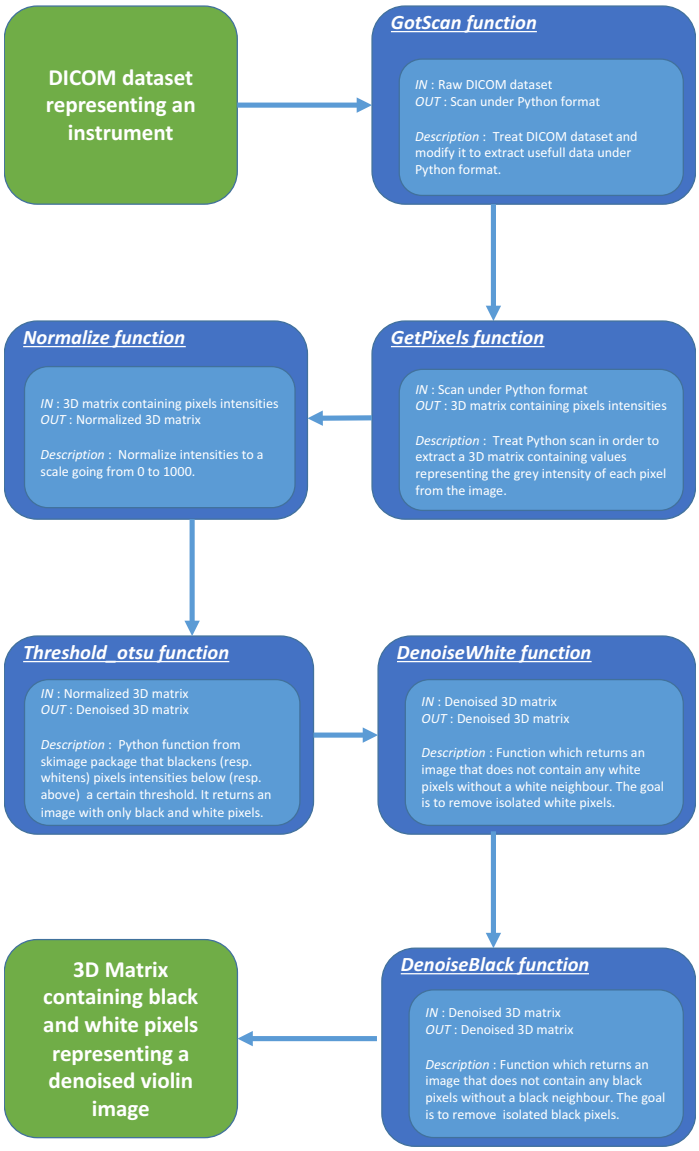


Figure 2.5: General methodology used via our Python program to produce a denoised image

In the light of the general methodology presented above, it is necessary to explain the body of each block from our figure 2.5.

We will proceed by first presenting the functions and their underlying algorithm in the form of a pseudo code. This is a way to describe an algorithm in near-natural language, without making references to any programming language.
In these pseudo codes, we will detail the data needed by the function to operate, the returned result, and the steps that were followed to obtain the results.

Let us begin by a very classical function that allows us to manipulate pixels in an easy way. This *Normalize* function described in algorithm 1 transforms our data in order to help us remember without difficulty that a 0 pixel value holds for black color and a 1000 pixel value holds for white color.

Algorithm 1 *Normalize* function

Data: *image* $\in \mathbb{R}^{512\times512\times n}$
This *image* should be an array of pixels intensity which have values $\in \mathbb{R}$
Result: *NormalizedImage* $\in \mathbb{R}^{512\times512\times n}$
This image is normalized in order to take pixel intensity values $\in [0, 1000]$
Algorithm :
$A = min(image)$
$B = max(image)$
$NormalizedImage = |\frac{image-A}{B-A} \times 1000|$
return *NormalizedImage*

Then let's explain the most important functions of the second step of image processing.
After using *threshold otsu* function [3], we obtain a 3D image that contains only black or white pixels, and thus a 3D matrix which contains only 0 or 1000 values. All pixels under a given threshold have been blackened and those above have been whitened. Pixels which are black correspond to the organic matter of the violin.
However, some pixels become black while they are not part of the instrument. This is due to the imperfect quality of the Python function *threshold otsu*.
DenoiseBlack and *DenoiseWhite* algorithms were created to deal with isolated black ot white pixels with no other same color pixels in their immediate "neighbourhood", what usually indicates they are encoded in the wrong color.

[3] Function which returns threshold value based on Otsu's method.
Parameters : **image** : (N, M) ndarray = Grayscale input image.
Returns : **threshold** : float = Upper threshold value, all pixels with an intensity higher than this value are assumed to be foreground.
For the instruments we used, the threshold value was approximately equal to 50 in most scans. This means that pixel intensities below 50 have been blackened and the ones above 50 have been whitened

One can see examples in figure 2.6 circled in red and orange.

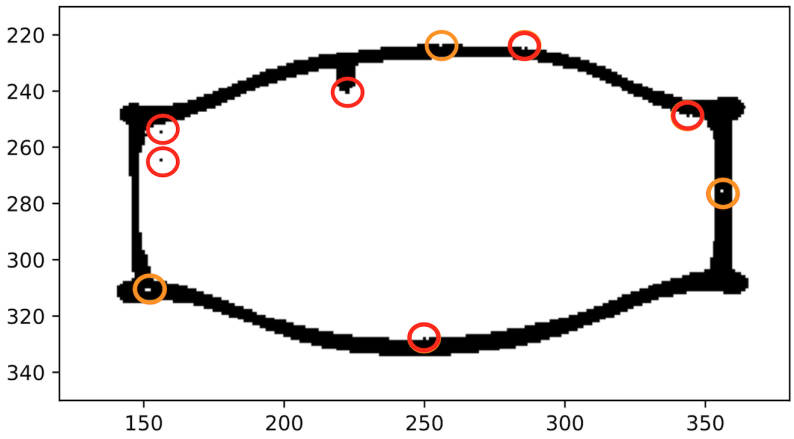
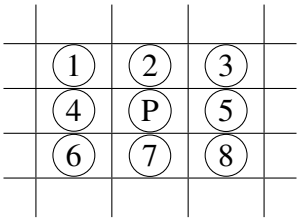


Figure 2.6: Pixels errors circled on a given cross-section of an instrument

We will now explain the notion of "neighbourhood" and the technique we used to characterize it. But first let us define the adjacency matrix of a given pixel P:



One "naive" way would have been to check, for each pixel, whether each of its eight neighbours are white or black and decide whether it must keep its current color. But this technique is not very efficient. Indeed, we need to check every pixel from a $512 \times 512 \times (\approx 2000)$ set of data which is highly time-consuming.
We overcame this issue by using a clever technique explained in the pseudo code 2. Each pixel is represented by the average of its value and those of its eight neighbours. Again, the addition needs to be executed quickly in order to avoid long computation time. This is also explained in pseudo code 2.

Once these computations were made, the goal was to decide if a pixel was well colored or not. To do this, we must fix constraints on the number of white/black pixels allowed in the neighbourhood of a given pixel.
In the case of a black pixel, we required that at least four of its neighbours must be black. Otherwise it would be whitened. Examples are circled in red on figure 2.6.

In the case of a white pixel, we required that at least four of its neighbours must be white. Otherwise it would be blackened. Examples are circled in orange on figure 2.6.

Algorithm 2 *DenoiseBlack* function

Data: *SecondImage* $\in \mathbb{R}^{512\times512\times n}$

This image must contain pixels intensity values $= 0$ (black) or 1000 (white)

Result: *ThirdImage* $\in \mathbb{R}^{512\times512\times n}$

This image does not contain any black pixel without at least four black neighbours.

Algorithm :

while $i \leq n$ **do**

$image = SecondImage[:,:,i] \in \mathbb{R}^{512\times512}$ (we work $z-slice$ per $z-slice$)

• Create a new matrix $\in \mathbb{R}^{514\times514}$ by adding two lines and two columns :

$$M = \begin{pmatrix} 0 & 0...0 & 0 \\ 0 & image & 0 \\ 0 & 0...0 & 0 \end{pmatrix}$$

• Create eight stacked matrices $\in \mathbb{R}^{514\times514}$, one for each configuration :
Top, Bottom, Left, Right, Top-Left, Top-Right, Bottom-Left, Bottom-Right

$$M_{Top} = \begin{pmatrix} 0 & image & 0 \\ 0 & 0...0 & 0 \\ 0 & 0...0 & 0 \end{pmatrix}, M_{Bottom} = \begin{pmatrix} 0 & 0...0 & 0 \\ 0 & 0...0 & 0 \\ 0 & image & 0 \end{pmatrix}, ...$$

• Compute the averaged sum :

$Sum = \frac{1}{9}(M + M_{Top} + M_{Bottom} + M_{Left} + M_{Right} + M_{Top-Left} + M_{Top-Right} + M_{Bottom-Left} + M_{Bottom-Right})$

• Check which Sum indices contain an amount greater or equal than $\frac{5}{9} * 1000$ (meaning they have ≤ 3 black neighbours), and transform these pixels into white color.

$M[Sum \geq \frac{5}{9} * 1000] = 1000$

• Delete zero lines added at the beginning :

$ThirdImage = M[1 : end-1, 1 : end-1]$

end

return *ThirdImage*

The algorithm developed for the function *DenoiseWhite* is exactly the same except that the limit on the sum becomes :

$$M[Sum \leq \frac{4}{9} * 1000] = 0$$

Since we blackened a white pixel only if it had three or fewer white neighbours [4].

[4] Remind that a black pixel has a value of 0 and a white pixel has a value of 1000.

With all these elements we have gone from a DICOM CT scan to a completely denoised image that is stored as a 3D matrix of black/white pixels under Python format.

2.4 Violin shape treatment

With regard to the elements presented in the previous section, we are now able to pass from a DICOM CT scan to a 3D matrix of pixels containing a denoised image of the instrument. The next problem we faced was to deal with black points corresponding to matter outside the musical instrument.
Indeed, black points in our image represent both the violin, the scanning support, the strings, the bridge, the fingerboard, the tailpiece, etc. For our analyzes we did not need all these components, we wanted to focus on the violin's body.
In order to better visualize this issue, we have displayed two cross-sections of an instrument in the figure 2.7 : one that has undergone a CT scan in Erasme and one in Saint-Luc.

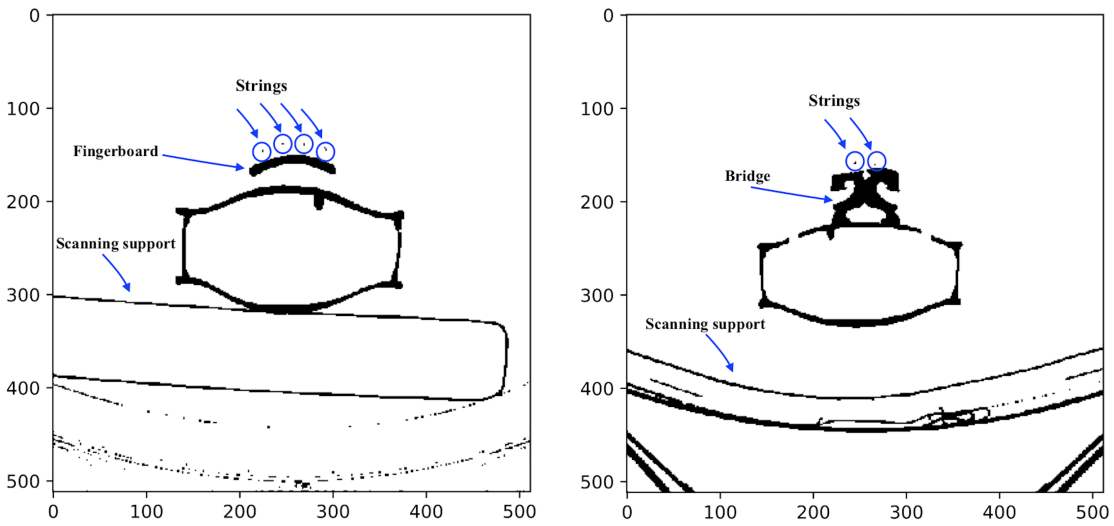


Figure 2.7: Physical components present in a cross-section of an instrument scanned at Erasme (left) and at Saint-Luc (right)

For the remainder of this research, we have decided to continue with the CT scans from Saint-Luc. As can be seen in the figure 2.7, the scanning support used at Erasme was in direct contact with the violin and the pixel intensities were mingled. This impliec developing an arduous pre-process phase to deal with that. In Saint-Luc, the scanning

A white pixel with 3 white neighbours (plus itself) has a *Sum* = $\frac{1}{9}(1000 + 3 * 1000) = 444.44$ and is thus blackened

system was more efficient because the support had a much lower pixel intensity than that of the violin so they did not interfere with each other.

Figure 2.8 shows the general methodology used to treat all components that do not belong to the violin's body.

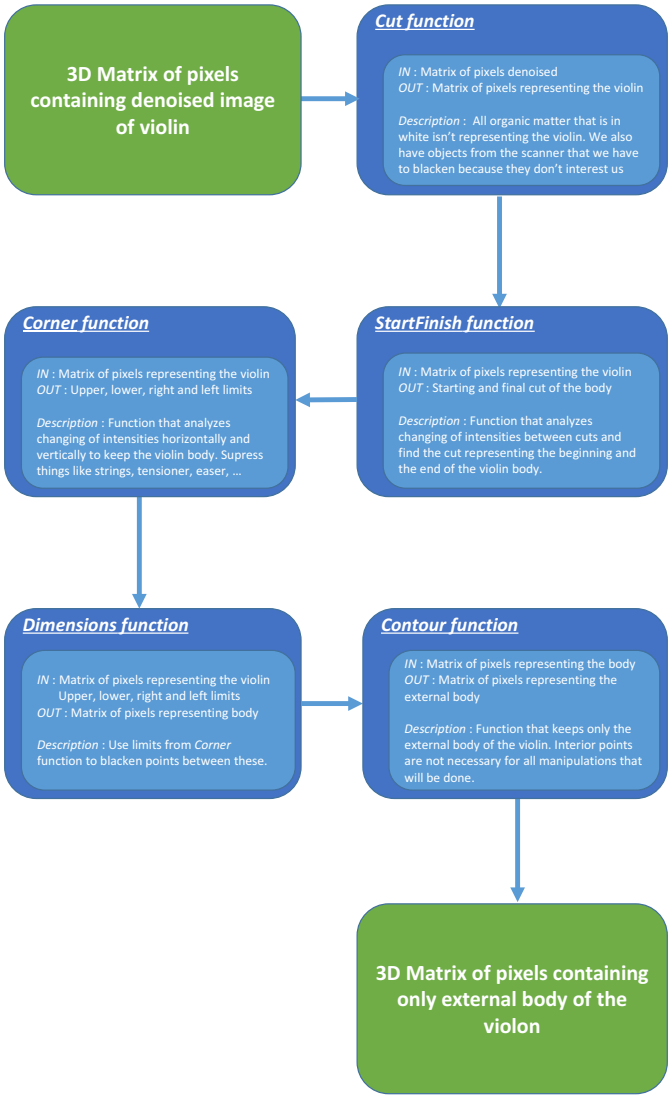


Figure 2.8: General methodology used via our Python program to obtain the external shape of a violin's body

We will explain each block from figure 2.8 the same way we did in the previous section. The only function we will not describe is the function *Cut* because it is com-

posed of manual limits that we set using the radiant viewer. These limits have been fixed individually because each violin is placed differently in the scanner. This function is part of the non robust aspect of our program because the one we created is not able to predict these limits.

In algorithm 3 we present the pseudo code of a function that returns us the bottom and the top of the violin body in terms of transverse cuts. Indeed, CT scans were made on a $701mm$ longitudinal distance and thus have collected more information than the one we are interesting in : the violin's body. The technique used in order to find $z - coordinate$ of the top and bottom of the violin is quite simple : for each $z - slice$ we just computed the sum of pixel values that belong to the slice and then looked for where the biggest changes in intensity occur.
These locations are the ones where the scan of the violin's body started and ended. Indeed, it is at the top and bottom of the sound box that our $z - slices$ contain the most material: and it is therefore for these slices that we will have the minimum sum (since a black pixel = 0).

Algorithm 3 *StartFinish* function

Data: *FourthImage* $\in \mathbb{R}^{512\times512\times n}$
This image is denoised and does not contain white and black imperfections
Result: *start* $\in \mathbb{N}$, *finish* $\in \mathbb{N}$
The values where the violin body has begun/finished to be scanned
Algorithm :
while $i \leq n$ **do**
$image = FourthImage[:, :, i] \in \mathbb{R}^{512\times512}$ (we work slice by slice)
• Compute the sum of the pixel intensities of this slice
$Sum = \sum_{i=1}^{512}\sum_{j=1}^{512} image[i, j]$
Store this sum
end
$start$ = Slice where is located the minimum of the $n/2$ first sums
$finish$ = Slice where is located the minimum of the $n/2$ last sums
return *start, finish*

Once we had the $start$ and $finish$ values we needed to cut our 3D matrix and keep only the part between these two limits.

At this point we were left with only the violin's body since we have removed unnecessary cross sections with the function *StartFinish* by examining the violin in the $z - direction$.
The second step was then, on each $z - cut$, to remove unnecessary components like the

strings, the bridge, etc as shown in the figure 2.7.
The function *Contour* computes the four limits (left, right, upper and lower) of the violin body according to the continuity of the horizontal/vertical black sum as shown in figure 2.9. To do this we simply placed ourselves at locations where the sum is the lowest, looked for a black pixel (which represents matter) in the corresponding row/column and continued in the corresponding direction until there no black pixel remained in the row or column. This location then corresponds to a border of the violin body.

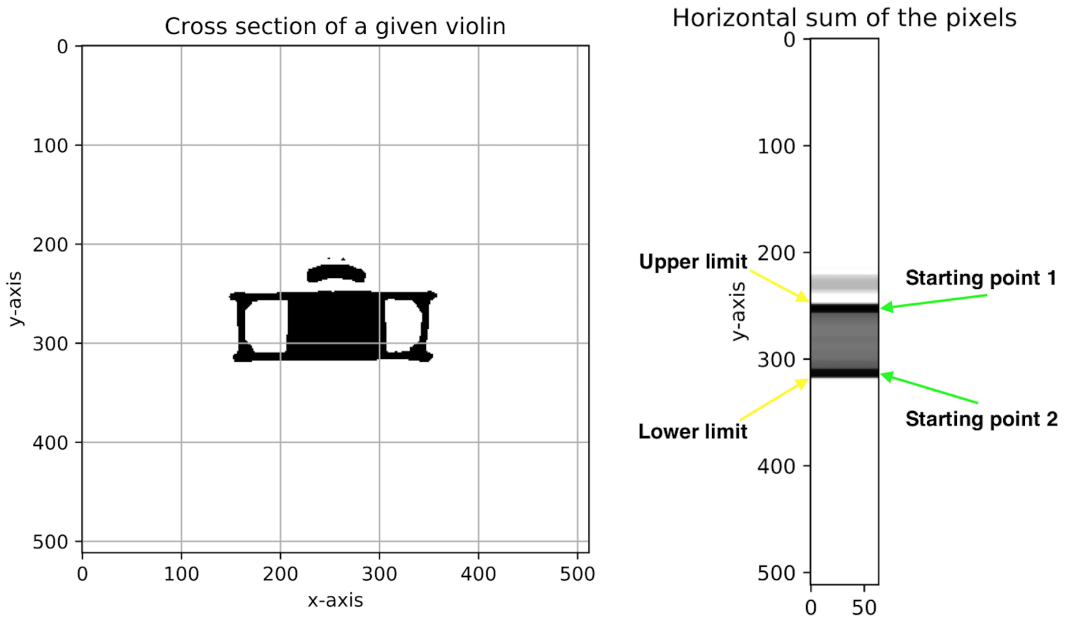


Figure 2.9: Continuity of the horizontal sum for a given cross-section

We can see in this figure 2.9 that the starting points are located near the extremities of the violin. This is linked to the fact that when we make the horizontal sum, the rows near the edge contain many pixels representing matter. And that's why we choose to start at these points to avoid too many unnecessary loop turns.
Then we notice that we stop when no black pixels are present in the horizontal sum: this means that we are at one extremity of the violin.
It is then sufficient to repeat this operation for the four sides and store the corresponding values.

This procedure is explained in algorithm 4.

Algorithm 4 *Corner* function

Data: *FourthImage* $\in \mathbb{R}^{512\times512\times n}$, $axis = x$ or y
This image is denoised and does not contain any white and black imperfection
The *axis* tells us if we work horizontally or vertically

Result: $Right, Left, Top, Bottom \in \mathbb{N}$
Represent limits of the violin body in each direction

Algorithm :

while $i \leq n$ **do**

$image = FourthImage[:, :, i] \in \mathbb{R}^{512\times512\times n}$ (we work slice by slice).

• Compute the sum of pixels intensity along $axis$

if $\{axis = x\}\ \forall j\ \ Sum[j] = \sum_{i=0}^{512} image[i, j]$

if $\{axis = y\}\ \forall i\ \ Sum[i] = \sum_{j=0}^{512} image[i, j]$

• Store the two positions where Sum is the lowest along the $axis$.

These positions are then where the intensity of black is the higher meaning that we are near the side of the violin.

$Positions =$ Positions of $Maximum(Sum)$

• For example to obtain the ***Right*** limit.

Start from the right *Positions* along $x - axis$ and go on the right direction until there is no black pixel left in the corresponding column.

$Right = Positions[1]$

while $\exists$ *black pixel along* $x - axis$ **do**

$\quad Right = Right + 1$

end

Same principle for $Left, Top, Bottom$

end

return *Right, Left, Top, Bottom*

Once we computed left, right, upper and lower limits we used the function *Dimension* presented in the algorithm 5 to resize our image for each $z - cut$. We obtained a matrix that defines the violin's body without any other components that are useless to us.
To free up memory, we stored this 3D matrix in 3D vectors x, y, z by storing only the coordinates of the black pixels (which represents the violin body).

Algorithm 5 *Dimension* function

Data: $FourthImage \in \mathbb{R}^{512\times512\times n}$: Denoised image without any white/black imperfections
$Right, Left, Top, Bottom \in \mathbb{N}$: Limits of the violin body in each direction
Result: $x, y, z \in \mathbb{R}^{m\times1}$
Vectors containing black values (representing the matter) positions of the violin
Algorithm :
Put white intensity for pixels outside $[Right : Left, Top : Bottom]$
Extract the position of the black pixels ($value = 0$) from $image$
$x, y, z = where(image == 0)$
return x, y, z

Knowing that we only need the external body of the violin for most of the analyzes we will perform, we have created a function that returns it to us.
This algorithm 6 named *Contour* goes through all rows and columns and returns the extreme points for each row/column that corresponds to the external body of the violin.

Algorithm 6 *Contour* function

Data: $x, y \in \mathbb{R}^{m\times1}$
These vectors contain the positions of the black values in our image (representing violin)
Result: $NewX, NewY \in \mathbb{R}^{p\times1}$
These vectors contain the black values positions of the exterior body of the violin
Algorithm :
• Along $x - axis$:
Keep only the maximum and minimum for each $x - coordinate$
For $i \in x - axis$
$NewX[iter] = [maximum(x[i])]$
$NewY[iter] = i$
$NewX[iter + 1] = [minimum(x[i])]$
$NewY[iter + 1] = i$
$iter = iter + 2$
• Idem along $y - axis$
return $NewX, NewY$

By means of this algorithm we went from a denoised 3D matrix of black/white pixel to three position vectors x, y, z which represent locations of the points from the exterior body of the violin shape.

Chapter 3

Rotation problem

We have presented all the elements needed to go from a DICOM CT scan to a 3D matrix representing the violin's body that will be stored as three-dimensional vectors x, y, z. This chapter explains the problems we ran into with the positioning in the scanner.

3.1 Problem explanation

The encountered problem was linked to the fact that the violin was never perfectly aligned with the axes of the scanner because the instruments had been manually laid on the support.
As a result, when displaying the violin's shape via our Python program, the violin body was never aligned with any mathematical reference axes. This problem is illustrated in figure 3.1 and 3.2 in which we can clearly see the above mentioned misalignments.

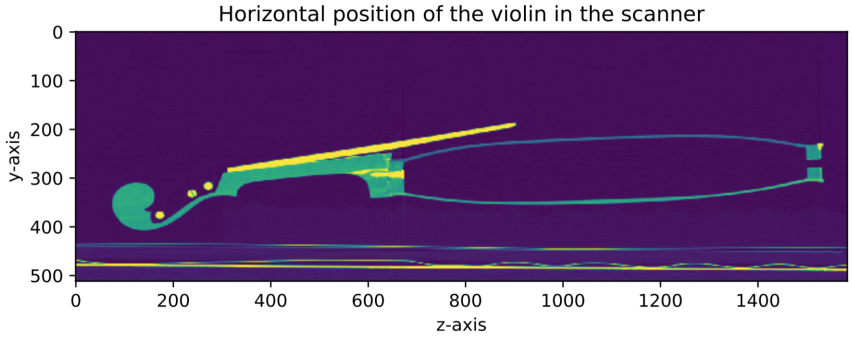


Figure 3.1: Longitudinal cross section of a violin in the scanner

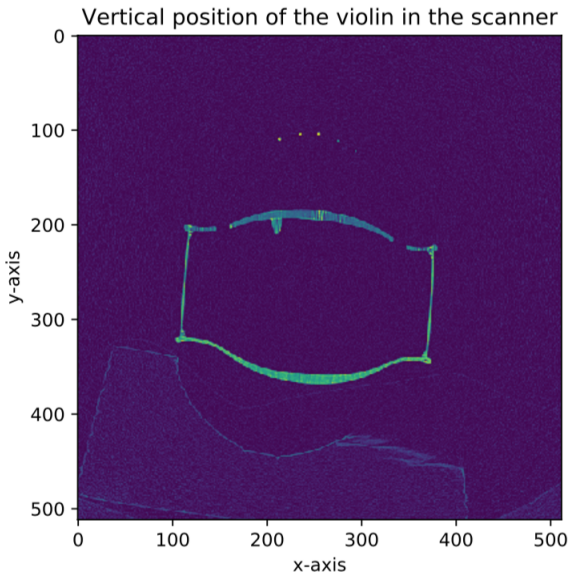


Figure 3.2: Transverse cross section of a violin in the scanner

Before proceeding to any manipulations and before computing any specific points, we needed to fix that problem. Note that we have started this research with previously existing CT scans. This explains that no devices to position the violins was available to us prior to doing the CT scans.
We thus suggested two ways to solve this issue :

- The first one was to create a computer function that can compute three corrective angles of rotation for each Cartesian direction. This would in turn allow us to align the instrument with a mathematical reference system. This technique is developed in section 3.2.
- The second technique is to tackle the problem at its root : make new CT scans with carefully calibrated positioning of the instrument in the scanner via a dedicated mechanical support. This technique is developed in section 3.3.

3.2 Solution 1 : Creating a corrective function

In order to identify the axes and align the instruments, one might be tempted to use the violin's neck as a guide, however this method would not yield valid results. As has been explained in the introduction, the necks of many ancient violins have been replaced, which means that most necks are late additions to an older soundbox. Moreover, necks are not always perfectly aligned with the soundbox. For these reasons, it is important to

align the soundbox without taking any external parameters into consideration.

Figure 3.3 explains the general methodology used to shift from a non-aligned instrument to an aligned one by finding the mathematical reference axes.

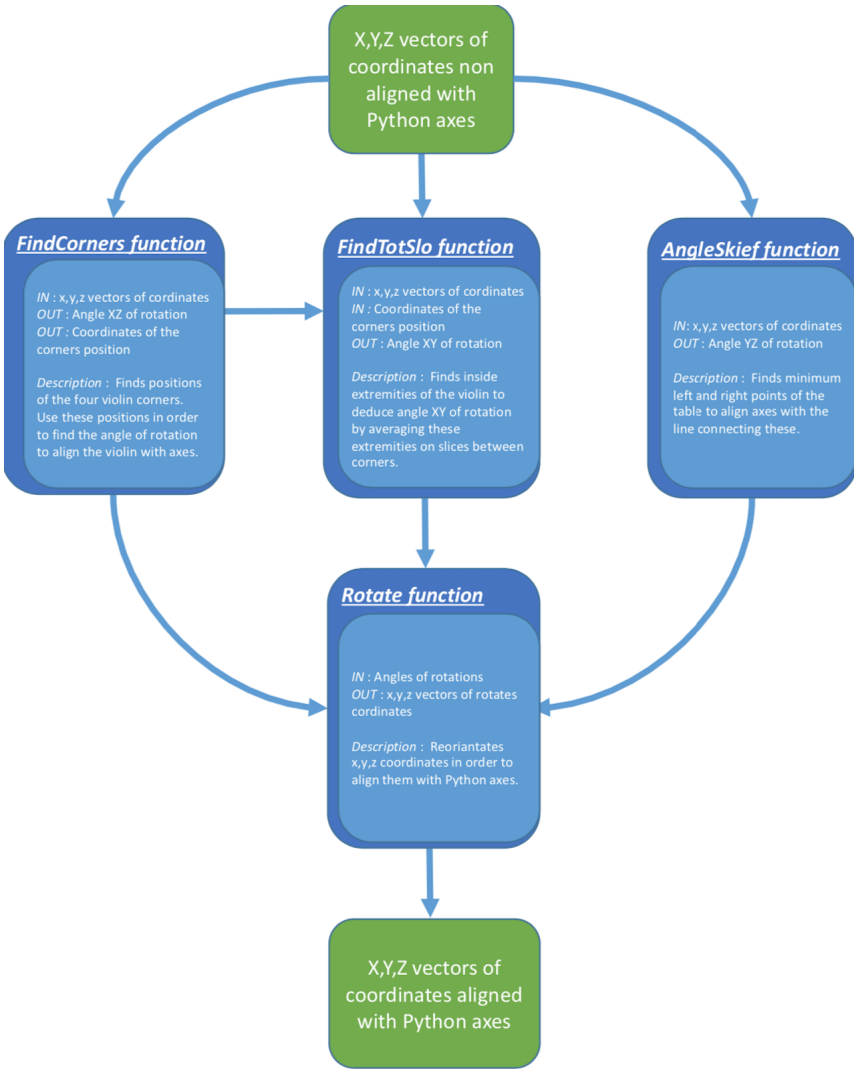


Figure 3.3: General methodology used via the Python program to obtain the external shape of a given violin body

Given that the general methodology is now fixed, we will explain in the next section the role of each function and present their underlying algorithm under the form of pseudo codes.

3.2.1 XZ rotation

We will begin with the *FindCorners* function that computes the corrective rotation angle to be applied regarding the $X - Z$ plane.
In order to find this angle, we first had to fix reference points and lines that would characterize our system. Our considerations were based on the fact that the middle part of the violin is more stable, geometrically speaking, than the extreme ones. Indeed, if a violin is recut, it must have been done on the top and/or on the bottom. Transformation is not likely to occur in the middle of an instrument. Sometimes, violins have been recut by splitting the table and back in two (i.e. by opening the joint of the two boards that make up them) and then reducing the width of these two halves by the center of the instrument, before gluing them back together. However, that represents a lot of work and the procedure is rather rare.
In this respect, we decided to start our computations from the four corners located at the four edges of the C-bouts as shown in figure 3.6.

Figure 3.4: Location of the four corners located at the four edges of the C-bouts

To find the corrective rotation angle, we first had to determine the position of the four corners. To do this we worked separately on each longitudinal section of the violin.

In each slice, we separated the left and right part. Then we had to figure out the location of each of the corners' points.
This was achieved thanks to a specific property inherent to these points: they are located where the slope of the tangent is maximal/minimal.
We made a spline approximation of each part. To do this we have aligned each part horizontally so that no $x - coordinate$ has two $y - coordinate$. In this way we obtain smooth spline curves. Then we used the Python *UnivariateSpline* function [1] which allowed us to approximate each part of the violin.
After that, we calculated the slope of this spline and found the places where it was maximum or minimum. This is illustrated in figure 3.5.

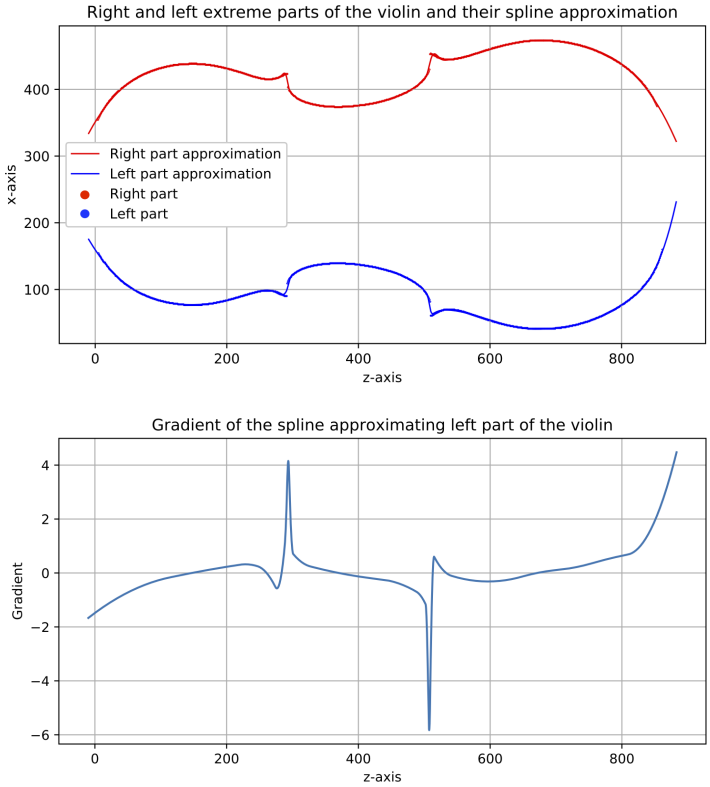


Figure 3.5: Right (red) and left (blue) violin parts, their spline approximations and the gradient of the left part

[1] One-dimensional smoothing spline fit to a given set of data points. Fits a spline $y = spl(x)$ of degree k to the provided x, y data.
Input : $x(N,)$ = 1-D array of independent input data. Must be increasing.
$y(N,)$ = 1-D array of dependent input data, of the same length as x.
$w(N,)$ (optional) = Weights for spline fitting. Must be positive. If None (default), weights are all equal.

These places will therefore be those where the corners are located.
To find the angle, we coupled the four corners two by two so that both upper corners were connected together and likewise for the lower corners. This is illustrated in figure 3.6 (left).
After that we averaged the two lines slopes in order to quantify the violin inclination in the reference axes. Then, we just reorient the violin according to this angle to align it with the reference axes as can be seen in figure 3.6 (right).

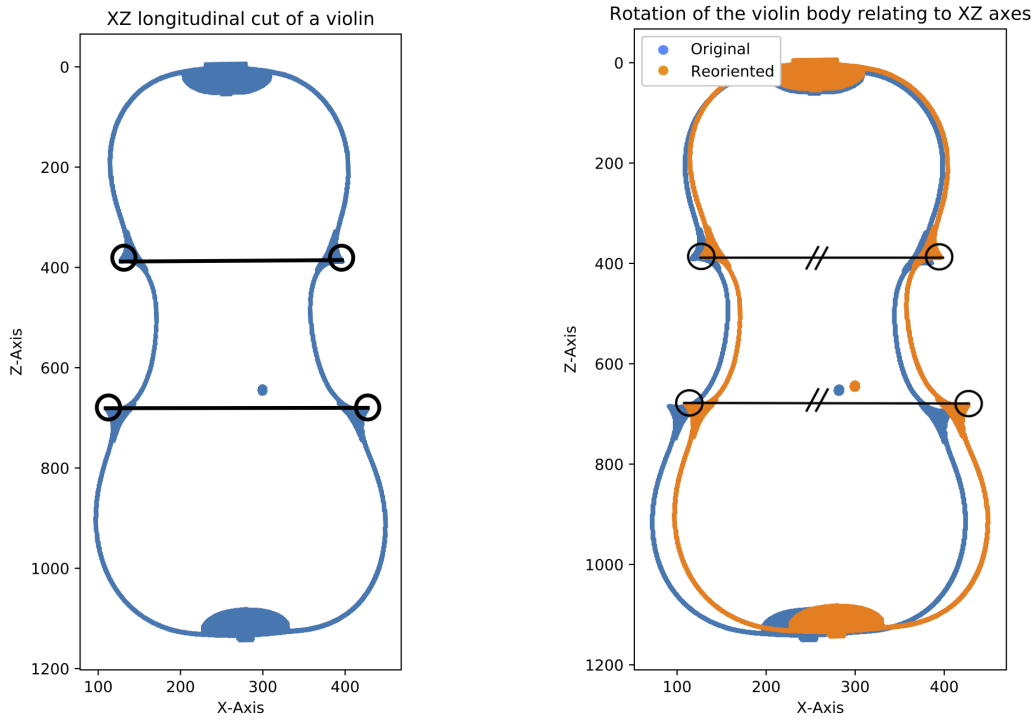


Figure 3.6: XZ cut of a violin that represents an original (blue) and reoriented (orange) violin body

N.B. the dot in the middle of the soundbox corresponds to the soundpost, a dowel placed inside the instrument, slightly lower that the bridge, under the highest string. The soundpost reinforces de structure of the instrument, and it also plays an essential role from an acoustic point of view.

In order to realize this reorientation process we followed the steps described in algorithm 7.

Algorithm 7 *FindCorners* function

Data: • $x, y, z \in \mathbb{R}^{m\times 1}$

These vectors contain black values positions of our image

• $couches, startCouches \in \mathbb{R}$

Represents at which y slice of the violin body we start, and how many slices we want to consider

Result: $angle \in \mathbb{R}$

Average slope of the two lines between the upper and lower corners

$Pointe1, Pointe2 \in \mathbb{R}$

Average Z-position of violin top and lower corners

Algorithm :

for *Each* $Yslice \in [startCouches, startCouches + couches]$ **do**

Cut slice into left and right parts (see Figure 3.5 top)

For each part, computes the gradient. (see Figure 3.5 bottom)

• Right side :

Compute coordinates $(XD1, ZD1)$ and $(XD2, ZD2)$: points where the gradient is maximun.

• Left side :

Compute coordinates $(XG1, ZG1)$ and $(XG2, ZG2)$: points where the gradient is minimun.

Compute the average slope : $\frac{1}{2}\left(\frac{ZD2-ZG2}{XD2-XG2} + \frac{ZD1-ZG1}{XD1-XG1}\right) \rightarrow angle$

Compute the average z-position : $\frac{ZD1+ZG1}{2} \rightarrow Pointe1$, $\frac{ZD2+ZG2}{2} \rightarrow Pointe2$

end

return $angle, Pointe1, Pointe2$

3.2.2 XY rotation

Now we will describe the *FindTotalSlope* function that computes the corrective rotation angle to be applied on the violin's body regarding the $X - Y$ plane. Again, we had to find two reference points in a given cross-section in order to deduce the corrective rotation.
The question was to select which of the upper part (the violin sound board) or the lower part (the violin's back) was the better to select. We have chosen the violin's back because violin makers build their instruments placed on the back, which suggests that this part should be more stable, geometrically speaking, and meant to stay horizontal.
Making violins from the back is a method typical for the Low Countries. It does not align with Italian building practices, which are the norm used all over the world nowadays. Our methodology however, was developed taking into consideration that most instruments analyzed in this master's proof were produced in the Low Countries. [4]

As shown in the left of figure 3.7, the black line that links the corners' interior points should be horizontal, and the angle determined by the slope of the latter characterizes our corrective angle.
This angle allows us to align the violin's body with the mathematical reference axes, as shown on the right of figure 3.7.

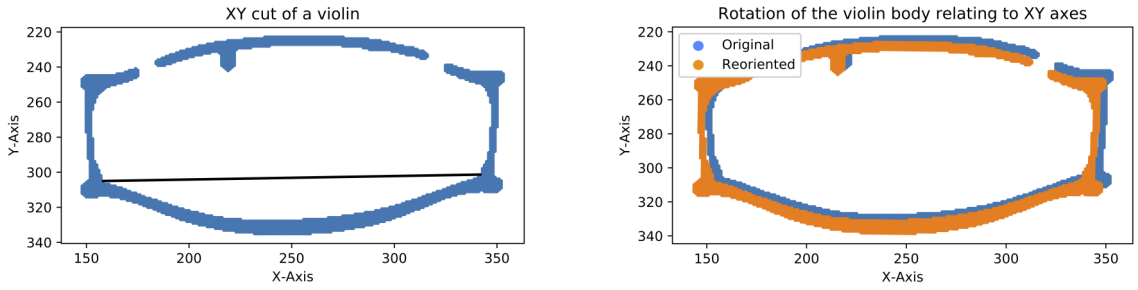


Figure 3.7: XY cross section of a violin that represents the original (blue) and the reoriented (orange) body

To find the corrective angle of rotation we had to find the different slopes of the straight lines of the bottom of the violin as shown in figure 3.7. To do this we worked on each cross-section of the violin. In a given slice we first spotted the corner's points inside the violin as seen in figure 3.7. Once these points were found, no other data was needed to find the slope of the line that connects them. Subsequently, the slopes of these straight lines had to be averaged for each cross section. Finally, thanks to the average slope obtained, we were able to define the angle of rotation to be applied to align the violin so that the slope is zero. The algorithm that allowed us to calculate this angle is described in the pseudo code 8.

Algorithm 8 *FindTotSlop* function

Data: • $TopBackX, TopBackY, TopBackZ \in \mathbb{R}^{r\times 1}$
Vectors containing the values at the top of the back part of the violin (see figure 3.8 (orange))
• $Pointe1, Pointe2 \in \mathbb{R}$
Average Z-position of violin spikes

Result: $angle \in \mathbb{R}$
Angle of rotation to align the violin with the mathematical axes

Algorithm :

for *Each Zslice* **do**
Keep only the lower part of the violin
Keep only the maximum $y - point$ for each $x - coordinate$
Delete points from which the variation in coordinate y with their neighbour is greater than the average of the variations (see figure 3.8 (green))
Start from the middle of the $x - axis$
• Go to right and find the extreme point with the maximum $y - coordinate \Rightarrow P1$
• Go to left and find the extreme point with the maximum $y - coordinate \Rightarrow P2$
Compute the slope between $P1$ and $P2 \Rightarrow angle$

end

return $angle$

Figure 3.8 illustrates the methodology applied to a given cross-section.
The first points we kept appear in orange on the figure: these are the minimum points of the violin back.
The green dots indicate the restriction of the orange points kept thanks to the criterion of variation in relation to the neighbours.
Once these green points were found, it was just a matter of keeping the most right and most left point, which clearly define the line we were looking for.

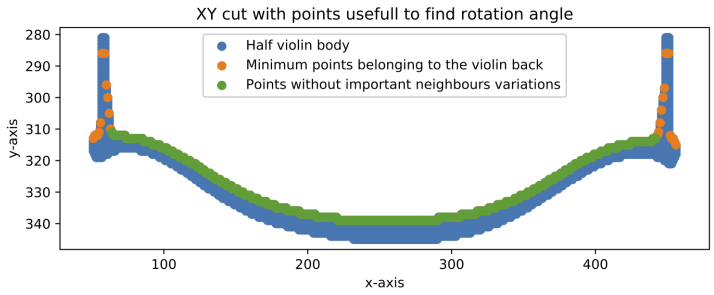


Figure 3.8: Points needed in order to compute corrective rotation XY angle

3.2.3 YZ rotation

We present here the *AngleLast* function which is the last one to correct the violin misalignment.
In order to compute the YZ corrective rotation angle, we again had to fix two reference points for each $Y - Zslice$. In this case, these points are defined as the minimum points from the violin back (one at the left side and the other one at the right side).
Figure 3.9 (left) displays the chosen reference line used to find the corrective rotation angle, and figure 3.9 (right) illustrates the reoriented violin body.

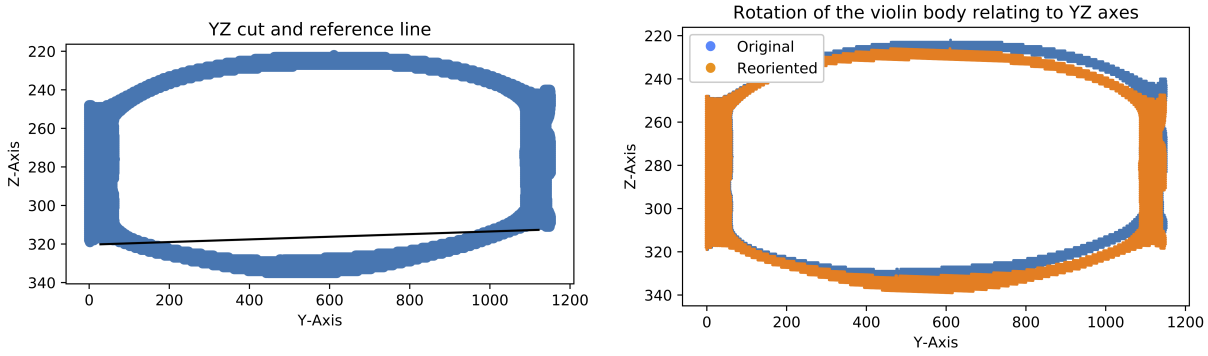


Figure 3.9: YZ cross-section of a violin that represents the original (blue) an reoriented (orange) body

To find these points we worked separately on each $y - zcut$.
In each cut, we kept at first the points that belong to the violin's back. To do this it was necessary to keep the points with a maximum coordinate in z for each y.
Then we separated the back of the violin in two : one part on the right and another on the left with respect to the middle of the $y - axis$.
For each part, it was then sufficient to find the minimum in z that represents the desired point. By connecting the points of each parts we then found the desired line on which to align our instrument.
The detailed behaviour of the function is described in the algorithm 9.

Algorithm 9 *AngleLast* function

Data: • $x, y, z \in \mathbb{R}^{m\times 1}$
These vectors contain black values positions of our image

Result: $angle \in \mathbb{R}$
Corrective angle of rotation to align the violin with reference axes

Algorithm :

for *Each xSlice* **do**
  Keep only $z - maximum$ points for each $y - coordinate$
  Compute the minimum in the left and right direction
  Compute the slope between these maximum.
  Store the slope.

end

$angle$ = Average(All slopes)

return $angle$

Thanks to these three directions of the violin's reference system, we were able to align it. In order to test the quality of our rotations, we applied a second time our rotations functions to the aligned violin. As expected, the rotation angles were all $0°$, which confirms that our violin is well aligned. This can also be confirmed graphically by looking at the different cuts that are well aligned with the references fixed.

3.3 Solution 2 : New CT scans

The elements developed in the section 3.2 are programming solutions.
In order to have better aligned CT scans that could improve the performance of all the functions created, we thought of a second technique : to tackle the problem at its root and perform new CT scans by calibrating the position of the instruments in the scanner. We know that our functions will always be necessary because positioning will never be perfect. But by calibrating the position, the robustness of our program could be almost perfect. We would indeed be starting from bases that would be approximately the same for all violins and we would therefore know nearly exactly how any violin would appear in Python.
A second advantage of this technique would be to create a support with a clear pixel contrast with the violin in order to facilitate and especially to improve the image's denoising step explained in chapter 2.

With the help of Mrs. A-E Ceulemans, Mr. Paul Fisette and Mr. Xavier Bollen, we thought about creating a support with a 3D printer. Mr. Bollen, who is a specialist in 3D printing, kindly accepted our request.

The goal was to create a support that would allow us to center the violins along the three axes as they pass through the scanner. The support had to be made of a material that would either not be radio-opaque, or a very radio-opaque material, the wood constituting violin body being in between. A very radio-opaque material, especially metal, might however be problematic for CT-Scanning, as it might cause image deformations. We also had to design the system in such a way that it would allow us to correctly orient the instruments without wasting time during the operation.

The support invented is a plate with four adjustable feet. Each foot is surmounted by a ball on which the violin (or viola) rests.
As the scanner base is curved, we have also added a curved block under the plate.
Representations of this support are displayed in the figures 3.10 and 3.11.

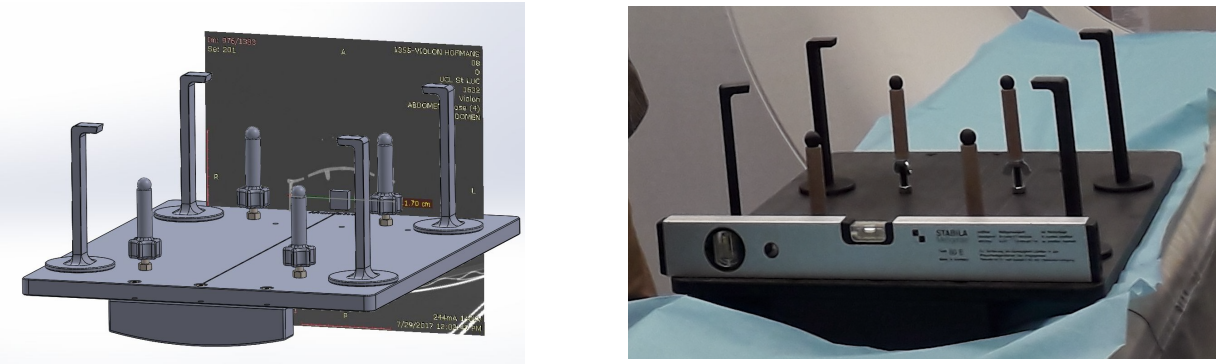


Figure 3.10: Support created to carefully calibrate the positioning of the violin

Figure 3.11: Support created to carefully calibrate the positioning of the violin

The support built at the EPL was tested at MIM before a scanning session organized at CUSL/Cliniques universitaires Saint-Luc on 4 May 2019. It proved difficult to use

because of the rotating feet, as the violins tended to rotate along with the feet. For this reason, these rotating plastic feet were replaced by wooden feet driven by metallic butterfly nuts. However, time was too short to find plastic nuts. The metallic nuts unfortunately caused artefacts [2] , which are optical deformations of the scan results."

These artifacts are quite problematic and therefore imply that the image processing phase needs to be reworked. In order to visualize the problem caused by these metallic nuts we can refer to the figure 3.12 (left). Our Python program as it has been designed does not offer the possibility to completely remove these artifacts. The program had been created well before these scans were available, and the time remaining when they were obtained (on 12 May 2019) was too limited to restart the image processing phase. Figure 3.12 (right) shows what our program returns to us after the basic image processing.

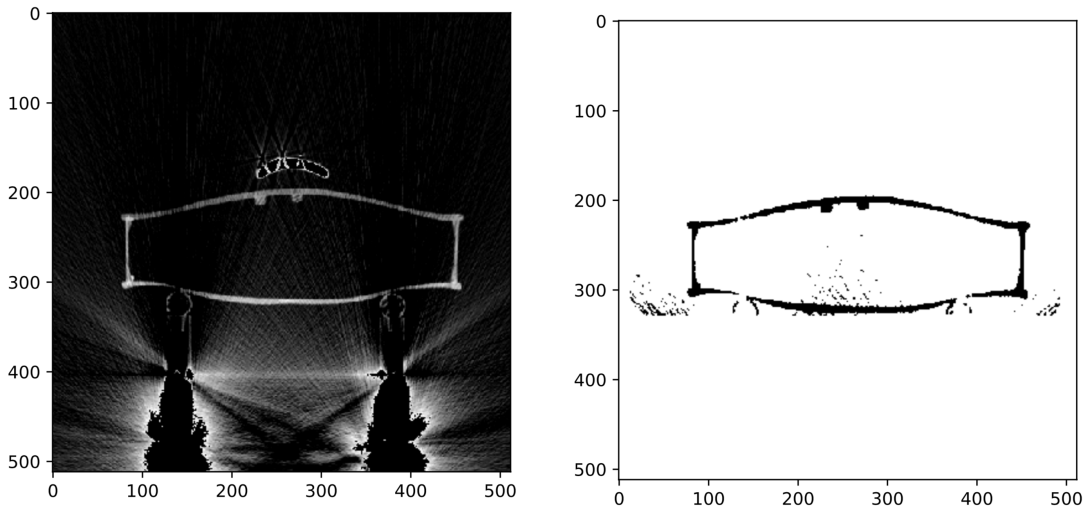


Figure 3.12: Cross-section of a violin with artifacts

That does not mean we do not know how to get anything out of these scans. Certainly, the image processing phase is not adequate, but the results obtained could still be significant. In our analysis, it should be taken into consideration that about a hundred cuts are of concern and that the analysis may be biased for them.

An idea that could be implemented in the future would be to compare a violin from an old scan with the application of our rotation and a violin from a new scan with support to see the difference.

[2]Information on this subject can be found on `https://www.vulgaris-medical.com/encyclopedie-medicale/artefact-en-imagerie-medicale`

Chapter 4

Acquisition of external features

So far, we have developed the necessary ingredients to move from a DICOM CT scan to a denoised image representing the body of a violin aligned with the mathematical reference axes.
From this point onwards, we are entering the essential phase of our research : creating functions that allow us to retrieve the external characteristics of a given instrument.
Indeed, the main goal is to objectively quantify the singularities and geometric anomalies of ancient violins and violas on the basis of a significant corpus of digital representations acquired through medical imaging.
As explained in chapter 1, the challenge was to enable organologists to better understand the original morphology of members of the violin family under the Ancient Regime and to evaluate the authenticity of ancient instruments.

To achieve this, and after discussion with Mrs. A-E Ceulemans who knows a lot of things on our subject, we decided to analyze the following characteristics :

- Position of the channel all around the violin table.
- Location and shape of the maximum zone of the violin table
- Position of the inflection points of the violin table.
- Degree of parallelism of the corners.
- Distance between the minimum points and their corresponding boundary points.

4.1 Channel computation

4.1.1 Computing elements needed

When an instrument is recut in the top/bottom part of the sound box, the depression surrounding the violin vaults disappears. The position of this depression, called channel, is then very interesting to characterize.
The general methodology used to determine the spatial coordinates of this channel is explained graphically in figure 4.1. All the functions used will then be detailed.

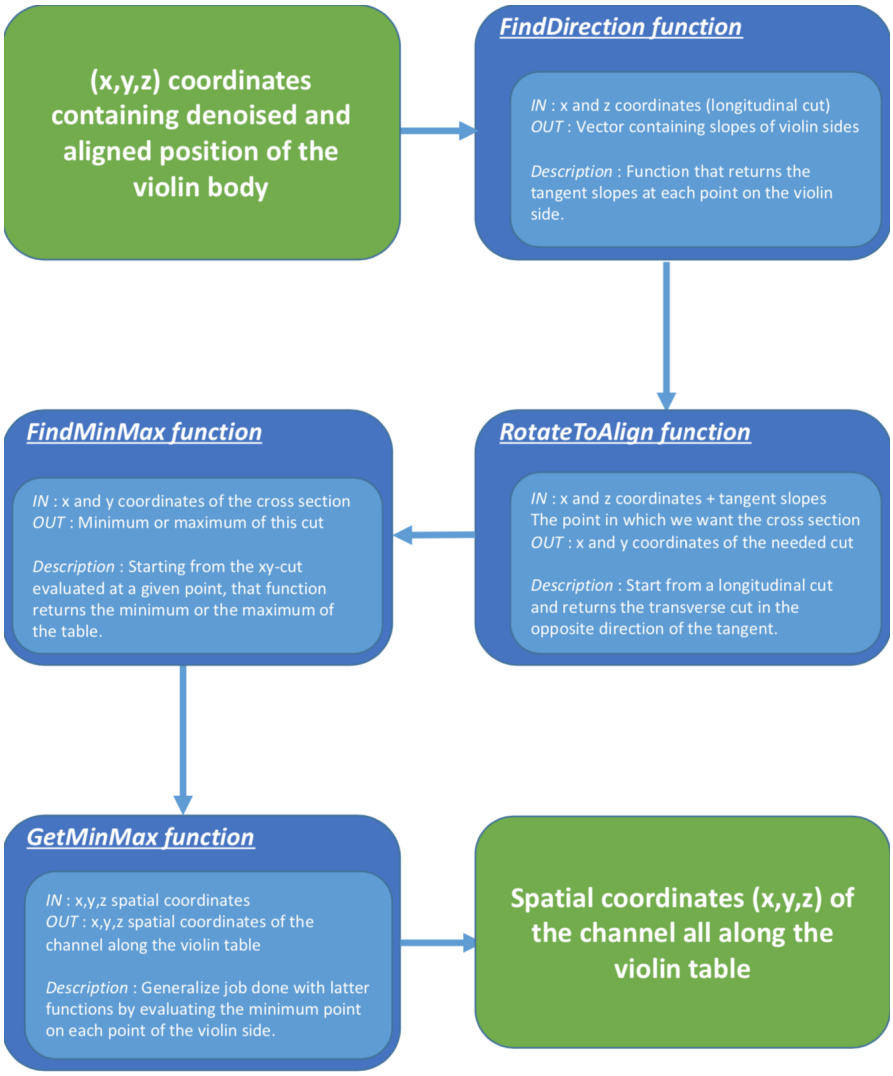


Figure 4.1: General methodology followed to compute channel points

In order to define a specific minimum point of the violin table, we had to determine

several things.
Once placed at a point on the edge of the violin, we had to decide in which longitudinal direction we would seek this minimum.
To do this we have broken down our longitudinal section into four parts as shown in figure 4.2. The lines used to delimit Parts I and III have been placed where the slope of the tangent on the violin side is zero. This was not the only possible choice, but it will not influence our future calculations.

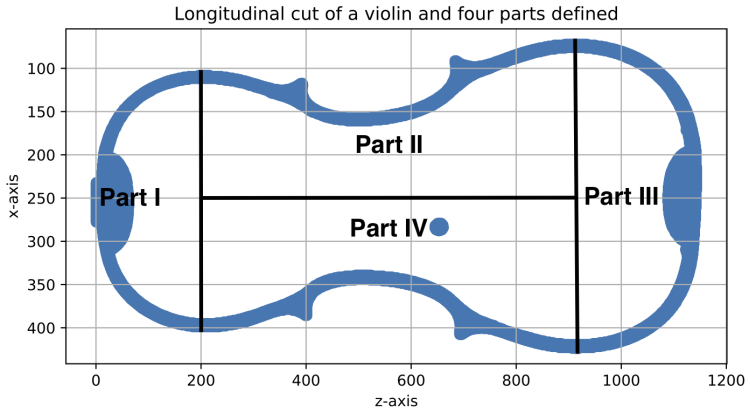


Figure 4.2: Defined four parts of a longitudinal violin cut

Then we had to place ourselves at each point on the edge of the violin and look for the minimum of the violin table in a direction to be defined.
As a result, three possibilities have been put forward to set this research direction :

- The naive idea was to set the direction of research so that it is vertical in domains 2 and 4 and horizontal in domains 1 and 3. But this technique was deemed too simplistic and did not provide the most accurate results.

- A second idea was to keep this vertical direction for zones 2 and 4 and make a circular approximation of zones 1 and 3. Then, using this approximation, we were able to fix our research direction being perpendicular to the circle. But once again this technique was quite too simplistic for zones 2 and 4 and yielded poor results.

- Last but not least, the idea was to make, for each zone, a spline approximation (as explained in section 3.2) of the violin's border. Once the equation is known, it is sufficient to set the research direction to be perpendicular to the slope of the tangent. This technique was clearly the most accurate because we could determine the minimum in the most natural sense in relation to the curvature on the violin side.
  The function that finds the research direction for each violin boundary point is described in algorithm 10.

Algorithm 10 *FindDirection* function

Data: • $x, z \in \mathbb{R}^{t\times 1}$
These vectors contain black values positions of our image for a given y-slice

Result: • $spl, spl1 \in \mathbb{R}^{t\times 1}$
Spline approximations of right and left side of the violin
• $xs, xs1 \in \mathbb{R}^{t\times 1}$
Abscissa points of spl and $spl1$
• $tanUp, tanDown \in \mathbb{R}^{t\times 1}$
The tangent value of the spline curve described by violin right/left side

Algorithm :
Separate the violin into left and right part
Do this **for loop** for each part :
for *each point* $\in z$ **do**
  Compute the spline approximation of the side $\Rightarrow spl/spl1$
  Compute the abscissae of this spline $\Rightarrow xs/xs1$
  Compute the gradient of the spline $\Rightarrow tanUp/tanDown$
end
return $tanUp, tanDown, xs, xs1, spl, spl1$

Once these directions of research were determined, we still needed to vertically align the violin with them in order to look for the minimum in a vertical way. We rotated it so that it would be vertically aligned in the opposite direction to its tangent. The function *RotateToAlign* performs this task and returns the violin that has been rotated as shown in figure 4.3.

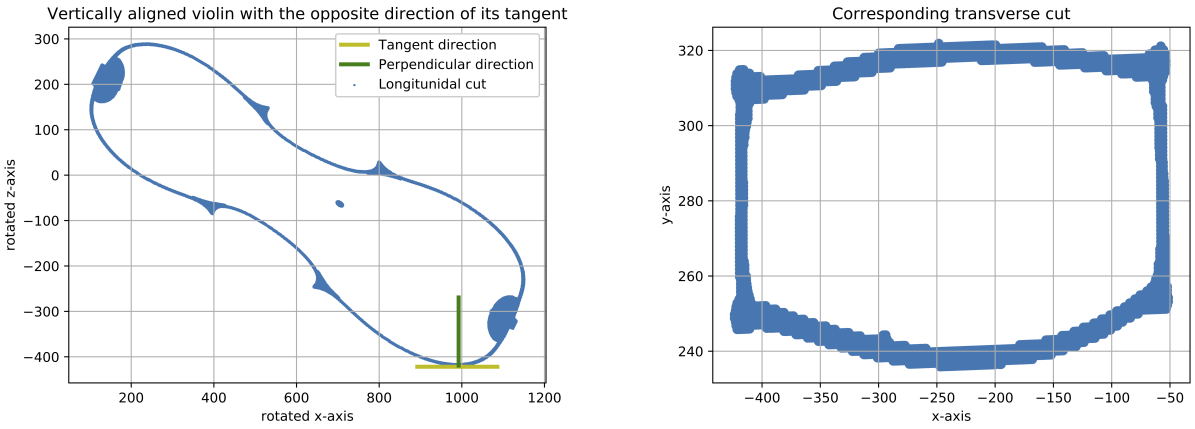


Figure 4.3: Longitudinal cut of a rotated violin, vertically aligned with opposite direction of its tangent and its corresponding transverse cut.

Once the appropriate cross-section was found, we could focus on finding the channel point.
To do this, we started by keeping the maximum points of the cross section which are points belonging to the violin table. (see orange points in figure 4.4) Then, we keep the left/right part of the table depending on which side's channel point we wanted to calculate. (see green points in figure 4.4).

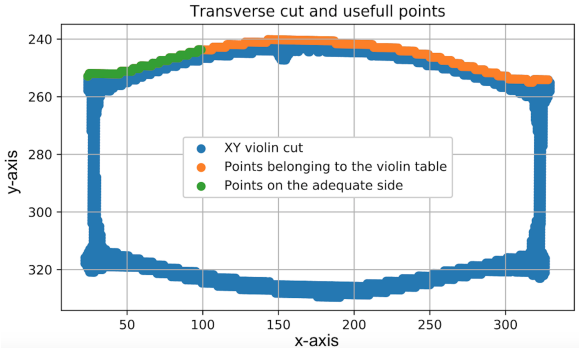


Figure 4.4: Adequate transverse cut and useful points to find the minimum

At this stage, we must make a few remarks.
First of all, it is important to note that a parameter was decided to define the desired thickness for the cross section. This parameter could not be too small : in this case we would have had a cross-section with a limited number of points and the analysis would therefore not have been accurate as shown in figure 4.5 (left). It could not be too large either : in this case we would have had too many points for the same cross section and the minimum between several adjacent sections could have overlapped as shown in figure 4.5 (right). This would have reduced the accuracy of our program. In practice this was finally determined through trial and error. We set this parameter by trying several different scenarios and keeping the one that gave the best results. We could also have determined it automatically, but this would have had almost no impact on our results.

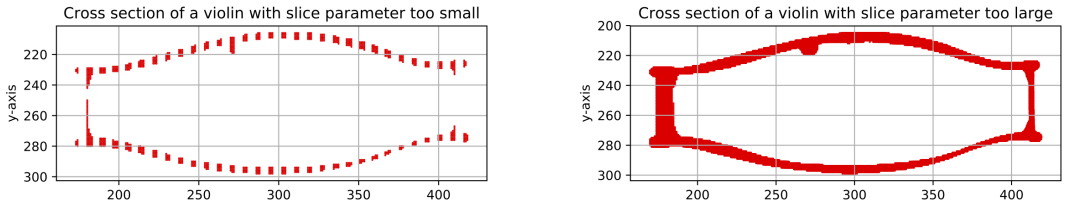


Figure 4.5: The cross section of an instrument with the thickness parameter too large and too small

Second, there were two ways in which the minimum could be defined :

- By simply taking the minimum relative to the $y - axis$ for each cross section. But as we will explain later, this technique did not lead to very representative results because of the pixel resolution.

- By approximating the green part of the figure 4.4 with a spline. And then compute the minimum of this spline.
  This technique was a little more arduous than the first one : it was necessary to define weights to assign to each $x - coordinate$ in order to make the best spline approximation.

These two techniques were implemented and we compared the differences between them during their use.
The function that computes the minimum of a given $XY - cut$ is presented in the algorithm 11.

Algorithm 11 *FindMinMax* function

Data: • $x, y \in \mathbb{R}^{m \times 1}$
These vectors contain location of points from a cross section
$string = max$ or min
Specifies if we are looking for the maximum or the minimum

Result: $xExtr, yExtr \in \mathbb{R}$
Minimum/maximum position along each axis

Algorithm :

• Keep only the violin table or the violin back (see figure 4.6 (orange)) .
Cut the table/back into two parts by keeping the first and last $\frac{1}{4}$ points. (see figure 4.6 (green and red))
• Use a $BoxPlot$ rule to delete outliers that does not belong to the table/back.
$Q1 =$ first quartile , $Q3 =$ third quartile , $IQR =$ interquartile range
Keep the values where $(y > Q1 - 1.5 * IQR)$ and $(y < Q3 + 1.5 * IQR)$
• Define the weights for the spline approximation by allocating importance to beginning and ending parts :
For right part : $weights[0 : \frac{1}{8}len(x)] = 5$, $weights[\frac{1}{8}len(x) : end] = 1$
For left part : $weights[0 : \frac{7}{8}len(x)] = 1$, $weights[\frac{7}{8}len(x) : end] = 5$
Approximate the upper part of the violin sound box by a spline (or not if the first technique is used) using the previous weights.
Compute the minimum/maximum of this spline $\Rightarrow xEtr, yExt$
return $xExtr, yExtr$

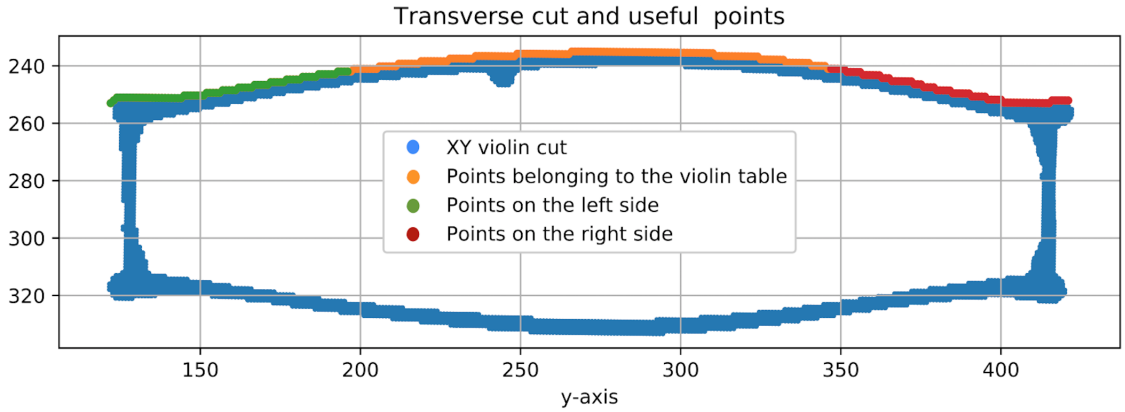


Figure 4.6: Transverse cut and useful points

We could then compute the minimum on a given $XY - cut$. It was still necessary to generalize this to find the minimums along the entire edge of the violin.
This was done by our function *GetMinMax* which is described by the algorithm 12.

Algorithm 12 *GetMinMax* function

Data: • $x, y, z \in \mathbb{R}^{m \times 1}$

These vectors contain the positions of the black values in our image (representing the violin body).

• $spl, xs, tan \in \mathbb{R}^{t \times 1}$

The coordinates of the spline approximating the right and left sides of the violin + the values of the tangents.

$range$

The number of maximum/minimum we want all around our violin.

Result: $xPos, yPos, zPos \in \mathbb{R}$

X-Y-Z positions of all the found maximum/minimum

Algorithm :

for $iter \in range$ **do**

Use $RotateToAlign$ to align the violin in successive perpendicular directions.
Keep only the corresponding cross section of the violin.
Use $FindMinMax$ to find the extremum coordinates.
Rotate with respect to the angle used in $RotateToAlign$ to obtain the initial coordinate system.
Store and concatenate these coordinates in $xPos, yPos, zPos$

end

return $xPos, yPos, zPos$

By means of this function we were able to compute the position of the channel points

all along the edge of the violin. The results are presented in the following subsection.

4.1.2 Results

By following the steps defined above, we were then able to compute the minimum of the violin soundboard from each border point.
As explained, we decided to test two techniques to calculate this minimum : the raw minimum or the minimum of the spline's approximation.
The graphs resulting from this analysis are presented in the figure 4.7 [1]. Left figure displays minimum points found with spline approximation and right figure represents raw minimums.

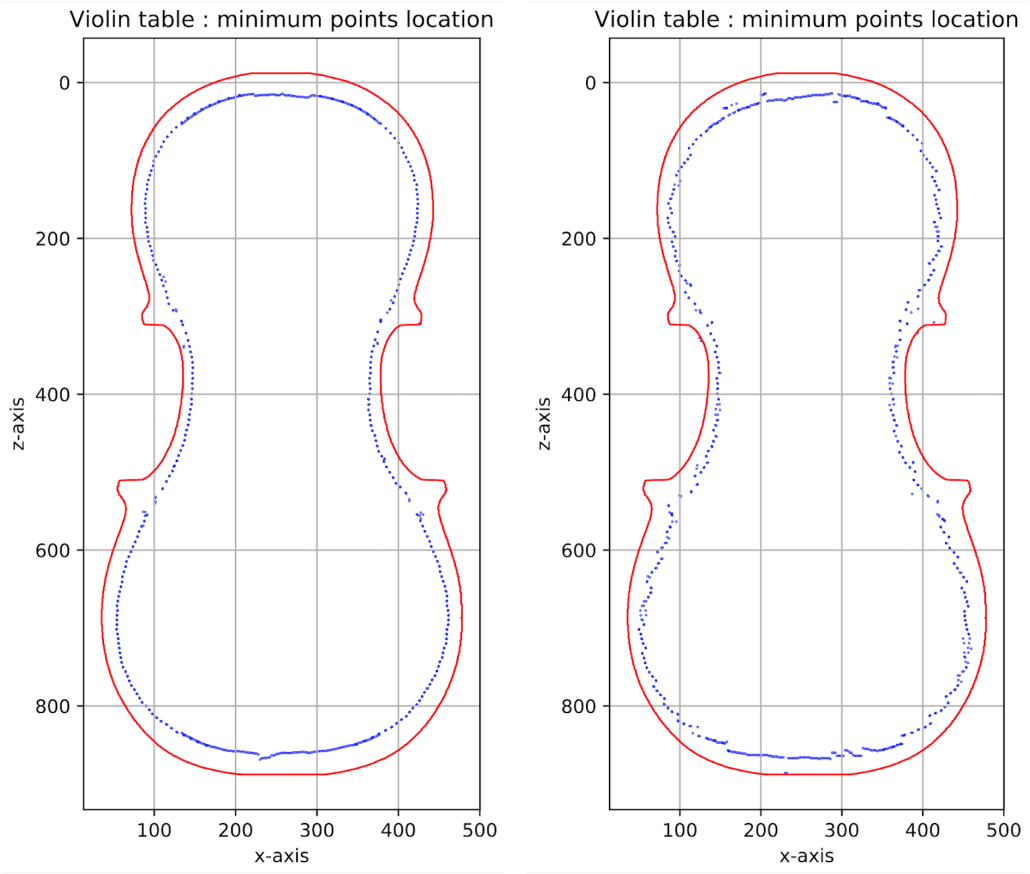


Figure 4.7: The location of the channel points (blue) of the violin table (red) with spline approximation (left) and with no approximation (right)

[1] These results holds for viola – Reference number : 2833 – Creator : Cuypers Johannes Theodorus – Geography : La Haye (Netherlands) – Date : 1761

These graphs illustrate the minimum positions of the violin soundboard. We could easily generate minimum points from the violin's back. These points are displayed in the figure 4.8.
The left figure displays the minimum points found with spline approximation and the right figure represents raw minimums.

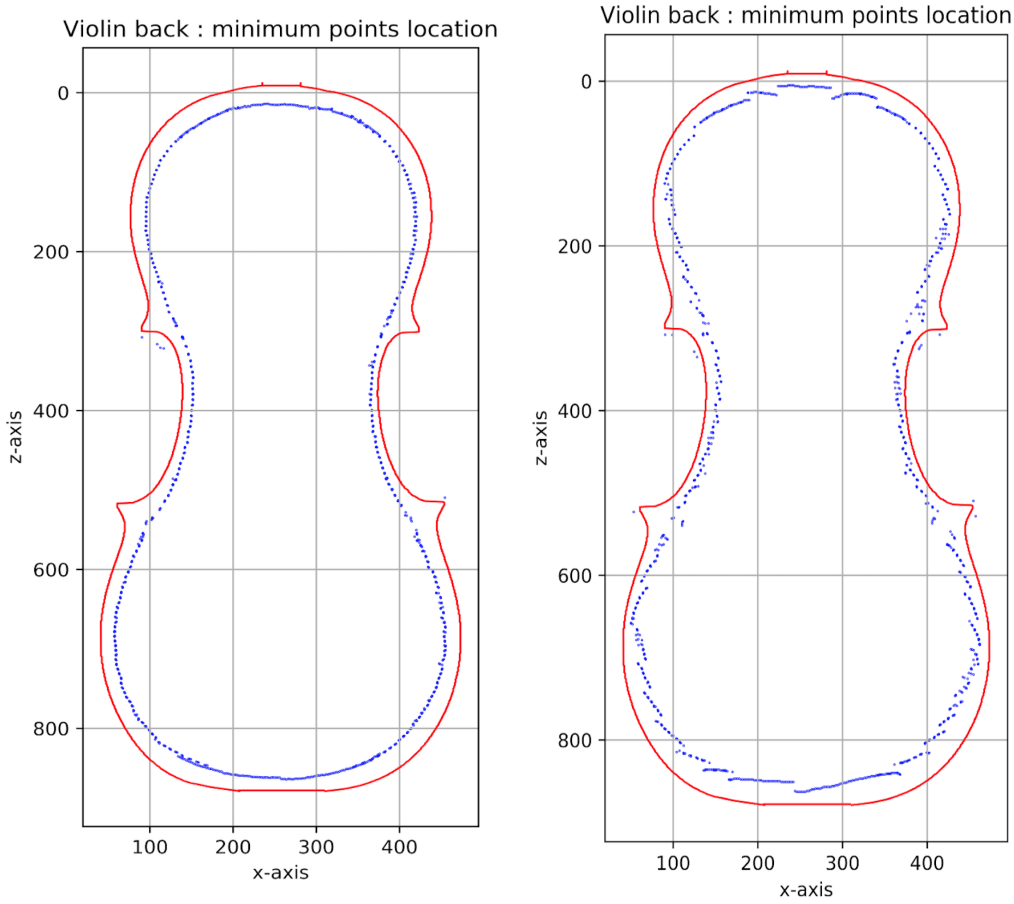


Figure 4.8: The location of the channel points (blue) on the violin back (red) with spline approximation (left) and with no approximation (right)

NB : The view used for these two graphs is a view from above the instrument. The view from the back is therefore taken from the inside of the soundbox.

In the previous figures, it is clear that the location of the channel points is more regular with the spline approximation.
This is because the position of the gross minimum is not well defined, what can be explained with what was presented in chapter 3 and 4:

- The violin points result from the intensity of the pixels present on the starting DICOM CT scans. As mentioned, these pixels have limited accuracy in each direction and therefore imply the approximate position of the raw minimum.
- The instruments also had to be aligned with reference lines that were set using the Python program. Naturally this alignment is not perfect from a mathematical point of view and thus implies the imperfect positioning of the raw minimum

It may therefore be that the minimum found is not exactly the same as what we would have found if we used the gross minimum technique without approximation.
As we can see in the figure 4.9, the minimum raw points are not exactly the channel points of the instrument for the two reasons expressed above.

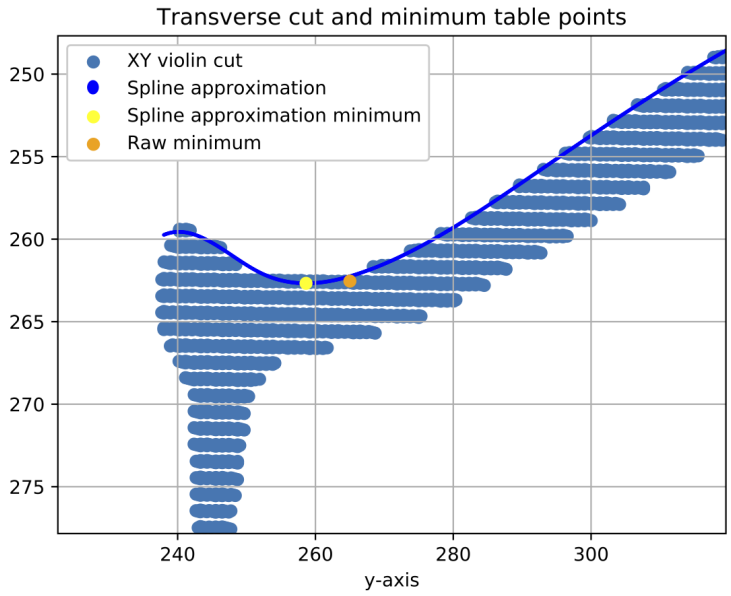


Figure 4.9: Minimum research on a transverse cut with and without spline approximation

It is for these reasons that we performed all our analyses with the spline approximation method, because it was more accurate and representative.
It makes it possible to compensate for the sometimes approximate position of the raw minimum as shown very well in the figure 4.9.

In order to have a better idea of the location of the channel points we created graphs representing the transversal projection of these points. This is done in the figure 4.10. The upper figure shows the projection of the channel from zones 2 and 4 and the lower one the projection of the channel from zones 1 and 3 (notations coming from figure 4.2.

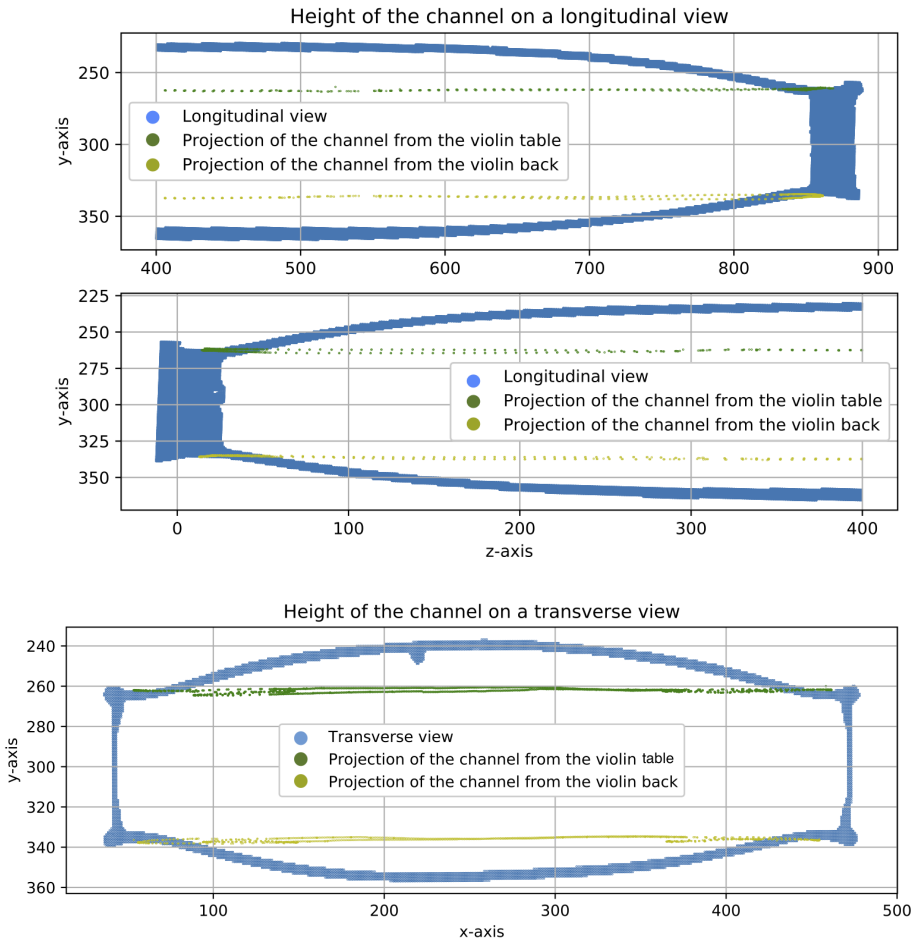


Figure 4.10: Longitudinal and transverse height of the channel

From this point onwards, a fair number of graphs are available to get an idea of the position of the channel points. But we can still take the analysis further by graphically characterizing the distance between this channel and the edge of the violin. To do so we used the algorithm that calculates the minimum for each cross-section 12. We simply added a function that stores the location of the respective border point. Then, all that was left to do was is to calculate the distance between these points and their corresponding minimum, all along the edge of the violin. Thanks to these results, the graph 4.11 could be made to best represent such distance.

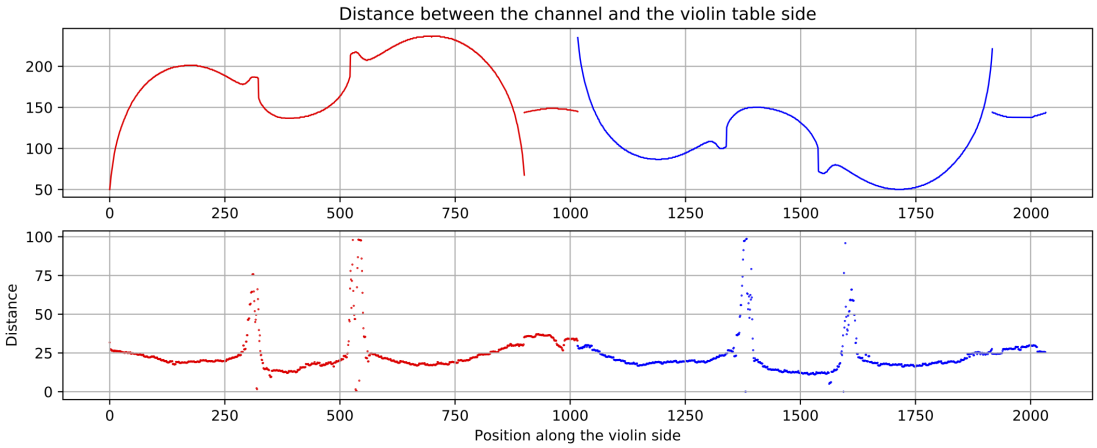


Figure 4.11: Distance between the violin side and the channel

NB : The two red and blue lines are there to indicate where we are on the top and bottom part of the violin.

Using the figure 4.11 the behaviour of the distance between the border and the channel throughout the violin could then be analyzed.
We noticed specific behaviour in some places. Indeed, when we are located near the corners of the violin (positions $300, 900, 1300, 1600$) the distance increases significantly. This seems logical since the corners are located a little behind the border of the violin and therefore the distance increases.
A second type of behaviour was visible at the top and bottom of the violin sound box. This may mean that the violin has probably not been recut, since it is normally in these places that the channel approaches the edge or disappears completely. In order to have a better idea about this, we would need to compare the behaviour of this distance on many violins.
Finally we noticed that the distance seems rather symmetrical on both sides of the instrument. This seems logical but may not always be the case and can therefore be an important feature to note.

4.2 Computation of the maximum zone

Another way of characterizing the instruments was to identify the highest points/zone of the violin table and back, i.e. the top of the vault.
To do this, we followed the same steps as for the search for channel points :

- Find the direction perpendicular to the edge of the violin.
- Store the adequate cross-section
- Compute the maximum of this cross-section

But by following these steps, only the absolute maximum would be found, and from which we would not be able to define a zone. This zone is therefore defined as the part of the vault located around the highest point(s), with a tolerance of 1, 2 or 3 mm for example.
To fix this and obtain a zone, we decided to calculate the maximum (as we did for the minimum) and then define its $x - coordinate$ in the corresponding transverse direction with an accuracy factor of δx :

$$(x, z) \Rightarrow (x + \delta x, z) \text{ and } (x - \delta x, z)$$

It was then be possible to define a zone around each overall maximum from each cross-section.
This parameter δx can be modified according to the precision required for the corresponding maximum zone.
If we set $\delta x = 10$ which means an accuracy of $10mm$ we get the following maximum surfaces for the violin table in figure 4.12 [2]. This graph illustrates the longitudinal projection of this zone.

[2]These results holds for viola – Reference number : 2833 – Creator : Cuypers Johannes Theodorus – Geography : La Haye (Netherlands) – Date : 1761

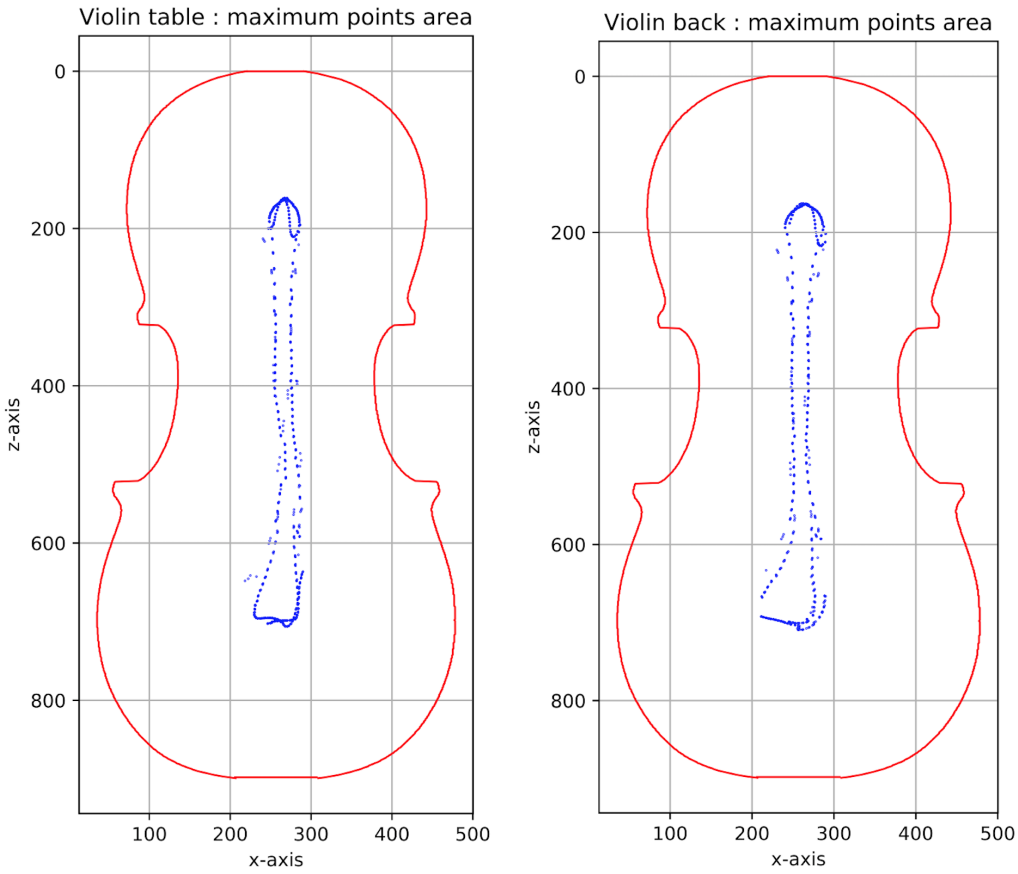


Figure 4.12: The location of the maximum point zone (blue) for the violin back (red left) and for table (red right)

NB : The view used for these two graphs is a view from above the instrument. The view from the back is therefore taken from the inside of the soundbox.

Once these points were calculated, we could also start displaying their projected position in the $y - zplane$. To do this we projected the maximum points of the violin table and those of the violin's back.
Indeed, it is important to check whether this zone is curved or flat. The luthiers attach great importance to the examination of the "top" of the vault. In "good" violins, this zone is arched, but on violins of poorer quality, it seemed to be somewhat "flat" (to quote the luthiers).
The figure 4.13 shows the $y - z$ projection of the maximum zones of the violins' tables and backs. This figure suggests that the latter are curved.

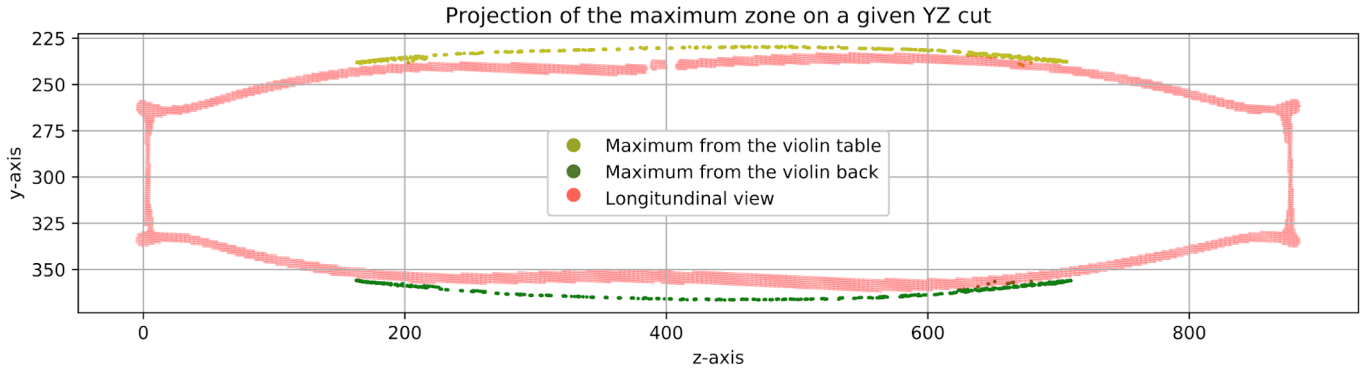


Figure 4.13: Longitudinal height of the maximum zone from the violin's back (green) and table (yellow)

NB : The red part represents a longitudinal section of the violin taken at 1/4 of the width of the violin.

4.3 Inflection point computation

We will now discuss another characteristic of the violin: the inflection points of the violin's table/back. Indeed, the zone formed by these inflection points could help luthiers and musicologists in their discussions.

The methodology used to calculate these inflection points is almost the same as that used to calculate the channel points.

- Find the direction perpendicular to the edge of the violin
- Store the corresponding cross-section
- Keep the points on the back or table of the violin
- Approximate the corresponding curve by a spline and find the inflection points.

This methodology is detailed in the algorithms 13 and 14.

The first algorithm was used to find the coordinates of the two inflection points of a cross-section of the table or the back of the violin.

Algorithm 13 *FindInflexion* function

Data: • $x, y, z \in \mathbb{R}^{m\times 1}$

These vectors contain black values positions of our image (representing violin body).

Result: $PIX, PIY, PIX2, PIY2$

The X-Y coordinates of the first and second inflection points of the table on a given z-slice

Algorithm :

• Keep only the violon table/back.
Cut the table/back into two parts, right and left.
Use $RotateToAlign$ to align the violin perpendicular to the tangent on its side.
• Approximate the left part with a spline.
Compute the gradient of this spline. $\Rightarrow grad$
Find point where $grad = 0 \Rightarrow PIX, PIY$
Repeat for the right part $\Rightarrow PIX2, PIY2$ **return** $PIX, PIY, PIX2, PIY2$

Once these points were determined, it was sufficient to generalize this in the same way as what had been done previously for channel points.
This is what we do in the algorithm 14

Algorithm 14 *GetInflexion* function

Data: • $x, y, z \in \mathbb{R}^{m\times 1}$

These vectors contain black values positions of our image (representing matter)

• $spl, xs, tan \in \mathbb{R}^{t\times 1}$

The coordinates of the spline approx. on the right and left sides of the violin+ the values of the tangents.

$range$

Number of inflection points we want around our violin.

Result: $xPos, yPos, zPos \in \mathbb{R}$

The X-Y-Z positions of all the inflection points found

Algorithm :

for $iter \in range$ **do**
  Use $RotateToAlign$ to align the violin in successive perpendicular directions.
  Keep only the corresponding violin cross section.
  Use $FindInflection$ to find coordinates of the inflection points.
  Rotate with respect to the angle used in $RotateToAlign$ to obtain the coordinates in the initial coordinate system.
  Store and concatenate these coordinates in $xPos, yPos, zPos$
end
return $xPos, yPos, zPos$

Thanks to these two algorithms developed, we were able to define the locations of these inflection points along the edge of an instrument as shown in the figure 4.14 [3].

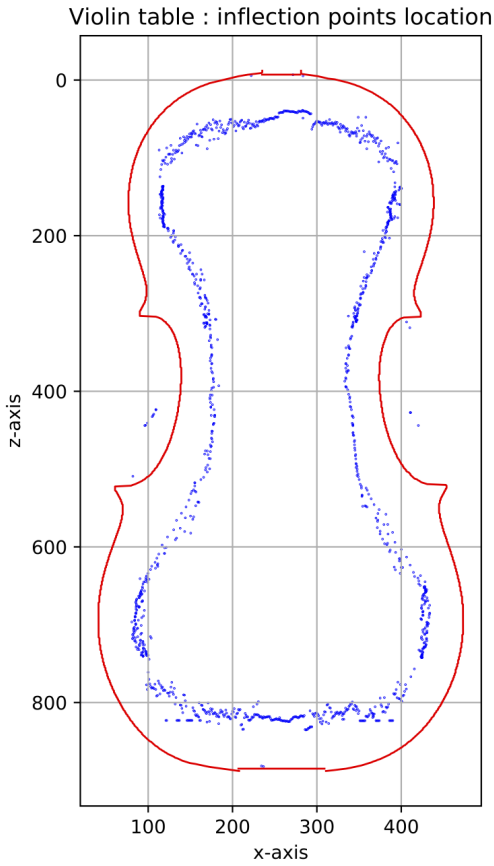


Figure 4.14: Location of the inflection point zone (blue) for the violin table (red)

The first thing that can be noted is that the position of these inflection points is not as clear as that of the channel points.
Unfortunately, this seems logical. Indeed, the notion of the inflection point is more sensitive than that of the minimum of the table. The inflection points must be defined using the spline approximation, as a small difference in the approximation of this spline may result in a more pronounced displacement of the position of the inflection point
A second reason is that in some cuts, part of the matter is "missing" due to the passage through a sound-hole [4]. As a result, the approximation of the spline contains fewer points

[3]These results holds for viola – Reference number : 2833 – Creator : Cuypers Johannes Theodorus – Geography : La Haye (Netherlands) – Date : 1761

[4]For any explanation of the specific vocabulary on violins, refer to the figure A.1 in the appendix issued from `http://www.williamsfineviolins.com/wp-content/uploads/2014/09/Parts-Poster.jpg`

to approximate and is not as accurate as the minimum for which we only need points near the edge.
This explains why the accuracy in those case is not as good as for channel points. In spite of this, we can see quite clearly the general appearance of the zone defined by these points.

Another way to visualize these points is to project them into the YZ plan as we did for the minimum points.
The projected positioning is shown in figure 4.15

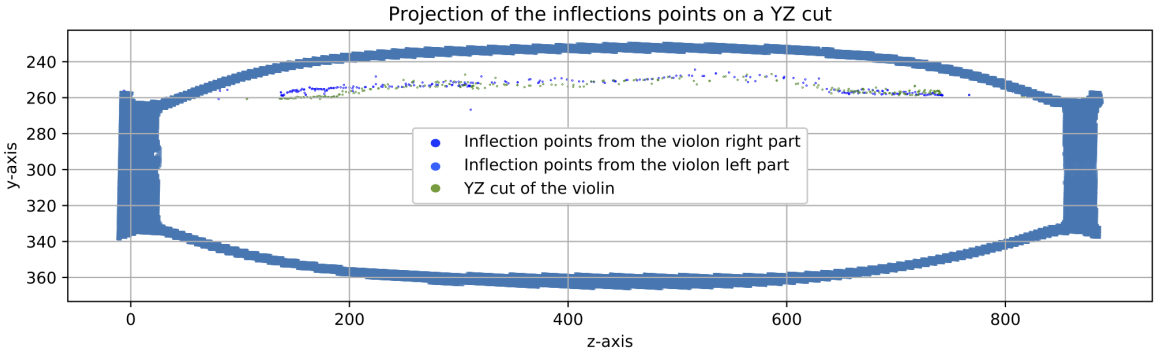


Figure 4.15: Longitudinal height of the inflection zone from the violin table

NB : The blue part represents a longitudinal section of the violin taken at half the width of the violin.

These graphs can help get a better idea of the positioning of these inflection points along the edge of a given instrument.

4.4 Computation of the parallelism of the spikes

The last characteristic that our program coult quantify is the degree of parallelism between the four corners located on the violin border.
The first thing to do was obviously to calculate the spatial coordinates (x, y, z) of these corners.
To do so, we simply use the algorithm 7 which finds the $xz - position$ of the four spikes for each $y - slice$. We therefore simply used that function for each $y - slice$ to obtain a dataset containing the spatial coordinates of these spikes.
Once this work was done, we had to quantify the degree of parallelism between them. At this stage, we suggested two ideas:

- Project these 3D spikes on the $xy - plane$ and then do a linear regression to get the $xy - slope$. And also do this work with the projection on the $yz - plane$.
- Calculate their angle two by two in the 3D space to obtain a 3D angle

The notations and legend from figure 4.16 were used :

Figure 4.16: Legend and notations used for the spikes positioning

4.4.1 2D angles with projection

In order to compute these 2D angles we decided to work first with $xy - coordinates$ and then with $yz - coordinates$.
To find the four angles we simply computed the linear regression of the four spikes and stored the slope of this regression that we converted into degrees. And we repeat this process with $yz - coordinates$.
The formula used to pass from two slopes m_i, m_j to a angle in degrees was the following :

$$angle = tan^{-1}\left(||\frac{m_i - m_j}{1 + m_i m_j}||\right) \times \frac{180}{\pi}$$

The figures 4.17 and 4.18 display projections of these spikes and their linear regression. The tables 4.1 and 4.2 present the corresponding 2D angles.

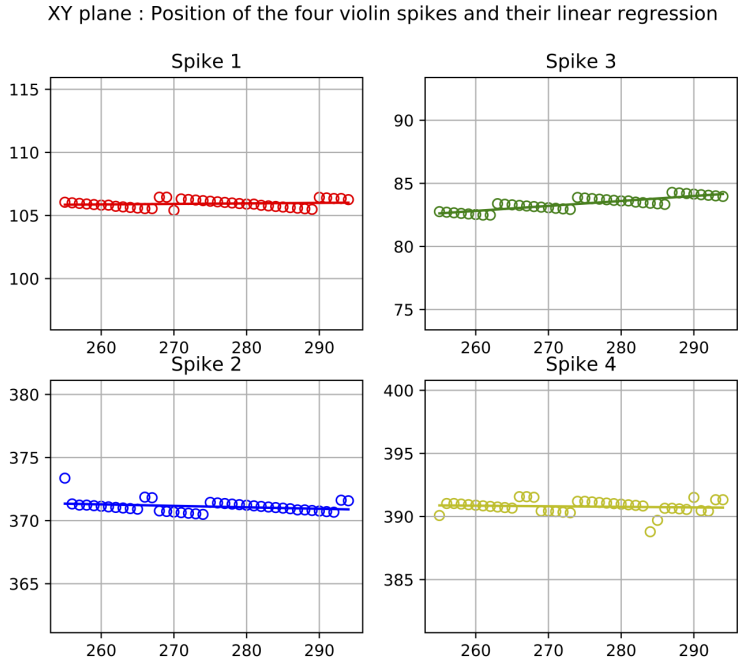


Figure 4.17: Orthogonal projection of the corners in the XY plane and their linear regression

Angle	P1-P2	P1-P3	P1-P4	P2-P3	P2-P4	P3-P4
XY Angle	0.9°	2.02°	0.5°	2.92°	0.41°	2.51°

Table 4.1: XY angles between corners

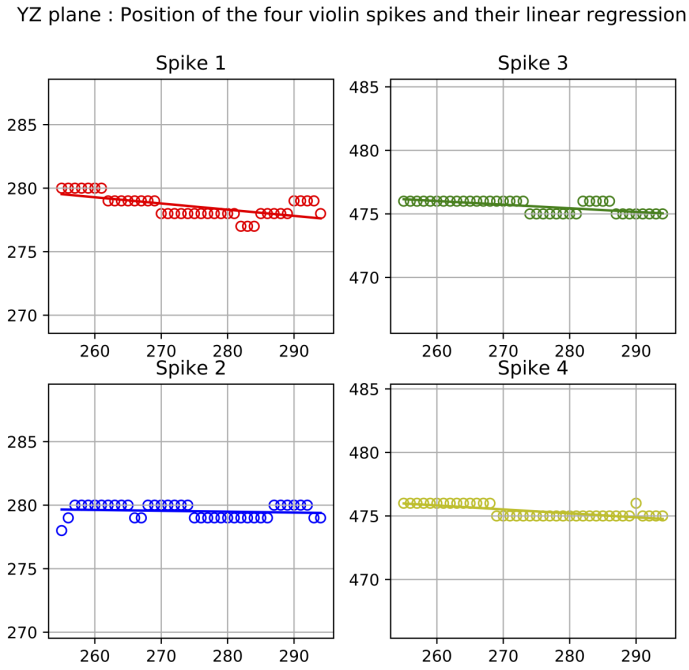


Figure 4.18: Orthogonal projection of the corners in the YZ plane and their linear regression

Angle	**P1-P2**	**P1-P3**	**P1-P4**	**P2-P3**	**P2-P4**	**P3-P4**
YZ Angle	2.41°	1.16°	1.01°	1.24°	1.4°	0.16°

Table 4.2: YZ angles between corners

These tables therefore give us an idea of the positioning of the spikes between them. With regard to the violin used, we noticed that the angles varied between $0°$ and $5°$.

4.4.2 3D angles

The process of calculating 3D angles was a bit trickier than the previous one.
On each of the spikes i having $m_i = xy - slope$ and $n_i = yz - slope$ we computed the coordinates of their corresponding unit vector :

$$U_i = \left(\frac{n_i}{\sqrt{m_i^2 + n_i^2 + m_i^2 n_i 2}}; \frac{m_i n_i}{\sqrt{m_i^2 + n_i^2 + m_i^2 n_i 2}}; \frac{m_i}{\sqrt{m_i^2 + n_i^2 + m_i^2 n_i 2}} \right)$$

We then made the scalar product of the two unit vectors that gave us the cosine of the angle formed.
Finally, we had to compute the $arccosine$ of the scalar product to obtain the exact angle between the two lines in radiant that we converted into degrees.
The figure 4.19 illustrates the spatial positioning of the spikes and the table 4.3 summarizes the 3D angles between them.

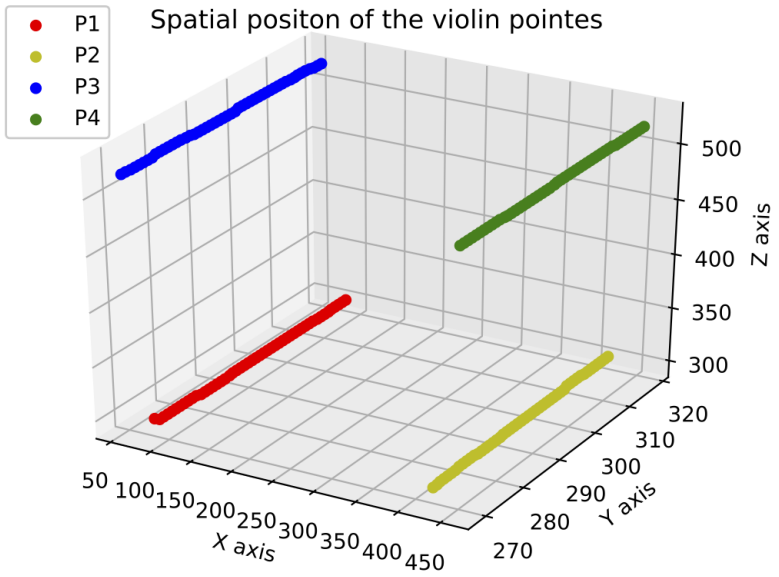


Figure 4.19: Spatial location of the spikes

Angle	**P1-P2**	**P1-P3**	**P1-P4**	**P2-P3**	**P2-P4**	**P3-P4**
3D Angle	$-2.57°$	$2.33°$	$-1.12°$	$-3.17°$	$1.45°$	$-2.52°$

Table 4.3: 3D angles between spikes

Detailed tables of the angles calculated for six different instruments are presented in appendix A.3, A.2 and A.1.

Chapter 5

Comparison

In this last chapter we will present and comment on some of the graphs that our program has created. These figures are really the final outcome of this research: they will help luthiers and musicologists to form a more objective opinion regarding the various questions they may ask themselves.
We decided to compare:

- A non-recut violin/alto and a cut violin/alto to show the channel's behaviour.
- An instrument with a very symmetrical higher area and an instrument with a low symmetry.
- An instrument whose inflection line is closer to the edge and another whose inflection line is further from the edge.
- An instrument with regular and almost parallel spikes and another with very irregular spikes.

Unfortunately, we will not be able to comment on all the graphs of all the instruments scanned in this text. That being said, we have nevertheless selected the most interesting cases to allow the reader to understand what this work can lead to.

5.1 Behaviour of channel points on a recut and non-recut instrument

First, we decided to highlight the differences that may exist in the channel position for two different instruments. As we explained in chapter 2, the channel position can fade in a recut instrument. The graph 5.1 [1] shows the channel of an instrument that has probably not been recut and the figure 5.2 [2] shows that of an instrument that has probably been recut.

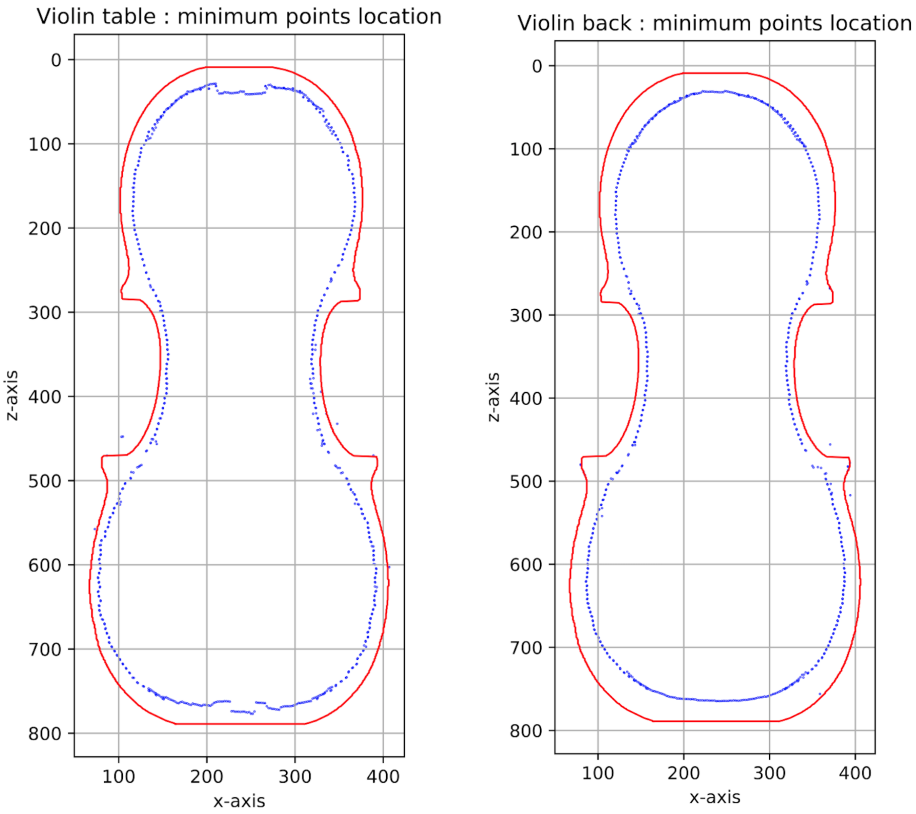


Figure 5.1: The location of the channel points (blue) for the table (red left) and the back (red right) of a probably non recut violin

[1] These results holds for viola – Reference number : 2846 – Creator : Hofmans Matthys IV – Geography : Antwerpen (Belgium) – Date : before 1679

[2] These results holds for violin – Reference number : 1355 – Creator : Hofmans Matthys IV – Geography : Antwerpen (Belgium) – Date : before 1679

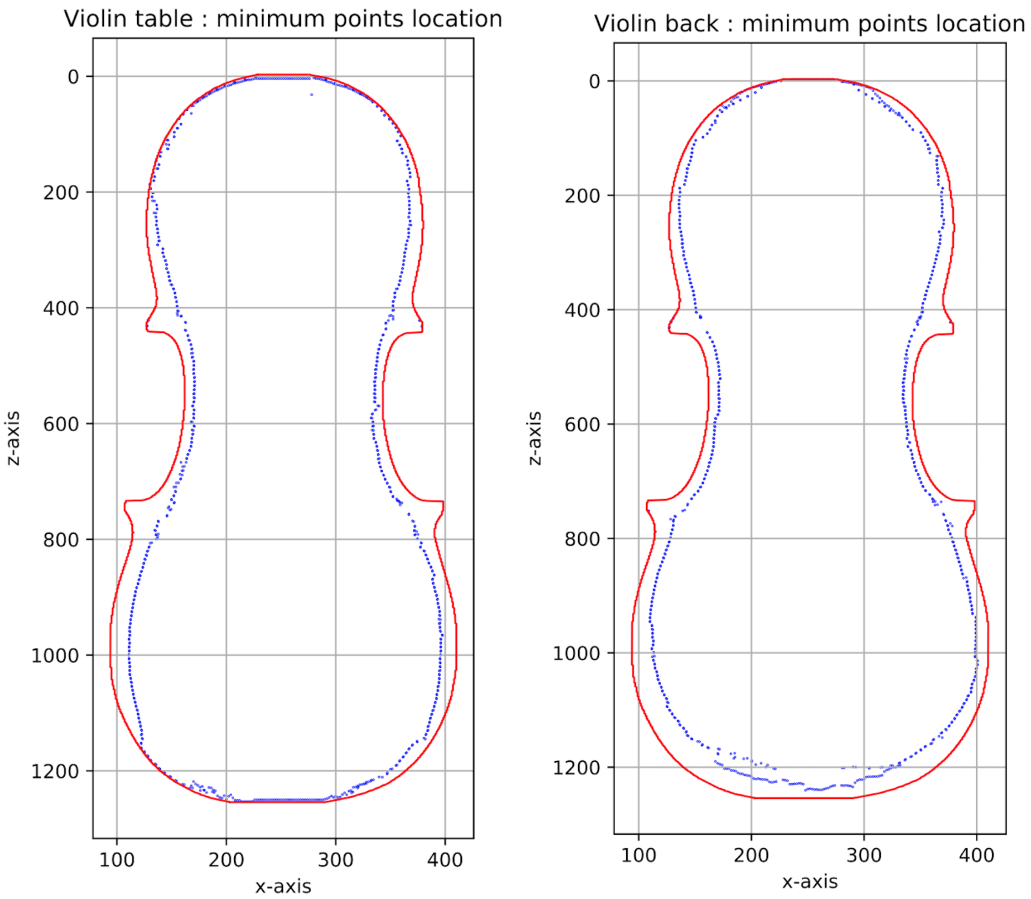


Figure 5.2: The location of the channel points (blue) for the table (red left) and back (red right) of a probably recut violin

The difference observed between the position of the channel was really the primary objective of this research. Thanks to this type of graphics, luthiers and musicologists will be able to have at their disposal a strictly scientific result proving the channel's behaviour around the entire instrument.

5.2 Behaviour of the maximum zone for two different instruments

A second criterion for comparison between two instruments was to identify the difference in symmetry between the two maximum zones defined in section 4.2 which could indicate that the instrument had suffered deformation.
To do this we present the figure 5.3 (left) [3] with a symmetrical zone and the figure 5.3 (right) [4] with a non-symmetrical zone.

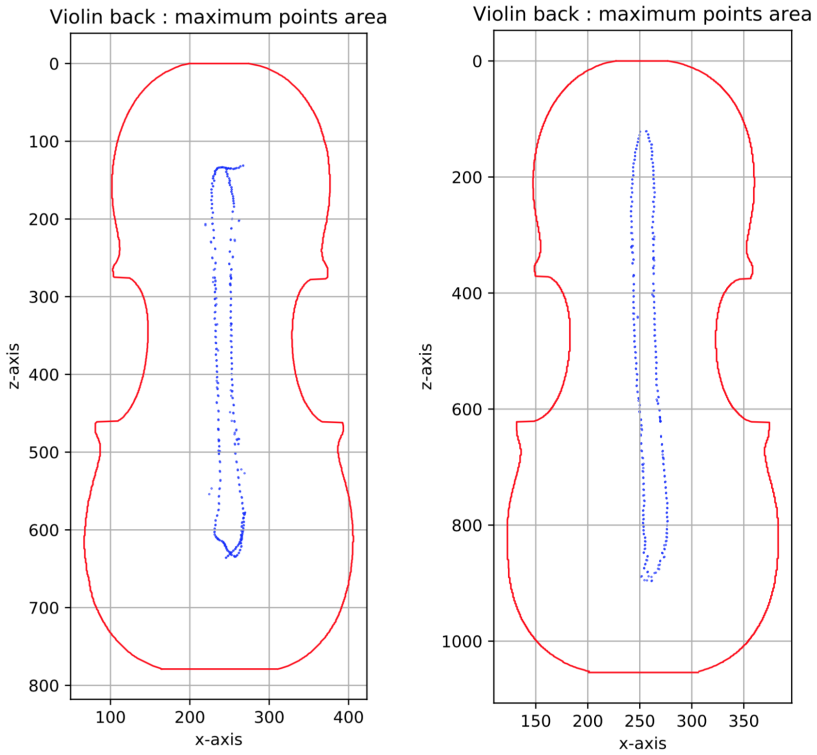


Figure 5.3: The location of the maximal zone (blue) for two instruments

Thanks to these two graphs we can notice the differences that can appear when analyzing the maximum area. In the first graph the area is quite symmetrical while in the second it shifts to the right on the bottom of the violin. We also notice that the area is much more stretched on the right graph.

[3] These results holds for violin – Reference number : 1355 – Creator : Hofmans Matthys IV – Geography : Antwerpen (Belgium) – Date : before 1679

[4] These results holds for violin – Reference number : 2792 – Creator : Hofmans Matthys IV – Geography : Antwerpen (Belgium) – Date : 1665

5.3 Behaviour of the inflection zone for two different instruments

Another feature that allows us to compare two instruments to be used is the position of the inflection points on the table or back. To our knowledge no one had ever looked at this stylistic characteristic, so it was interesting to analyze it. We therefore decided to show the differences that can appear by representing the inflection zone of two instruments. The figure 5.4 (left) [5] represents an instrument with an inflection line closer to the edge, and the figure 5.4 (right) [6] represents another whose inflection line is further away from the edge.

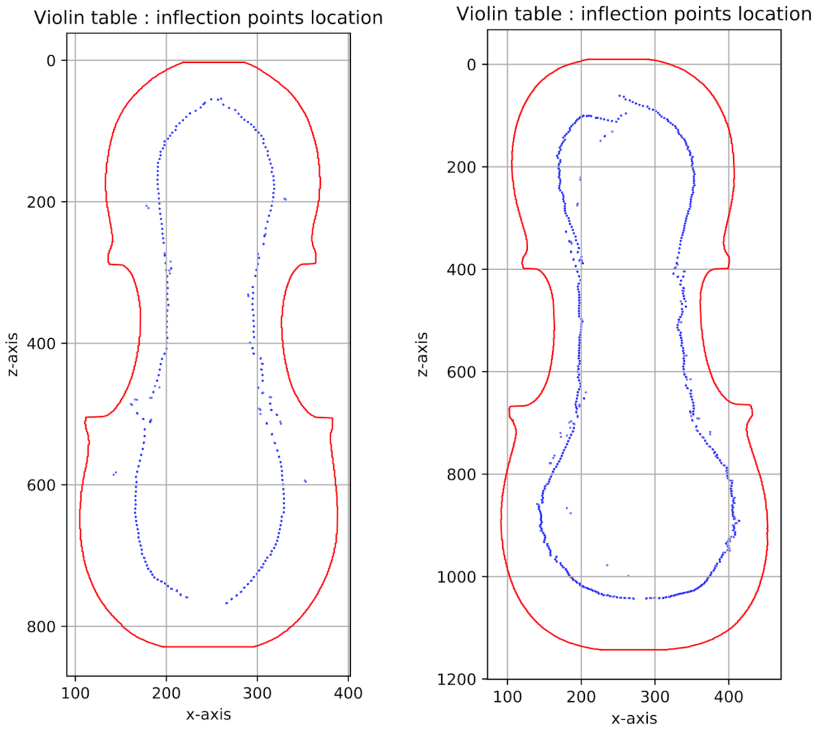


Figure 5.4: The location of the inflection zone (blue) for two instruments

It is important to note that the computation of inflection points is generally not as accurate as that of minimum points. The position of a given inflection point is indeed more sensitive than that of the minimum of a given cut. When we approximate the table or background by a spline, a small difference in the positioning or computation of the

[5] These results holds for viola – Reference number : 2832 – Creator : Cuypers Johannes Theodorus – Geography : La Haye (Netherlands) – Date : 1762

[6] These results holds for viola – Reference number : 2846 – Creator : Hofmans Matthys IV – Geography : Antwerpen (Belgium) – Date : before 1679

latter impacts the position of the inflection point a little more than that of the minimum point.
Despite the risk of errors and inaccuracy, we still get a significant look. But we can still notice imprecise results as on the upper part of the figure 5.4 (right), which still need to be improved.

5.4 Behaviour of the spikes angles for two different instruments

The last characteristic that could be analyzed via our Python program was the degree of parallelism that can exist between the four spikes of an instrument. To do this we compared the figure 5.5 (left) [7] which represents an instrument whose spikes are regular and parallel to figure 5.5 (right) [8] from another instrument with very irregular spikes.

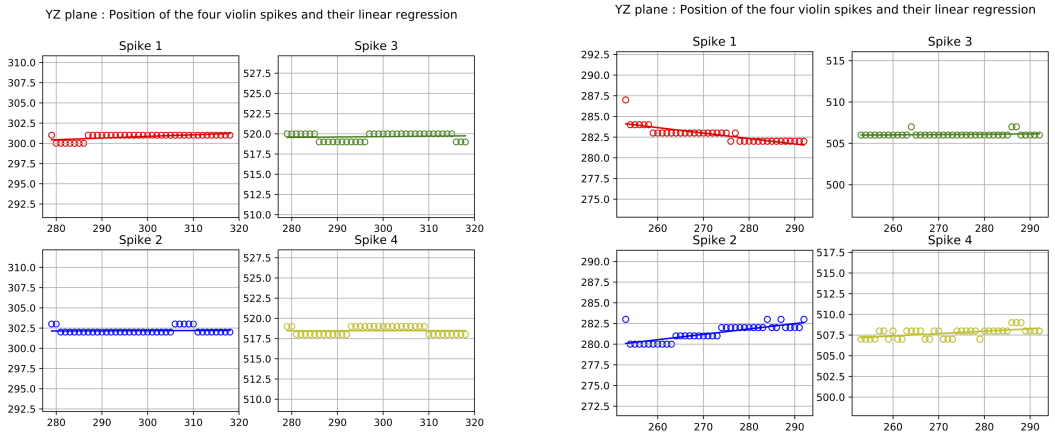


Figure 5.5: Projection of the four spikes on the YZ plane for two instruments

A graph like the one on the right where the spikes are very irregular can have several origins : a defect, a harlequin, deformations in the wood... Despite the fact that our results do not reveal any major trends, this makes it possible to show how far we could go in research if we had a larger number of violins at hand to examine.

[7] These results holds for viola – Reference number : 2833 – Creator : Cuypers Johannes Theodorus – Geography : La Haye (Netherlands) – Date : 1761

[8] These results holds for violin – Reference number : 2848 – Creator : Hofmans Matthys IV – Geography : Antwerpen (Belgium) – Date : before 1679

Conclusion

Each instrument in the violin family is unique and there are many characteristics that can define a given instrument. As we have seen, these instruments have seen and endured centuries, and some of them have certainly been recut along the way. This research responds to a specific request from a musicologist, Mrs. Anne-Emmanuelle Ceulemans, confronted with the study of instruments that have always been built in a traditional way and are have never been perfectly standardized (especially in the 17th and 18th centuries). The challenge taken up by this thesis was therefore - among other things - to overcome the issues inherent to non-standardization with the help of modern tools.

To achieve this, we developed a Python program of about 2800 lines of code that works nearly autonomously : only a small preprocessing task has to be performed manually in order to obtain the expected results. This program is able to provide several external characteristics of a given instrument and the results are encouraging. The only limitation in its use is of course the technique used to obtain scans. This technique is not easy to replicate for a very large number of instruments and is therefore not ideal.
It should also not be forgotten that this work analyzed the characteristics of instruments made of organic matter, which is not an easy task from a computer point of view: wood moves, deforms, is not smooth,... As a consequence, some of our results, such as the determination of the inflection point area, are approximate although still very consistent with what was expected. It was therefore necessary to think in a really nuanced and critical way about these results, as it is essential with such research topic.
One way to improve our results could be to find the entire spline of channel points globally. This would require solving a "global" problem that is certainly more difficult to solve but could give us even more precise results.

This thesis highlighted many results that need to be analyzed from an organological point of view. The program carried out is a powerful tool, but one must bear in mind that all these results require careful interpretation in order to draw any conclusions on the given instruments. The only results that we know how to interpret clearly are of course those showing the position of the channel points. From an engineering point of view, we can observe what is happening thanks to all the characteristics given, but we are not in

an ideal position when it comes to explain these phenomena which are better suited as topic of an organological study
This work also shows the strength of a multidisciplinary approach, as well as the importance of dialogue and openness. During the preparation of this thesis, the continuous collaboration with Mrs. Anne-Emmanuelle Ceulemans who has provided her opinion as a musicologist on all the results obtained was inestimable. This has made it possible to better target expectations and create a computer program that best meets the needs of musicologists.
It was also very important to write a thesis that can be read by a large number of people from different backgrounds. Aside from the fact that the content of the program is purely computer-based, the final result is a tool that can easily be used by organologists.

This research, in addition to presenting results that can be used by musicologists and luthiers, highlights potential future developments in the field.
It should also be noted that all the instruments we have analyzed are quite old. We can therefore imagine that if we were to analyze more recent instruments, the analysis procedures could be very powerful and more accurate.
What is more, many instrument scans carried out in Saint-Luc could not be used immediately because of the artifacts caused by metallic butterfly nuts. We can therefore of course imagine that if a lead were to be found to treat these artifacts, these violins and violas could be analyzed, and external characteristics could be highlighted. Finally, to show all possible opening paths, we will refer to the figure 3.1. The colors in this image reflect densities. It's quite interesting because it reminds us of the techniques used by Geerten Verberkmoes in the article [12]. With the scanned images we may therefore have much more to investigate in the future than the topics this thesis covered.

List of Figures

List of Tables

Appendix A

Graphs and tables

The graphics of all the violins scanned in Saint-Luc are available via the following link. These graphs contain the following characteristics: Channel points, inflection points, maximum points, spikes parallelism.

https://www.dropbox.com/sh/7lm70eryr7xsloa/AAB5rGrEpIKaDDdTC0QJGfcSa?dl=0

In tables A.3, A.2 and A.1, we present all the values of the spike angles for the violins used :

3D Angle	P1-P2	P1-P3	P1-P4	P2-P3	P2-P4	P3-P4	Mean
Scan 8	$-3.32°$	$-5.05°$	$-2.16°$	$7.04°$	$1.35°$	$5.77°$	$4.12°$
Scan 9	$-4.17°$	$1.19°$	$-1.44°$	$-3.23°$	$3.12°$	$-0.26°$	$2.24°$
Scan 10	$2.23°$	$4.71°$	$6.24°$	$6.9°$	$8.3°$	$3.94°$	$5.39°$
Scan 11	$-7.49°$	$7.03°$	$-5.46°$	$-6.51°$	$2.14°$	$-6.33°$	$5.82°$
Scan 12	$-6.45°$	$7.72°$	$-5.3°$	$-2.11°$	$1.18°$	$-2.7°$	$4.24°$
Scan 13	$-2.57°$	$2.33°$	$-1.12°$	$-3.17°$	$1.45°$	$-2.52°$	$2.19°$
Mean	$4.37°$	$4.67°$	$3.62°$	$4.83°$	$4.67°$	$2.92°$	**4.18**$°$

Table A.1: 3D angles between spikes

2DX Angle	P1-P2	P1-P3	P1-P4	P2-P3	P2-P4	P3-P4	Mean
Scan 8	3.24°	0.84°	2.16°	4.08°	1.08°	3°	2.4°
Scan 9	4.03°	0.8°	0.91°	3.23°	3.12°	0.1°	2.03°
Scan 10	1.21°	3.37°	0.71°	4.58°	1.93°	2.65°	2.39°
Scan 11	0.56°	1.17°	0.19°	1.73°	0.37°	1.36°	0.9°
Scan 12	1.28°	0.91°	1.9°	0.37°	0.62°	0.99°	1.01°
Scan 13	0.9°	2.02°	0.5°	2.92°	0.41°	2.51°	1.57°
Mean	1.87°	1.52°	1.06°	2.5°	1.25°	1.768°	**1.66**°

Table A.2: 2DX angles between spikes

2DZ Angle	P1-P2	P1-P3	P1-P4	P2-P3	P2-P4	P3-P4	Mean
Scan 8	0.76°	4.98°	0.05°	5.75°	0.81°	4.93°	2.88°
Scan 9	1.06°	0.88°	1.12°	0.19°	0.05°	0.24°	0.59°
Scan 10	1.87°	3.29°	6.2°	5.16°	8.07°	2.91°	4.58°
Scan 11	7.49°	3.98°	5.45°	3.5°	2.04°	1.47°	3.99°
Scan 12	6.42°	7.63°	5.3°	1.21°	1.12°	2.33°	4°
Scan 13	2.41°	1.16°	1.01°	1.24°	1.4°	0.16°	1.23°
Mean	3.33°	3.65°	3.18°	2.84°	2.25°	2.01°	**2.88**°

Table A.3: 2DZ angles between spikes

where the following references hold

Scan	Name	Reference number
Scan 8	Viola Cuypers	2832
Scan 9	Viola Cuypers	2833
Scan 10	Viola Hofmans	2846
Scan 11	Viola Hofmans	2848
Scan 12	Violin Hofmans	2792
Scan 13	Violin Hofmans	1355

Table A.4: Meaning of the scan reference number

To explain the precise vocabulary used, we present figure A.1 issued from `http://www.williamsfineviolins.com/wp-content/uploads/2014/09/Parts-Poster.jpg`.

Figure A.1: Vocabulary used

Appendix B

Python codes and timings

Function	**Elapsed time**
GotScan	$\approx 60sec$
OriginalFirst	$\approx 4sec$
Normalize	$\approx 65sec$
thresholdotsu	$\approx 10sec$
second	$\approx 15sec$
DenoiseBlack + DenoiseWhite	$\approx 90sec$
Cut	$\approx 1sec$
StartFinish	$\approx 120sec$
FindImage	$\approx 7sec$
Contour	$\approx 80sec$
ObtainTableFond	$10sec(\times 4)$
AngleSkief	$\approx 4sec$
AngleLast	$\approx 2sec$
GetMinimumExt	$\approx 40sec$
GetMinMax	$\approx 30sec(\times 6)$
GetInflexion	$\approx 70sec(\times 2)$
GetInflexionExt	$\approx 7sec$
GetMinMax	$\approx 30sec(\times 6)$
ComputeSpikes	$\approx 7sec$
ComputeAngles	$\approx 0.1sec$
ComputeDistanceToMin	$\approx 5sec$
Total	$\boldsymbol{\approx 1000sec \approx 17min}$

Table B.1: Elapsed time for each function of our program for a given violin

The Python code used is available via the following link:

https://www.dropbox.com/sh/7lm70eryr7xsloa/AAB5rGrEpIKaDDdTC0QJGfcSa?dl=0

We present here a descriptive part of our code with the inputs, outputs and descriptions of some key functions :

```
"""
#############################################################
                MEMO2990 : TRAVAIL DE FIN D'ETUDES
#############################################################
                  Characterization of violins :
             a digital tool at the service of organology

                   Author : Lothaire Renaud
#############################################################
"""

def UncommentScan(path):
    """
       Script that stores the DICOM scan under Python format
       IN : path = path of the DICOM file
    """
    # Function to use the adequate "load" function to load
    # DICOM scan onto Python
    Scan,NScan,NFile = GotScan('path',NScan,NFile)

def TreatImage():
    """
       Script that treat the violin image and denoises it
    """
    # Function that extracts an array of pixels from
    # the DICOM loaded files
    OriginalFirst = GetPixels(Scan,NScan)
    # Function that normalizes pixel intensities from an
    # image onto a scale from 0 to 1000
    first = Normalize(OriginalFirst)
    # Function non robust which transforms some parts of
    # the image to become white
    first = Cut(NScan,NFile,first)
    # Function which finds the first and last image
```

```
    # representing the violin
    start,finish = StartFinish(NFile)
    # We cut the image in order to keep only the violin body
    first = first[start:finish]
    # Function which finds the threshold under which pixels
    # will become white
    thresh = threshold_otsu(first)
    # We keep the pixels above 0.5*thresh in black, other
    # become white
    binary = first > 0.5*thresh
    second = np.ones(( len(first[:,1,:]),len(first[1,:,:]),
                  len(first[1,:,:]) ))*1000
    second[binary] = 0
    # Function that denoises an image with respect to the
    # neighbour of each pixel
    four = np.ones(( len(second[:,1,:]),len(second[1,:,:]),
                len(second[1,:,:]) ))
    for i in range(len(second[:,1,:])):
        four[i] = DenoiseBlack(second[i],555)
        four[i] = DenoiseWhite(four[i],445)

def TreatViolin():
    """
        Script that treat violin shape by keeping only
        interesting parts
    """
    # Function which returns a 3D image with points between
    # fixed limits that becomes white
    NewImage = FindImage(four,NScan,NFile,start,finish)
    # We store the coordinates of the violin into a 3D vector
    (z,y,x) = np.where(NewImage==0)
    # Function that deletes the violin interior
    FinalX = []
    FinalY = []
    FinalZ = []
    for i in range(len(four)-2):
        NewX,NewY = Contour(x[z==i],y[z==i])
        FinalX = np.concatenate((FinalX,NewX))
        FinalY = np.concatenate((FinalY,NewY))
        n = len(NewX)
```

```
        FinalZ = np.concatenate((FinalZ,np.ones((n))*i ))

def TreatRotation():
    """
       Script that applies rotations to the violin to
       align it with axes
    """
    # Function which finds angles to apply in order to
    # align the violin with axes
    AngleYZ = AngleLast(x,y,z)
    # Function that returns a rotated image w.r.t 2D angle
    yRot,zRot = Rotate(np.arctan(AngleYZ),y,z)
    # Function that returns the violin back of an xyz data set
    semi = np.mean(y)
    UpperFondX,UpperFondY,UpperFondZ =
    ObtainTableFond2(0,len(four),x[yRot > semi],
                  yRot[yRot > semi],zRot[yRot > semi],"table")
    # Function which finds angles to apply in order to align
    # the violin with axes
    AngleXZ,AngleXY = AngleSkief(x,yRot,zRot,UpperFondX
          ,UpperFondY,UpperFondZ,FinalX,FinalY,FinalZ)
    # Function that returns a rotated image w.r.t 2D angle
    xRot,zRot2 = Rotate(-np.arctan(AngleXZ),x,zRot)
    # Function that returns a rotated image w.r.t 2D angle
    xRot2,yRot2 = Rotate(-np.arctan(AngleXY),xRot,yRot)
    # Function which returns the rotated violin table
    TableXNew,TableYNew,TableZNew = NewTable(TableX,TableY
                    ,TableZ,AngleXZ,AngleXY,AngleYZ)

def PreliminaryRun():
    """
       Script that computes preliminary stuff
       which is usefull for min/max/PI
    """
    # Function which returns a boolean vector to keep only part
    # from a matrix Rot between limit-delta and limit+delta
    maskZ = ReturnMaskForXYZ(zRot2,0.5,int(np.mean(zRot2)))
    maskY = ReturnMaskForXYZ(yRot2,0.5,int(np.mean(yRot2)))
    maskX = ReturnMaskForXYZ(xRot2,0.5,int(np.mean(xRot2)))
    # Function that keeps extreme points in xz direction and
```

```
    # then returns the derivative function for all the
    # extreme points, its absicae points and eventually the
    # splines that approximates the function
    tanUp,tanDown,xs,xs1,spl,spl1 = FindDirection(
          xRot2[tuple(maskY)],z[tuple(maskY)])
    where = int(np.floor(len(tanUp)/1000))
    # Function which computes all the minimums along the
    # right and left violin parts
    plotFinalExt,plotFinaExt = GetMaxExteriorXZExt(
          xRot2,zRot2,spl1(xs1)[0],spl(xs)[0])
    # Function which computes all the minimums along the
    # right and left violin parts
    plotFinal,plotFina = GetMaxExteriorXZ(xRot2,zRot2)

def ComputeChannel():
    """
       Script that finds the channel location on the
       violin table/back
    """
    # Function which computes all the minimums along the
    # upper and lower violin parts
    Tem1,Tem2,Tem3,Tem4,Tem5,Tem6 = GetMinimumExt(
          xRot2,yRot2,zRot2,maskY,xs1,spl1,xs,spl,table='table')
    # Function which computes all the minimums along the
    right and left violin parts
    _,Temp22,_,Temp44,_,Temp66,Precc = GetMinMax(
          'xs',xs,spl,xRot2,zRot2,where,mult,0,tanUp,
          yRot2,maskY,'min',table='table')
    Temp11B,_,Temp33B,_,Temp55B,_,Precc1 = GetMinMax(
          'xs1',xs1,spl1,xRot2,zRot2,where,mult,0,
          tanDown,yRot2,maskY,'min',table='table')


def ComputeInflection():
    """
       Script that computes the inflection points all
       along the violin border
    """
    # Function which computes all the IP along the upper
    # and lower violin parts
```

```
    T1,T2,T3,T4 = GetInflexionExt(xRot2,yRot2,zRot2,0,xs,spl
                            ,xs1,spl1,table = 'table')
     # Function which computes all the IP along the right
     # and left violin parts
    _,TempI22,_,TempI44,_,TempI66 = GetInflexion(
          xs,spl,xRot2,zRot2,where,0,tanUp,yRot2,maskY
          ,mult,table='table')
    TempI11B,_,TempI33B,_,TempI55B,_ = GetInflexion(
          xs1,spl1,xRot2,zRot2,where,0,tanDown,yRot2,maskY,
          mult,table='table')


def ComputeMaximumArea():
    """
       Script that computes the maximum area from
       the violin table/back
    """
    # Function which computes all the maxmimum
    # along the right and left violin parts
    _,Te22,_,Te44,_,Te66,_ = GetMinMax(
          'xs',xs,spl,xRot2,zRot2,where,mult,0,tanUp,
          yRot2,maskY,'droite',add,table='fond')
    Te11B,_,Te33B,_,Te55B,_,_ = GetMinMax(
          'xs1',xs1,spl1,xRot2,zRot2,where,mult,0,tanDown,
          yRot2,maskY,'gauche',add,table='fond')

def ComputeSpikes():
    """
       Script that computes the spikes locations
    """
    # Function that finds 2D coordinates of the
    # spikes from a 3D image by averaging each
    # coordinates of the spikes from each y-cut
    _,_,_,PointesAll = FindCorners(
          xRot2,yRot2,zRot2,40,int(np.mean(yRot2)-20))
    # Function which find the linear regression of a given spike
    fit1X,_,rank1,_,_ = np.polyfit(
          PointesAll[:,8],PointesAll[:,0],1,full=True)
    # Repeat for all the spikes
```

```
def ComputeAngles():
    """
       Script that computes 2D and 3D
       angles between the spikes
    """
    # Function to find the 3D angle between
    # d1 and d2 with fit containing slopes 2D
    angle3D = np.array(([Parallel(fit,1,2),
                    Parallel(fit,1,3),Parallel(fit,1,4),
                    Parallel(fit,2,3),Parallel(fit,2,4),
                    Parallel(fit,3,4)]))
    # Function to find the 2D angle between
    # d1 and d2 with fit containing slopes 2D
    angle2DX = np.array(([Parallel2D(fit,1,2,'x'),
                     Parallel2D(fit,1,3,'x'),Parallel2D(fit,1,4,'x'),
                    Parallel2D(fit,2,3,'x'),Parallel2D(fit,2,4,'x'),
                    Parallel2D(fit,3,4,'x')]))
    # Function to find the 2D angle between
    # d1 and d2 with fit containing slopes 2D
    angle2DZ = np.array(([Parallel2D(fit,1,2,'z'),
                     Parallel2D(fit,1,3,'z'),Parallel2D(fit,1,4,'z'),
                    Parallel2D(fit,2,3,'z'),Parallel2D(fit,2,4,'z'),
                    Parallel2D(fit,3,4,'z')]))

def ComputeDistanceToMin():
    """
       Script in order to compute the distance
       to the minimum all along the violin
    """
    RealX1 = np.linspace(int(np.ceil(min(zRot2)))
    ,int(np.floor(max(zRot2))),num=len(plotFinal))
    # We compute the spline approximation of the
    # given part of the violiin
    spll = UnivariateSpline(RealX1,plotFinal)
    xss1 = np.linspace(min(RealX1),max(RealX1),num=len(RealX1))
    tanUp = np.gradient(xss1,spll(xss1))
    # Function which finds distances between
    # border points and minimum points
    for i in range(len(xss1)):
       Precis1[i] = Precision(i,spll(xss1)
```

```
    ,xss1,0,tanUp,np.concatenate((Temp4,Tem4,Tem3))
    ,np.concatenate((Temp2,Tem2,Tem1)))
# Repeat for all the four parts of the violin
```

Bibliography

[1] Stephen Bonta et al. *Violoncello*. Jan. 2001. URL: https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000044041.

[2] David D. Boyden and Ann M. Woodward. *Viola*. Jan. 2001. URL: https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000029438.

[3] David D. Boyden et al. *Violin*. Jan. 2001. URL: https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000041161.

[4] A-E. Ceulemans. *De la vièle médiévale au violon du XVIIe siècle. Étude terminologique, iconographique et théorique, Tours, Centre d'études supérieures de la Renaissance - Turnhout*. Brepols, 2011, p. 117.

[5] Distinguished Violinmaker. Edgar Russ. *Video Violinmaking High Definition*. 9 Mar. 2016. URL: www.youtube.com/watch?v=nTp46vARaJ8&t=1s..

[6] "Haine Malou and Nicolas Meeùs." *Dictionnaire Des Facteurs D'instruments De Musique En Wallonie Et à Bruxelles Du 9e Siècle à Nos Jours*. Liège : Mardage, 1986.

[7] Darcy Mason and pydicom contributors. *Read DICOM directory*. URL: https://pydicom.github.io/pydicom/dev/auto_examples/input_output/plot_read_dicom_directory.html.

[8] Medixant. *RadiAnt DICOM Viewer*. Version 4.6.5. May 23, 2018. URL: https://www.radiantviewer.com.

[9] K. Moens. *Les voix médianes dans l'orchestre français sous le règne de Louis XIV: les instruments conservés comme source d'information.* Orchestre À Cordes Sous Louis XIV. Paris, Vrin, 2015, pp. 119–138.

[10] Laeticia Motte. *Conception d'un outil mathématique/informatique pour la classification d'instruments à archet*. Master Thesis, UCL/EPL, 2017, prom P. Fisette and A,-E Ceulemans.

[11] the Strad. *December 2018, Vol.129, NO.1544*. thestrad.com, since 1890.

[12] Geerten Verberkmoes et al. "An inside look at four historical violins by Brussels makers". eng. In: *GALPIN SOCIETY JOURNAL* 69 (2016). Ed. by Lance Whitehead, pp. 109–136. ISSN: 0072-0127.

[13] Guido Zuidhof. *Full Preprocessing Tutorial*. 2017. URL: `https://www.kaggle.com/gzuidhof/full-preprocessing-tutorial`.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**