Improving OMR for Digital Music Libraries with Multiple Recognisers and Multiple Sources. *(User manual)*

1.- Introduction

This manual describes the application made for the project "Optical Music Recognition from Multiple Sources" carried on at Lancaster Institute for the Contemporary Arts (LICA) and University of Leeds. The next paragraphs explain step by step the menus and the underling procedures for each option.

The application is made in python and the interface is written using the wxPython library compiled by py2exe. The source could be compiled in any other platform and even executed by command line using the MainMultiOMR library developed ad hoc. (See the documentation for more information)

The main menu options are:

- **Automatism**: runs the sikuli automatic process for each OMR
- **Process**: calculates the final result based on the different input files
- **Results**: writes xls files comparing the output with the ground file
- **Utils**: several options for checking the system
- **Help**: help files and documentation

2.-Hardware Configuration

The project has been installed at Lancaster University configuring 6 virtual machines focused on different tasks. The next picture represents the global scheme:

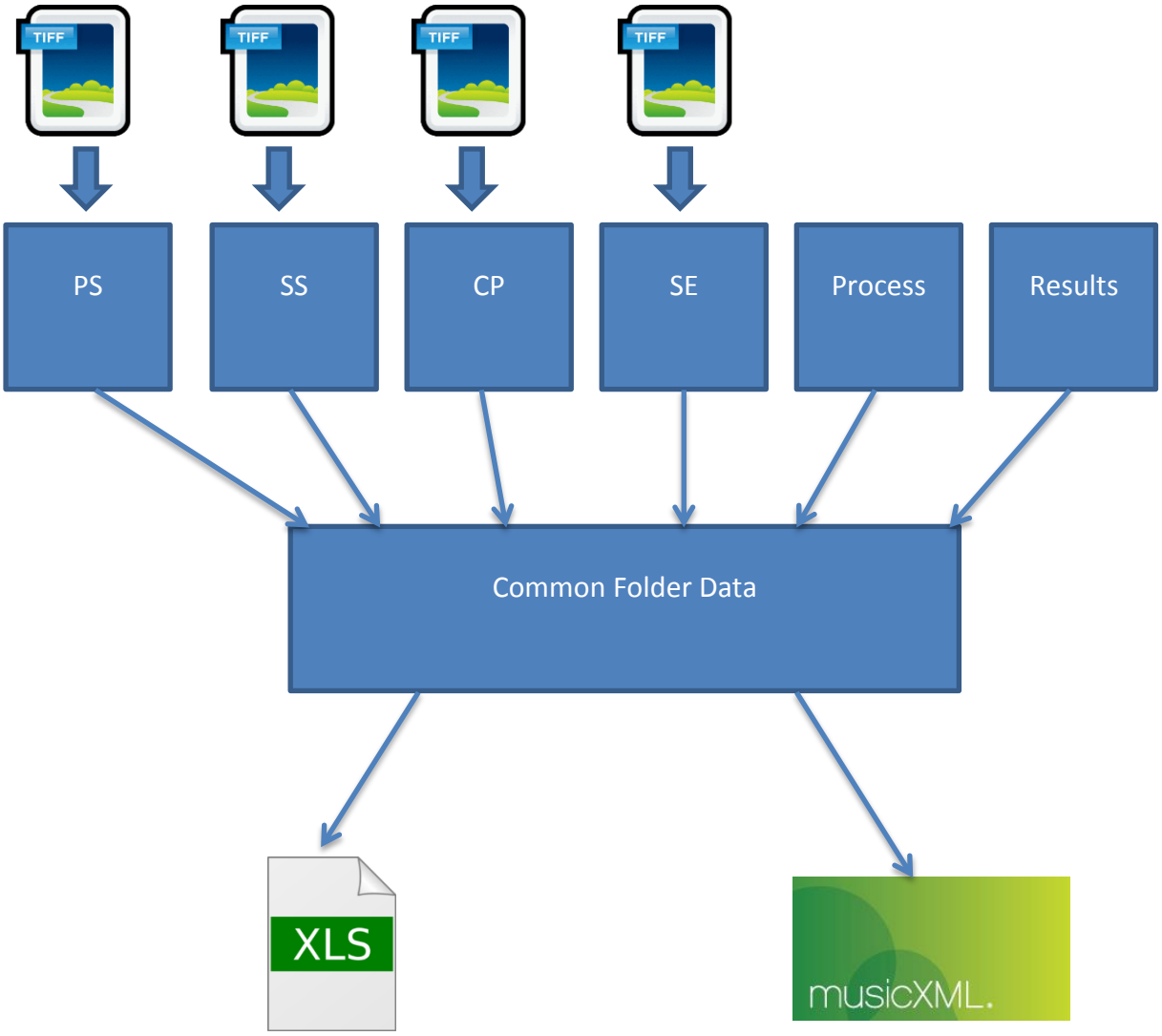


Figure 1: General Hardware Configuration

The first four machines are dedicated to convert tiff images to musicXML inputs. The last two process and evaluate the results.

3.-Automatism Menu

The first four options are related to converting each tiff folder to MusicXML inputs. The application search for .tiff files in the tree subfolder. The images inside each folder are processed as a movement or a complete score, ordered alphabetically.

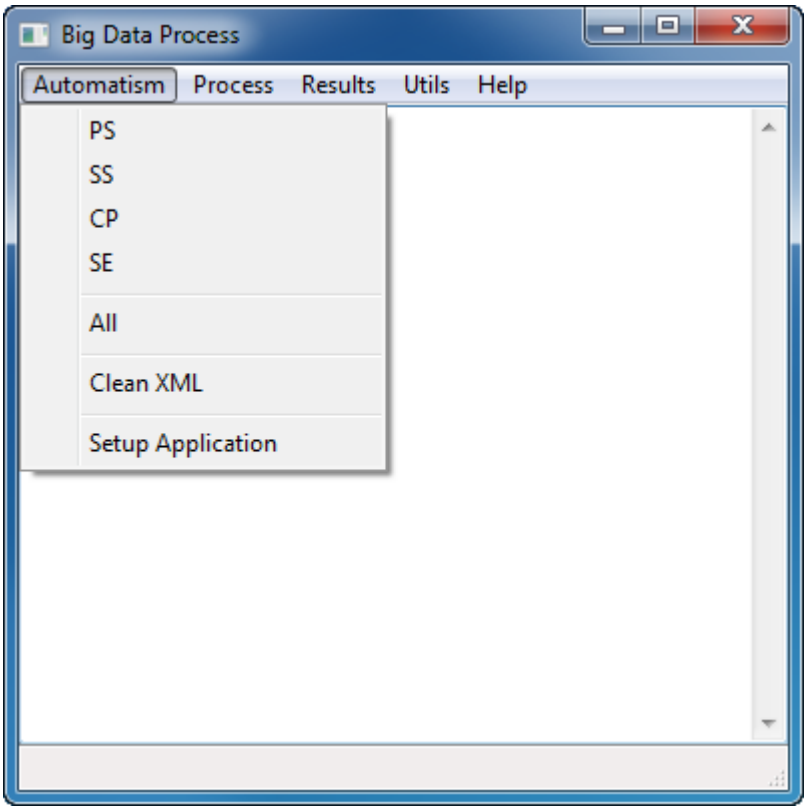


Figure 2: Automatism Menu

- **PS**: means PhotoScore software. The machine dedicated at Lancaster University is **lica-omr2.lancs.ac.uk**
- **SS**: means SmartScore software. The machine dedicated at Lancaster University is **lica-omr3.lancs.ac.uk**
- **CP**: means Capella software. The machine dedicated at Lancaster University is **lica-omr4.lancs.ac.uk**
- **SE**: means SharpEye software. The machine dedicated at Lancaster University is **lica-omr1.lancs.ac.uk**

The program calls the sikuli[1] automatic software. The virtual JAVA machine[2] is needed for running these options

3.1.-All (option)

This option sequentially calls the different sikuli process associated with the OMR software for running the complete automatic process just in one machine. For doing that, we have to install the four OMR software's in a single computer. The main drawback for this option is the time consuming process in a large scale.

3.2.-Clean XML (option)

This is an auxiliary option just for removing all the xml in the tree folder. It is useful if an automatic process restart is required.

3.3.-Setup Application (option)

In the data folder, we will have two subfolders for each piece: **OMRS** and **Process**

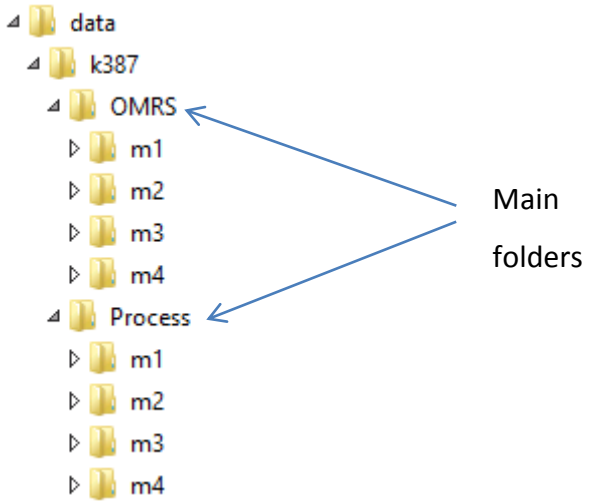


Figure 3: Folders detail. The Process *folder is created based on* OMRS *files*

[1] http://www.sikuli.org/
[2] http://java.com/en/download/

The meaning of each folder is:

- **Data**: The root folder. The name can be whatever
- **K387**: The opus number or the piece name. The name can be whatever
- **OMRS**: The main folder. Here we have to put the tiff files and get the xml files processes. This name is compulsory and fixed. The tree structure under this name is typically the movement folders
- **Process**: The structure of *OMRS* is duplicated for processing and results in the next options. The *Process* folder is created based on *OMRS* files

When we run the setup option, the information of OMRS folder is copied to Process folder and prepared and arranged for processing. Inside each movement we will have the **fullScore** and **parts** folders (name fixed) with the different versions (Eulenberg, Casella, Peters…). If our score is a string quartet, i.e., the **parts>(version folder)** have to be subdivided in different part subfolders (0,1,2,3…)

For string quartets 0 means Violin I, 1 Violin II, 2 Viola and 3 Cello

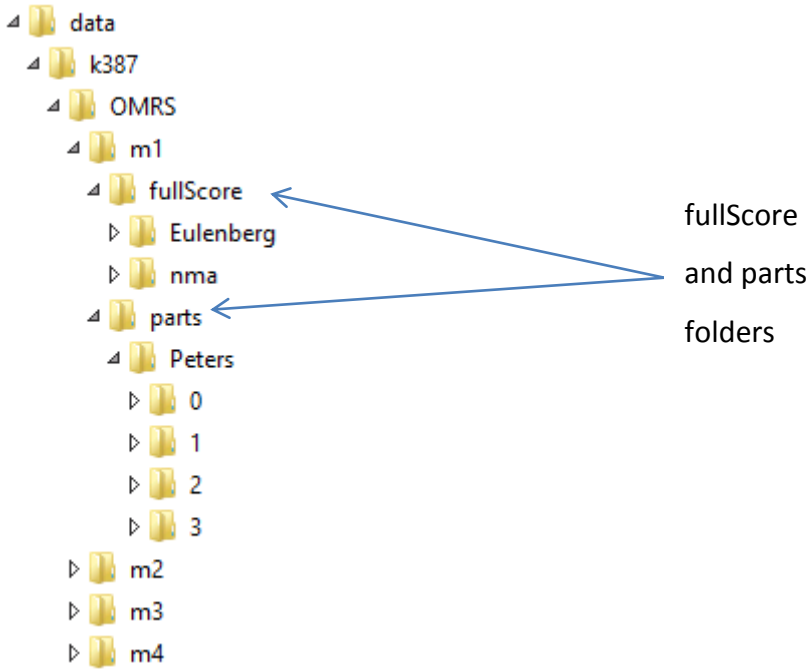


Figure 4: FullScore and Parts Subfolders detail. Typical one movement folders

For example, the next picture (figure 5) represents the information we have inside the 0 folder in Peters part. These files have to be inside one XML folder

- SS.xml, SE.xml, PS.xml and CP.xml are the xml output from the automatic sikuli process. These files will be automatically generated.
- V1-001, V1-002, V1-003 are the tiff files. These image files will be the input to the different OMR software. The names can be whatever, but they will be processed in alphabetical order
- The .mro files are auxiliary files created by SE SherpEye. Once SE.xml is created, they are not needed.

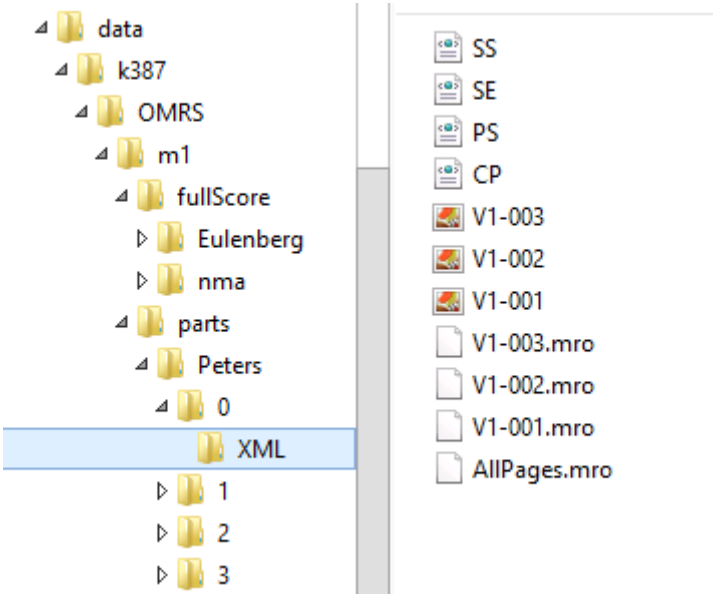


Figure 5: Files detail inside one XML folder. Parts option

For the fullScore folder, the structure is the same. The figure 6 shows this information in detail.

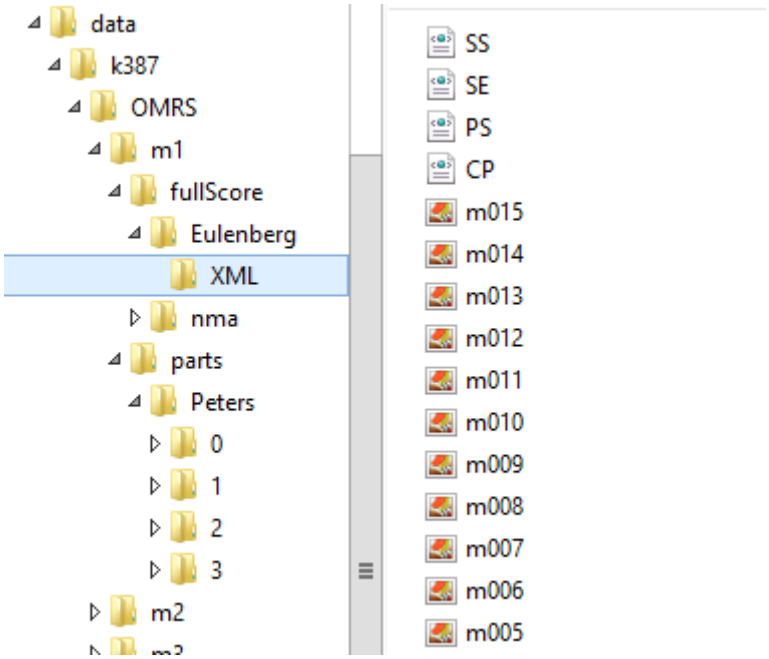


Figure 6: Detail of files inside one XML folder. FullScore option

Once the files are copied to the *Process* folder, the structure is slightly different. The tiff files are not duplicated (not necessary) and the xml files are renamed using the next scheme:

(FS or part number).(Version).(OMR software)

The tree structure of OMRS folder simplifies, but the name contains this information

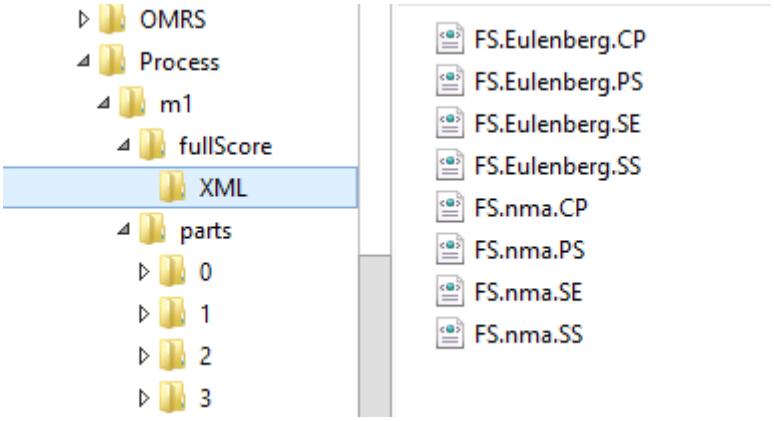


Figure 7: Detail of files inside one XML folder. Process>fullScore folder

The same structure is repeated for parts

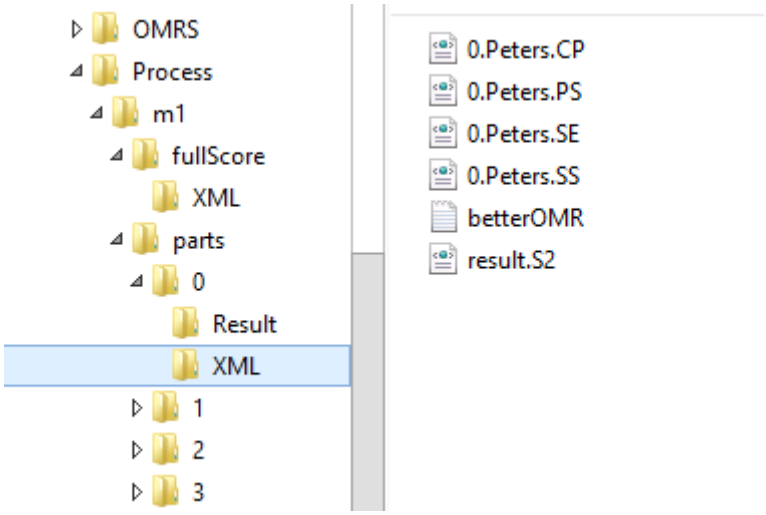


Figure 7: Detail of files inside one XML folder. Process>parts folder

In this case, **result.S2.xml** will be the result of each part. The file **betterOMR.txt** is one .txt file containing the index with the better OMRs involved. This point will be explained in the *Process Menu*.

4.-Process Menu

This menu is related to processing the output from OMR software in order to obtain one final xml (finalScore.xml)

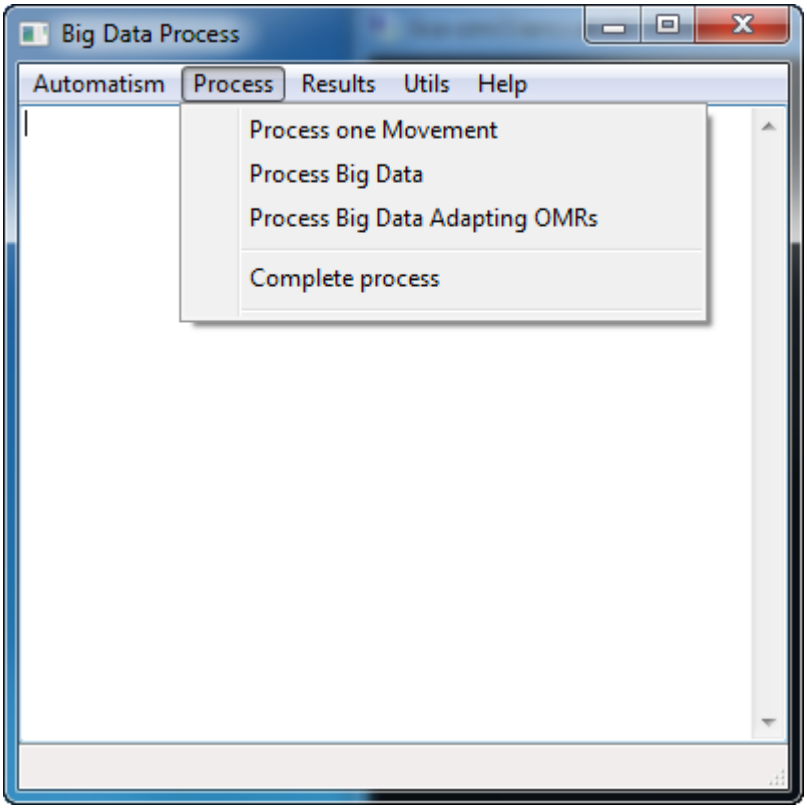


Figure 8: Automatism Menu

The different options are:

4.1.-Process one Movement (option)

For executing correctly this option, the folder to process should be one movement in the Process folder.

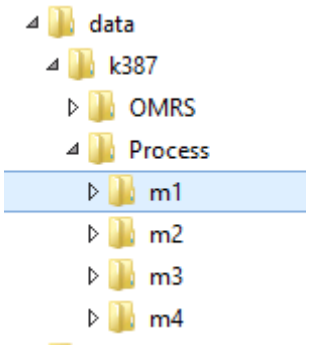


Figure 9: Folders to process. Option Process one Movement

The files that the program writes are:

- In each part folder

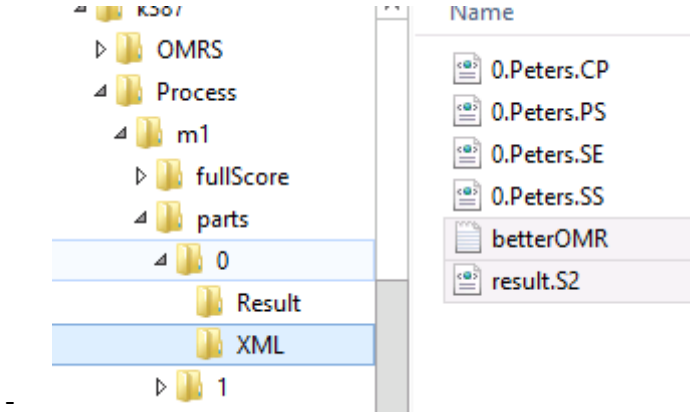


Figure 10: Option Process one Movement. Output files.

  - **result.S2.xml.** It is the part processed. The number of measures can be not correct mainly due to multiple rest measures in parts. The system will align the different result.S2.xml files with the better full score
  - **betterOMR.txt**. It is just an array of numbers that indicate the index of the files used in the process, according to the phylogenetic trees process.

For example, in this case we have in our betterOMR.txt file the numbers

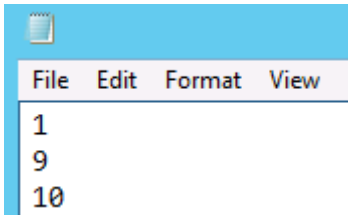


Figure 11: betterOMR.txt file detail

This means the system has taken the files 1, 9, 10 in alphabetic order. It will be useful for calculating the results.

3.Peters.C	3.Peters.P	3.Peters.S	3.Peters.S	FS.Eulenb	FS.Eulenb	FS.Eulenb	FS.Eulenb	FS.nma.CF	FS.nma.PS	FS.nma.SE
88,65672	92,09756	86,01399	90,41502	70,89479	73,28094	67,95635	78,0849	86,21032	88,98551	91,61677

Figure 12: Detail from the xls output file. The OMRs taken are in red

4.2.-Process Big Data (option)

The application runs a loop through the tree folders searching for files to process. If one movement is completed and all the .xml files have been created, the next steps are:

- Setup the application. The program copy the xml files and prepare the folders inside the Process folder
- Configure the ground. Convert .krn files to ground.xml or copy the .xml ground file
- Run the option “Process one Movement” for each movement completed

If there is not anything to process, the program waits searching for files every 5 seconds

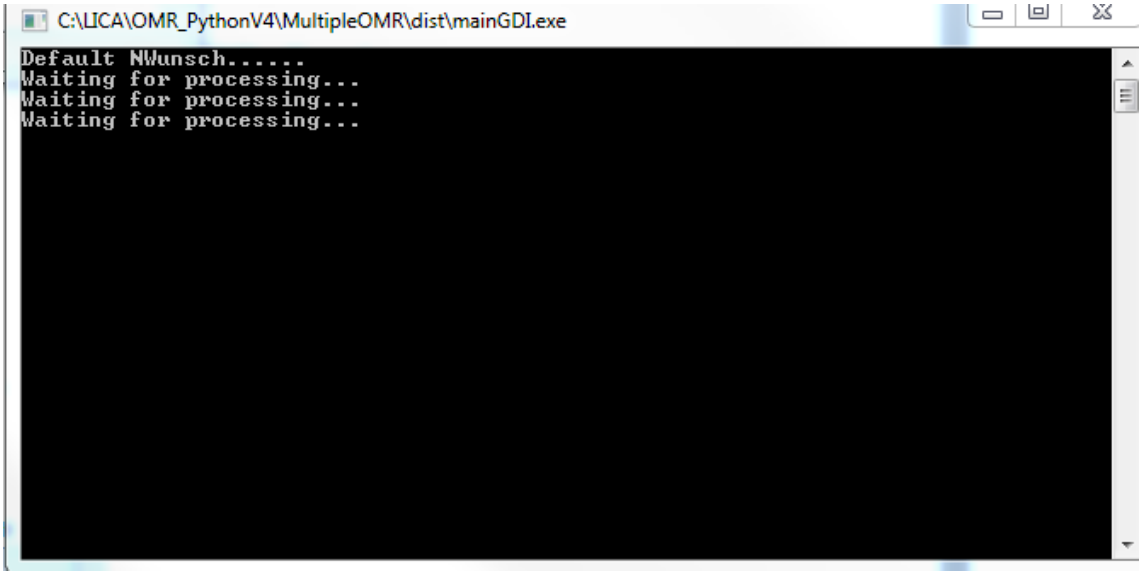


Figure 13: Process Big Data Option. The program is waiting for processing

4.3.-Process Big Data Adapting OMRs (option)

This option is similar to "Process Big Data", but several pre-processing functions are launched. This is especially suitable for piano music:

1.- Removing Gaps. The gaps in voices in music21 produce errors when music is converted to musicXML. For example, the measure in figure 14 is interpreted by music21 correctly, but the xml file is wrong.

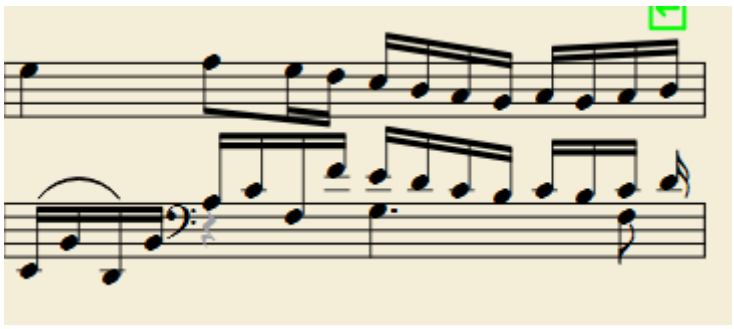
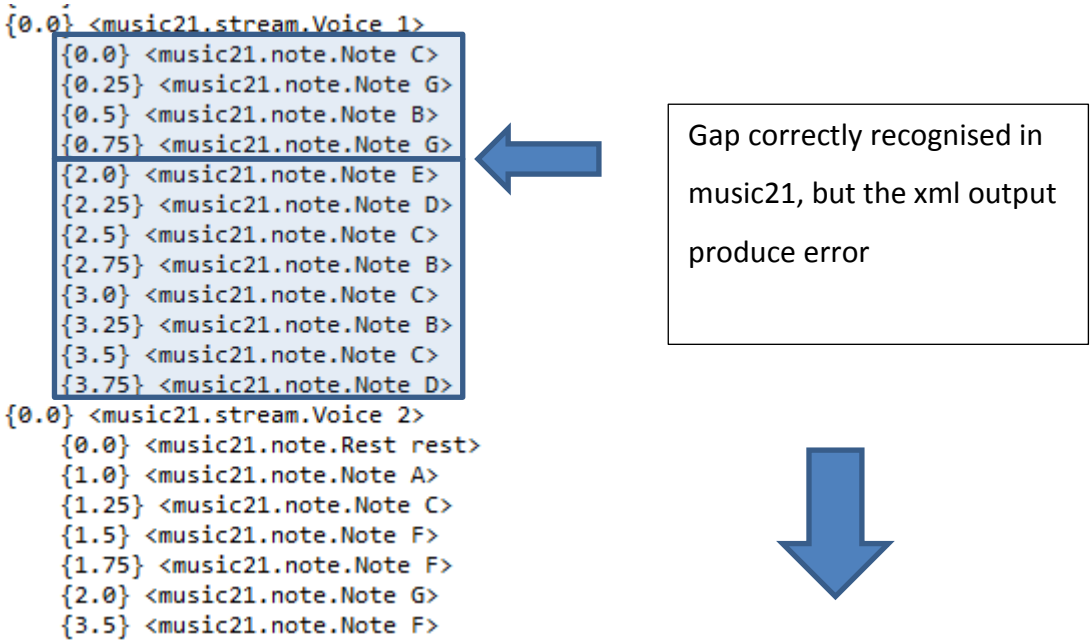


Figure 14: Input to music21 with one gap in one voice (rest in grey)



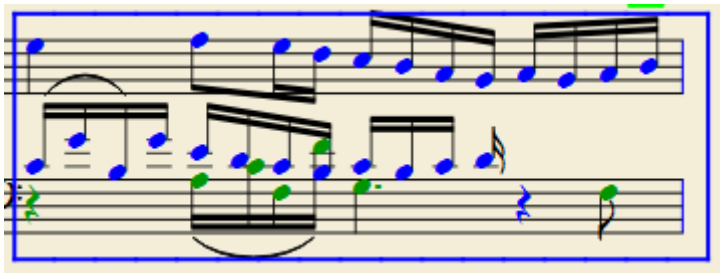


Figure 15: Detail of error in voices

To prevent this error, one process is executed for changing gaps by rests. Doing this, the voices remain correctly aligned.

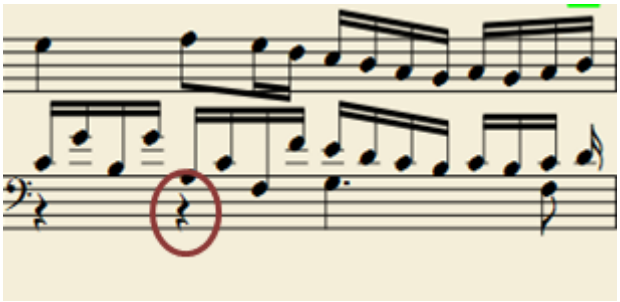


Figure 16: Error corrected inserting rests

2.- Converting voices to chords. In many editions the music is written differently swapping voices by chord. The next figure indicates the same piano measures in two different editions.

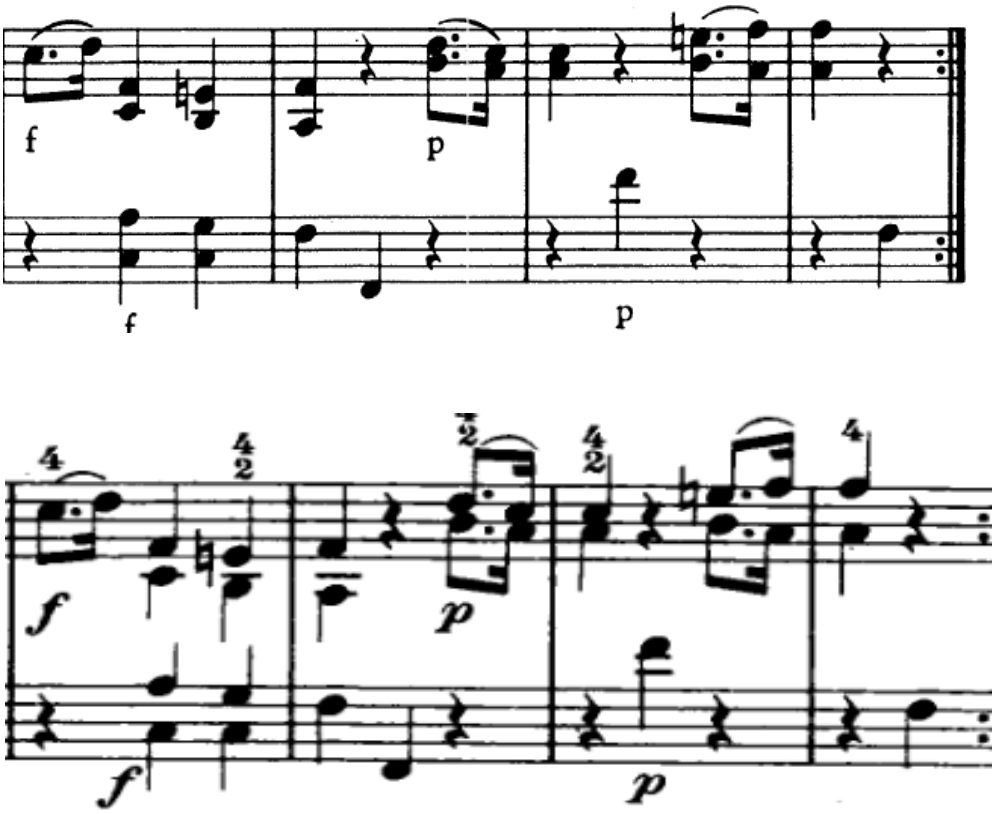


Figure 17: The same measures in two different editions

For comparing OMRs, the system needs equivalent files. The process implemented change the voices by chords based on the offset and rhythm. The figure 18 shows an example.

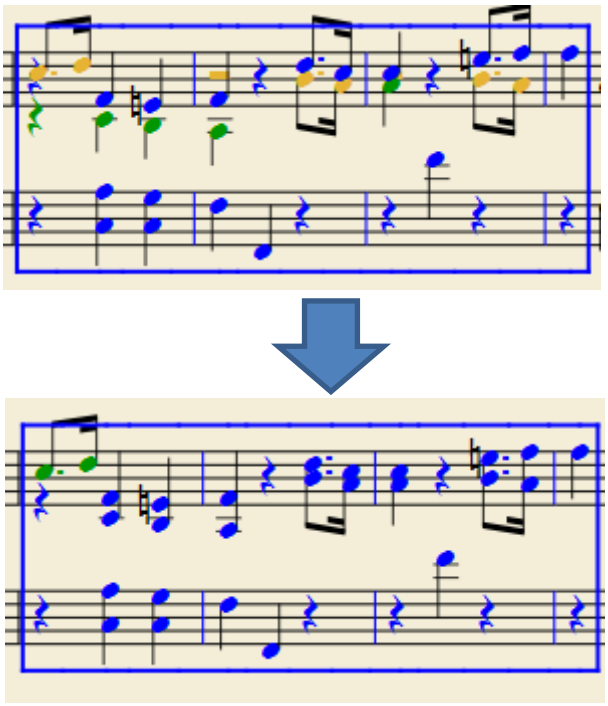
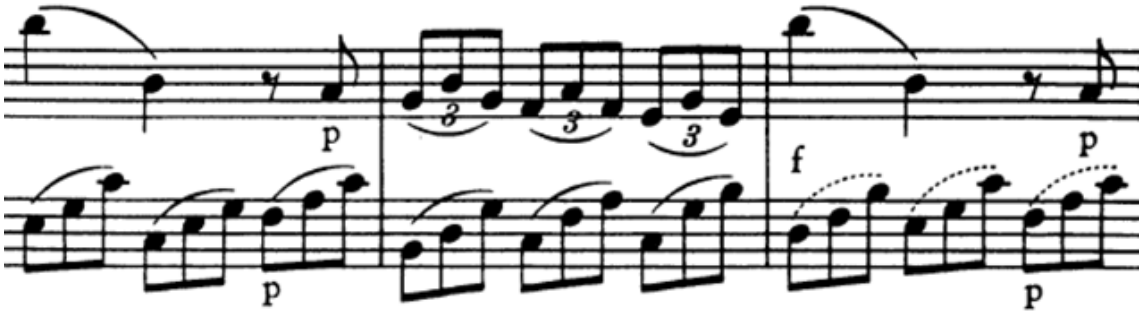
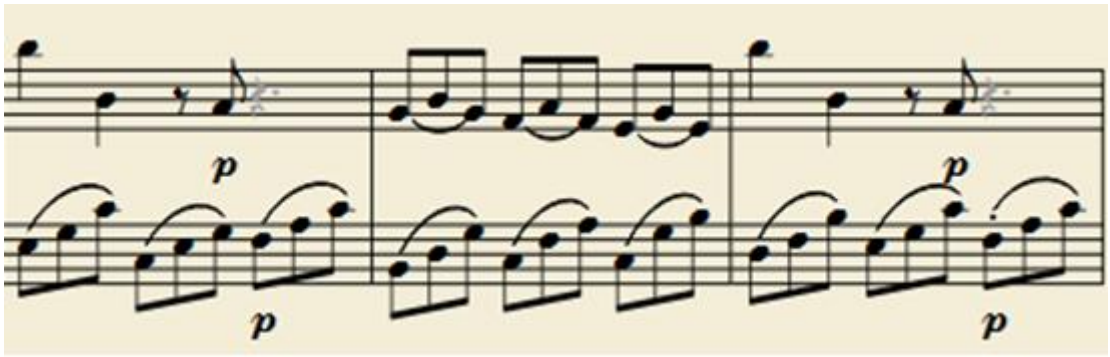


Figure 17: Algorithm to change voices by chords

3.- Triplets not written. In many piano scores, Mozart music for example, the triplets in the accompaniment are very usual, but not always specified. Some OMR correctly recognize the notes, but not the rhythm. The application detects if the length of the measure does not match with the time signature and search triplet patterns based on the beams.



Original score



Score recognised by SmartScore (triplets are not identified)

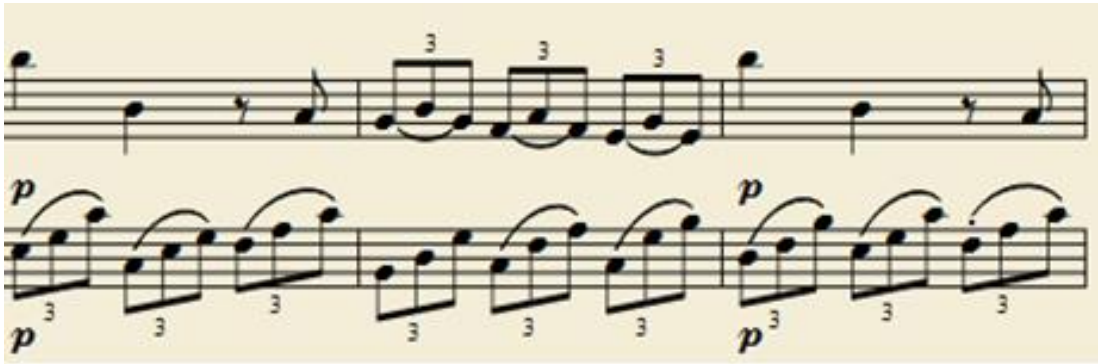


Figure 18: Algorithm for detecting triplets based on beams

4.4.-Complete Process (option)

This option is suitable for running all the processes step by step in a single machine.

The different tasks are:

1.- Runs the sikuli automatism for each omr

2.- Setups the application (see automatism option)

3.- Converts the ground using .krn or .xml files

4.- Runs multi-omr big data

5.- Gets big data results

5.-Results Menu

The options of this menu are related to obtain the results in xls format.

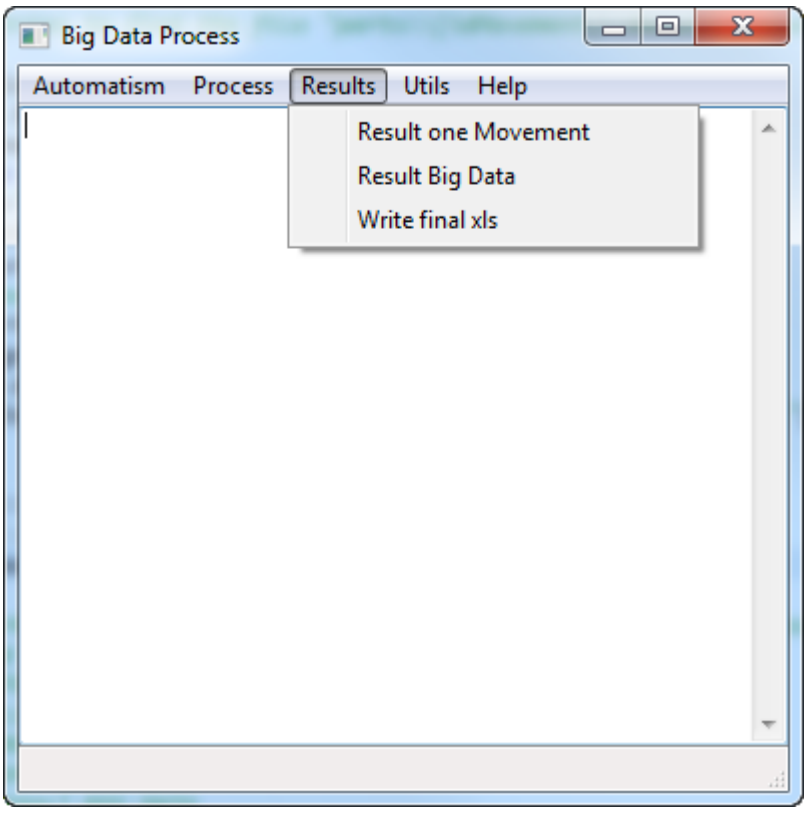


Figure 19: Results Menu

5.1.-Result one Movement (option)

For executing correctly this option, the folder to process should be one movement in the Process folder.

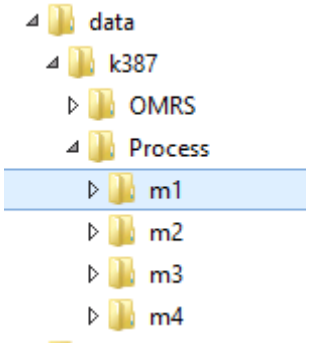


Figure 20

The program creates a folder named *Result* at the same level than XML (Figure 21). The files that this folder contains are:

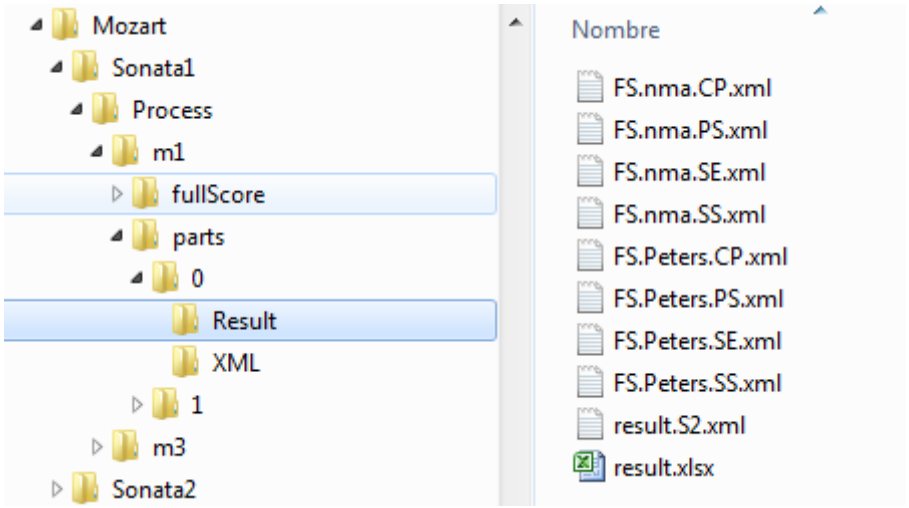


Figure 21

.

- **result.xlsx:** This file contains the errors comparing with the ground for each file. The first row represents the accuracy level based on pitch and rhythm. The second one is the name of the output file. The following rows are the real event (note, chord, rest…) versus the result obtained. According

C:\Users\Victor\Desktop\cach\Sonata1\m1\parts\0\Result\result.S2.xml
94.8692152918
result.S2.xml
[[u'N:A5_0.25', 0.25, '', 0.0, 0.25, '#00ff00'], '*', 103]
[[u'N:G5_0.25', 0.25, '', 0.25, 0.5, '#ff0000'], [u'N:G5_0.5', 0.5, '', 0.0, 0.5, None], 104]
[[u'N:E5_0.25', 0.25, '', 0.75, 1.0, '#00ff00'], '*', 106]
[[u'N:D5_0.25', 0.25, '', 1.25, 1.5, '#ff0000'], [u'N:D5_0.5', 0.5, '', 1.0, 1.5, None], 108]

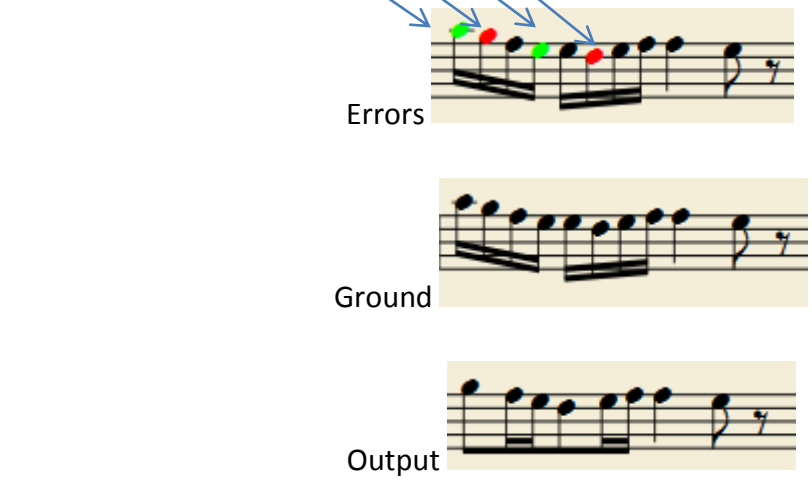


Figure 22

The figure 22 shows an example of errors. Missing notes are colored in green and bad notes (wrong rhythm or pitch) in red.

- **Xml files.** In these files are represented the errors in colors compared to the ground.

At upper level of the different parts, the application writes other two files:

- **resultGeneral.xlsx.** This is a sum up of the different *result.xls* inside each part. In this file is written the percentage of accuracy and in red the OMR that have been taken for aligning and voting.

result.S2.›	FS.Peters.	FS.Peters.	FS.Peters.	FS.Peters.	FS.nma.CI	FS.nma.PS	FS.nma.SE	FS.nma.SS
94,86922	87,2444	87,02595	94,54365	78,43327	77,33072	90,91826	92,66332	65,10264
98,46879	96,63573	95,56075	97,28774	88,66588	82,30501	97,05189	96,59224	92,18935

Figure 23

- **fullScore_errors.xml.** The final score with the different parts aligned. The errors are written in color:
    - Red means bad note
    - Green means missing note

5.2.-Result Big Data (option)

The application runs a loop through the tree folders searching for files to get results. If there is not anything to process, the program waits searching for files every 5 seconds.

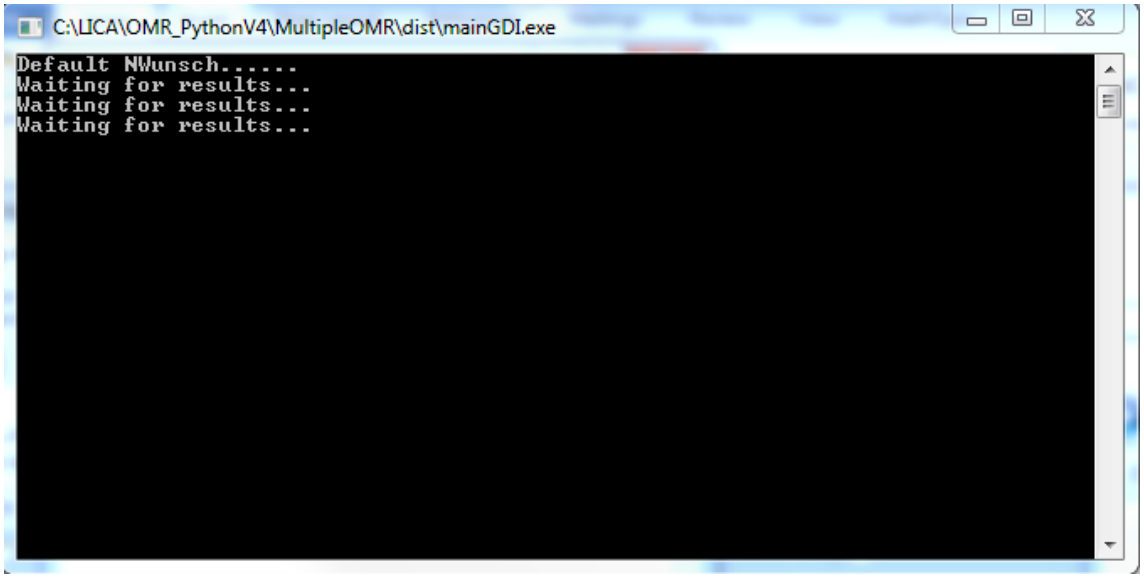


Figure 23. Result Big Data option

5.3.-Write final xls (option)

This option takes all the *resultGeneral.xlsx* files and calculates the average percentage of accuracy for each OMR. The result is written in *final.xlsx* file

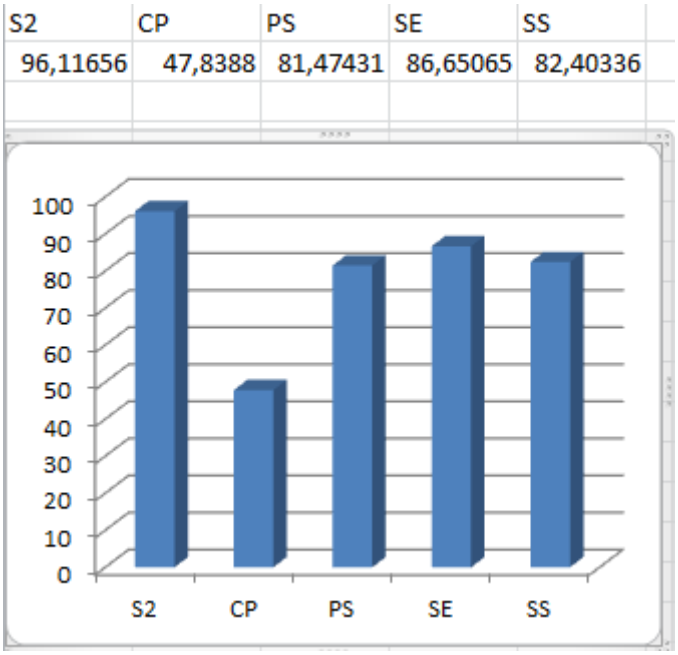
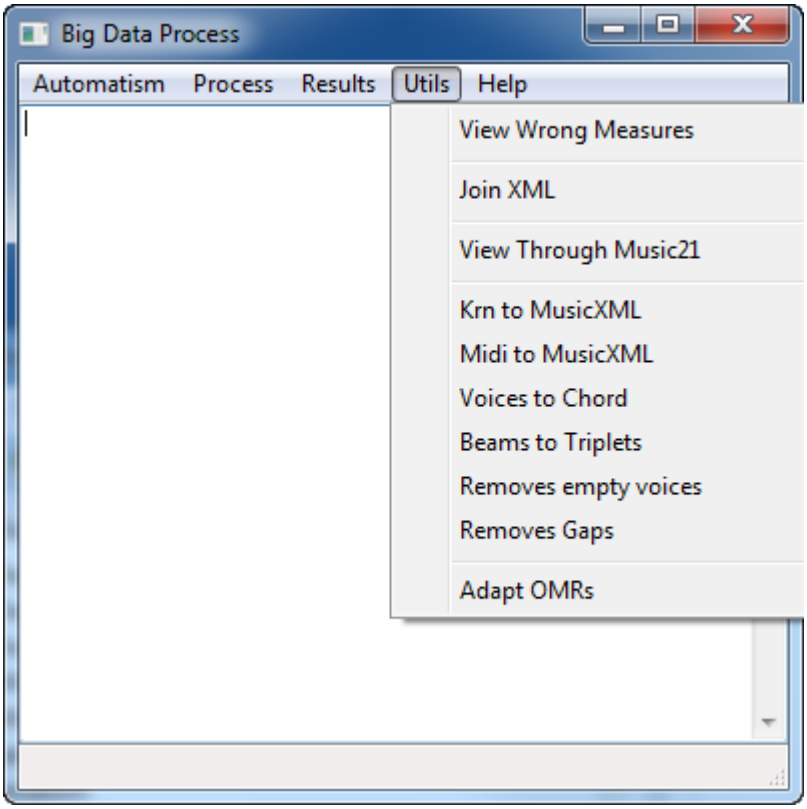


Figure 24. final.xlsx file

6.-Utils Menu

In this menu we have several options that can be used for detecting errors and calculating data. Many of them are integrated in the Process or Result menu, but the direct access to this algorithm are useful in many cases.

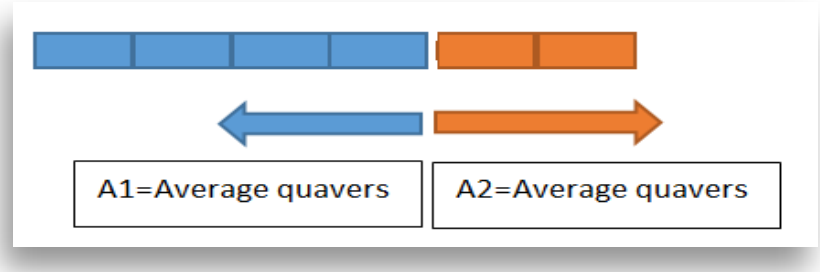


6.1.-View Wrong Measures (option)

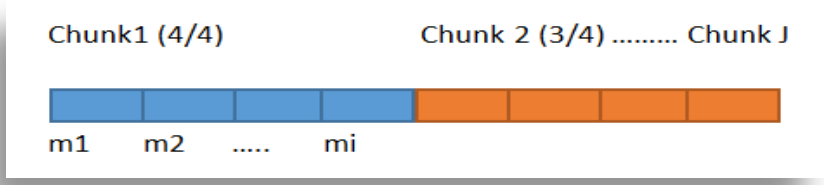
This option implements three different algorithms for calculating wrong measures:

1.- Based on the last time signature. If the length of the measure is different to the last time signature, the measure is flagged as an error.

2.- Based on transitions if time signature is wrong or missing

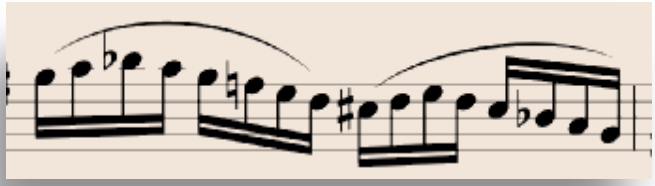


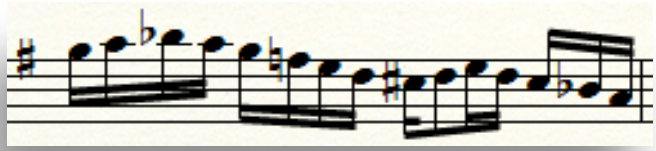
$$if\ abs(A1 - A2) > 1.5\ quavers,$$
$$possible\ transition$$



$$if\ m_i \in chunk_j,$$
$$if\ len(m_i)! = avg(chunk_j),$$
$$wrong\ measure$$

3.- Based on beam errors (odd beaming)





6.2.-Join XML (option)

This option joins different musicxml fragments. It is under development.

6.3.-View Through Music21 (option)

Sometimes, the output produced by music21 is not perfect due to gaps in voices. Other typical errors are key signatures and expressions misplaced. This option shows the file after being read and processed by music21.

6.4.-Krn to MusicXML (option)

It converts .krn files to musicxml using music21

6.5.-Midi to MusicXML (option)

It converts midi files to musicxml using music21

6.6.-Voices to Chords (option)

In many editions the music is written differently swapping voices by chord. For comparing OMRs, the system needs equivalent files. The process implemented in this option change the voices by chords based on the offset and rhythm. More information at 4.3.2

6.7.-Beams to Triplets (option)

In many piano scores, the triplets in the accompaniment are very usual, but not always annotated in the score. Some OMR correctly recognize the notes, but not the rhythm. The application detects if the length of the measure does not match with the time signature and search triplet patterns based on the beams. More information at 4.3.3

6.8.-Removes empty voices (option)

After changing voices to chords, it is possible that one of the voices is empty (just rests). This option detects these voices and removes them.

6.9.-Removes Gaps (option)

This option swaps gaps by rests in voices. Doing this, the voices remain correctly aligned in music21.

6.10.-Adapt OMRs (option)

This option takes all the .xml files in one folder and runs the algorithms 6.6, 6.7, 6.8 and 6.9. The files will be overwriting.