MCMA

Release 1.0

Aug 31, 2021

Contents:

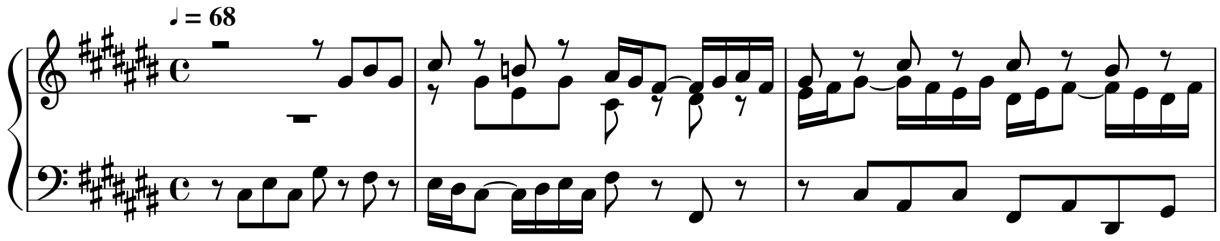
CHAPTER 1

About

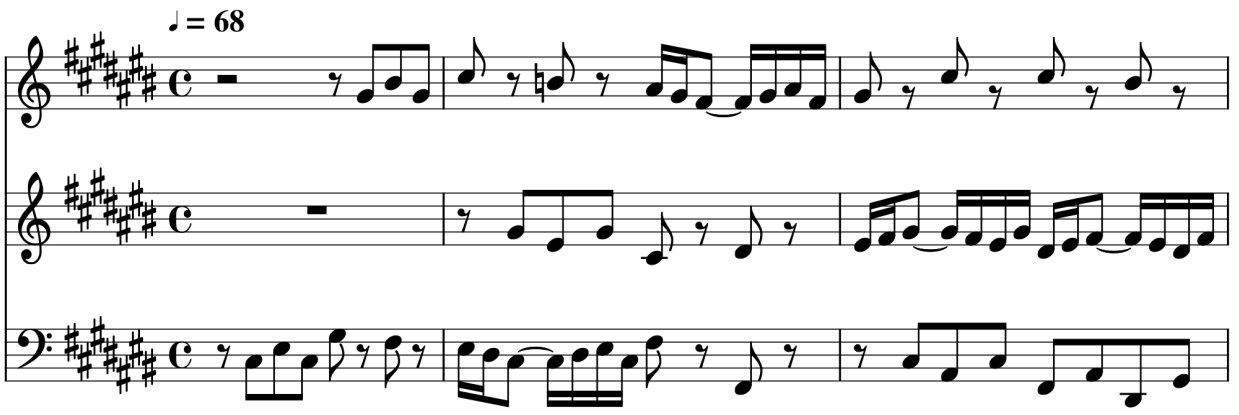
We present *Multitrack Contrapuntal Music Archive* (**MCMA**), a symbolic dataset of pieces specifically collated and edited to comprise, for any given polyphonic work, independent parts.

MCMA is geared towards musicological tasks such as (computational) analysis or education, as it brings to the fore contrapuntal interactions by explicit and independent representation. Empirically analyzing counterpoint music is nearly impossible without voice separation (e.g., by means of "exploding" a part into its constituent voices). Using **MCMA**, each part can be analyzed separately, which offers a unique perspective for investigating voice leading in counterpoint. A musicologist would be able to quickly find examples of retrograde inversions or augmentations in counterpoint melodic lines, or find statistically valid answers for their questions: for example, what is the most common interval to use for parallel contrapuntal motion? Is contrary motion preferred to other types of motion?

Below, an example of multitrack expansion: the first three bars of J.S. Bach's Fugue No.3 in C# Major BWV 872, from the Well-Tempered Clavier, Book II, (top) against the same extract in MCMA's corpus (bottom).

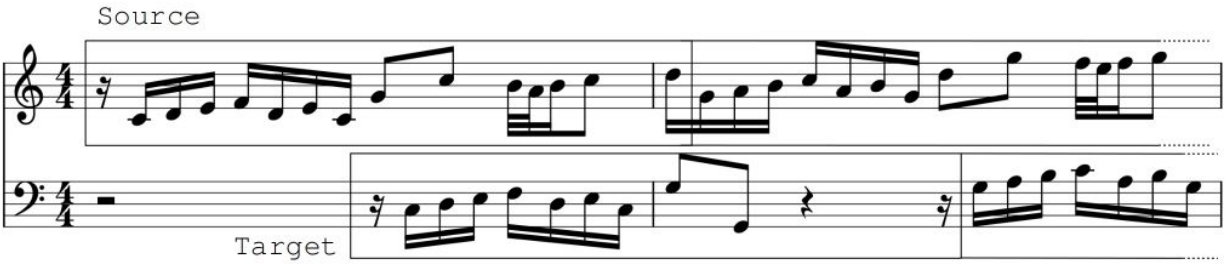


Original



Multitrack

Moreover, **MCMA** can be useful in the context of generative systems and machine learning. For example, the relation between two parts in a contrapuntal piece becomes paramount if one wants to employ natural language processing models, such as attention networks, and treats one part as the musical response to a "query". Below, an example of this approach: the first two bars of J.S.Bach's first two-part inventions.



Query-Answer

1.1 FAIR

MCMA is FAIR-compliant. It has an extensive metadata protocol (F2), which is described in detail in the **Metadata** section of the Guidelines for Contributors. The metadata is registered in a comma separated values file (F4, A1, A1.1): each music work is assigned a globally unique and persistent identifier (F1) and is richly described with a plurality of accurate and relevant attributes (R1).

1.2 Contribution Call

We open **MCMA** to contributions, in the hope that **MCMA** can continue to grow beyond our efforts. Please see the Guidelines for Contributors section.

If you wish to simply use **MCMA** without contributing, we provide instructions on how to download the current dataset as .zip file.

1.3 How to Cite

Aljanaki A., Kalonaris S., Micchi G. & Nichols E. (2021) MCMA: A Symbolic Multitrack Contrapuntal Music Archive. *Empirical Musicology Review*. Forthcoming.

CHAPTER 2

Download MCMA

There are two options to download the dataset.

2.1 Zip File

Download the latest zip file here: mcma-master.zip

2.2 Clone Gitlab Repo

Clone the MCMA gitlab repo. This is recommended for those who want to contribute to the project.

```
git clone https://gitlab.com/skalo/mcma.git
```

CHAPTER 3

Collections in MCMA

The following table gives the number of pieces from each collection that is currently included in MCMA.

CHAPTER 4

Guidelines for Contributors

Please read the following guidelines carefully.

1. Pieces/works/collections MUST be contrapuntal music. Conversely, homophonic pieces are not to be considered.
2. Preference must be given to complete works and/or collections.
3. When choosing works/collections, you are encouraged to use already existing open data as a base for your submissions but check the copyright notice of the files you start with to make sure they are freely shareable.
4. While priority is given to works that are not currently represented in MCMA, we do accept different versions of pieces already collected in the database. These versions should differ either for the reference text or, more importantly, for a different choice in multi-track voice separation.
5. Add a metadata.csv for each work/collection. Please see the **Metadata** section, below, for a detailed description.
6. Add one metadata record for every file.
7. Every file should be saved with the same name as the ID of the metadata.
8. The name of the work/collection folder should directly based on the content of your metadata's column "Collection". For example, if "Collection" is "Sonate da chiesa Op. 4", then your folder will be named "sonate_da_chiesa_op_4". Use the following convention: all lower case, all delimiters and punctuation marks become an underscore, except for the hyphen (hyphen it is).
9. All files must be saved in compressed musicXML format (mxl). Unlike MIDI, mxl supports correct pitch spelling (i.e. G# is not the same as Ab) and that's important information to have.
10. Always reference to an edition that you consider reliable, without edits. This is true for obvious matters such as notes and time signatures but also for more subtle ones.
11. Inside each file, every part should have its separate track. Typically, every part would be monophonic, but occasional polyphony might happen and should be kept. For example, in the specific case of JS Bach's Well Tempered Clavier, if a Fugue is considered to be in 4 voices keep only 4 parts (tracks). Intrapart polyphony can be added as voice expansion or as voice addition, depending on your subjective judgement, but always foregrounding musicological concerns.

12. All ornaments should be written in a symbolic way. It's preferable to omit an ornament than to have it written explicitly.
13. All repeats should be written with repeat bars. No repeat should be expanded.
14. Performance cues, such as rapidly varying metronome marks as used to indicate ritardando, should be removed.
15. At the moment, we do not aim to faithfully represent Figured Bass notations, dynamics, slurs, or tempos. But certainly you shouldn't throw this information away if you have it already, as our scope might expand in the future.

4.1 Metadata

The fields of **MCMA**'s metadata protocol are the following:

- **ID**: this should univocally describe the work, using some of the subsequent keys, in the following format: *Last Name-First Name (capital initials only)-Catalog Number* and a selection/combination of compositional style (if explicitly stated in the *Title*, but abbreviated if possible, *e.g.*, P for Prelude) and/or *Movement Number* that best describes the work. For example, BachJS-BWV870-1. Since the ID to every record of MCMA should be unique, the string "-An" should be appended to the ID of alternative versions of a piece already represented in the dataset. Here, *n* is an integer taking the first available value from 1 upwards, example "-A2" if the piece is already represented in MCMA by a reference and a first alternative version.
- **Last Name**: the composer's surname.
- **First Name**: the composer's forename and middle name(s).
- **Title**: the full work's name, *e.g.*, *"Allemande from Cello Suite No. 1 in G Major, BWV 1007"*.
- **Collection**: the name of the collection/work the piece belongs to, if it exists. For example, *"The Well-Tempered Clavier"*.
- **Catalog Number**: scholarly catalog number, *e.g.*, BWV 1007. In case of conflicting naming, please add a clarification in the **Notes** entry.
- **Movement Number**: an integer. In case of a theme/aria in a theme & variation_, or for the prologue in an *opera*, use 0.
- **Number of Tracks**: an integer. This will correspond to the number of tracks in the mxl file.
- **Instruments**: series of instruments names (top to bottom) using a semicolon as delimiter (*e.g.*, violin;viola;cello). If all parts are the same instrument, one instance of it is sufficient.
- **Year**: the year the work was composed, if known. Note: if only the parent Collection's dates are known, and if they spanned several years, this will be the year of completion.
- **Provenance**: name or URL of the dataset/database containing the original file used as basis for the contrapuntal multitrack expansion.
- **Reference Edition**: self-explanatory, *e.g.*, *Universal Edition*.
- **Link to Reference Edition**: the URL of an accessible version of the reference edition.
- **Comments**: anything else.

4.2 MCMA Utilities

4.2.1 Metadata Creation

The python utility `metadata_writer` can be used to create the metadata file. To do so, update the `basic_info` inside `metadata_writer.py` with the required information, and follow the instructions at the top of that file to generate a `metadata.csv` file. This requires basic knowledge of python and running python scripts.

4.2.2 File Cleanup/Normalization Utility

Once a set of files is prepared in a directory along with its corresponding `metadata.csv` file, another script, `normalize_instruments.py`, should be run to convert the MusicXML files into a standard format used by the MCMA corpus. This updates the MIDI sounds used in the files, adjusts volume and panning, and cleans up the title/subtitle/composer/etc displayed on the first page of each piece.

To run this utility:

- Open a terminal window.
- Change to the `mcma/utilities` directory; *e.g.*, `cd mcma/utilities`
- `python3 normalize_instruments.py`

Note that python version 3.6 or higher is required.

After running, the resulting new versions of the files are written to subdirectories titled `normalized`. It is recommended to verify the output of the utility and then to submit these cleaned up versions to MCMA via a merge request.

N.B. This utility depends on accurate metadata in the `metadata.csv` file. In particular, for each part in each track, the names of the instruments must be listed:

- If all instruments are the same for a piece, just write the instrument name in the Instrument column in the metadata; *e.g.*, `Harpsichord`
- If each track has different instruments, list the instruments, separated by `;` marks. *E.g.*, `Violin;Gamba;Harpsichord`

If an instrument is not known by the script, it will be replaced by the instrument with the closest name. If this results in the wrong instrument chosen, the file `utilities/midi_instruments.csv` can be updated to add the new instrument name to an existing general MIDI sound. For instance, `Gamba` has been added to the MIDI `cello` sound in this file so that the sound is approximated in MIDI playback.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search