

# CHAPTER 4

## Digital Representations

---

This section deals with Research Question (RQ1): Which digital representations are suitable encodings of the musical notations appearing in *Baishidaoren Geku*?

As explained in Section 2.1, the digital representations of musical notations must accurately reflect the encoded features of the analog notations. This ensures that the stored information faithfully represents the original notations, while also minimizing cultural biases that may arise when using formats specifically designed for staff notation or common practice period music.

It does so through the introduction of a digital representation for the three kinds of musical notations found in *Baishidaoren Geku*. This constitutes the foundation for symbolic digitization. We start with the *suzipu* representation, and then generalize it to the absolute pitch notation *lülüpu* and the *jianzipu* tablature for the instrument *guguin*.

Even though the focus of this dissertation lies on *suzipu* notation, introducing these two additional representations allows for the inclusion of all of Jiang Kui's 109 pieces in our KuiSCIMA corpus in Chapter 5, both with and without musical notations.

### 4.1 Digital Representation of *Suzipu* Music Pieces

As opposed to machine-readable corpora of music with specific note durations (e.g., those listed in the related works Chapter 2), which use well-established digital encodings such as MusicXML or MEI (Music Encoding Initiative), these encodings are not well-suited for the flexible and disputed system of secondary symbols in *suzipu* notations. In order to create a machine-readable representation as close as possible to the *suzipu* notation system, a novel system of symbolic representation is created. The system used in this corpus is based on a collection of attributes, which is eventually saved as JSON format.

In the root folder of the project's repository, the file `json_schema_suzipu.json`<sup>1</sup> is a schema giving a more technical documentation of the digital representation format used in this dissertation, and it can also be used to verify that a *suzipu* corpus file is of valid structure. Nevertheless, in Figure 4.1 a graphical overview is presented, and in the next section, a verbal elaboration is given.

---

<sup>1</sup>[https://github.com/SuziAI/gui-tools/blob/v2.1/json\\_schema\\_suzipu.json](https://github.com/SuziAI/gui-tools/blob/v2.1/json_schema_suzipu.json)

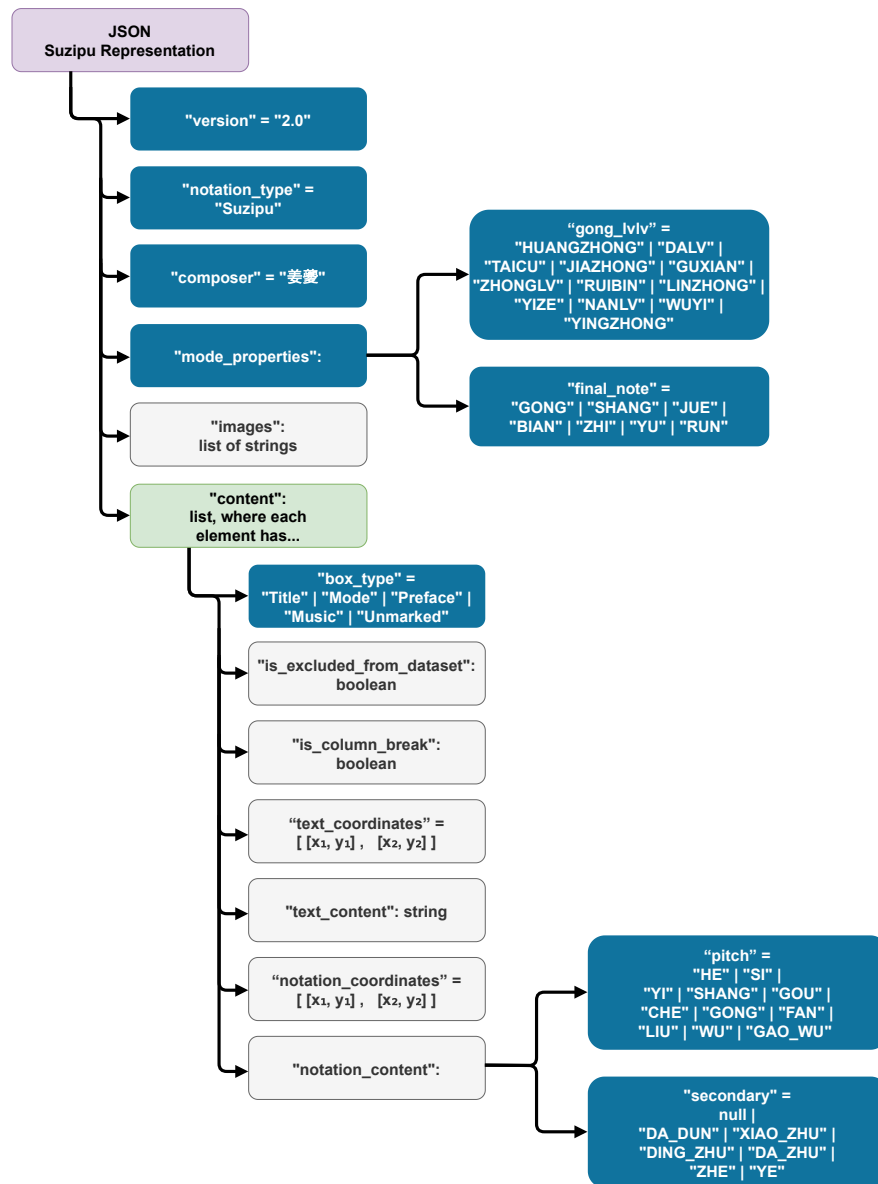


Figure 4.1: Graphical overview of the JSON representation for *suzipu* notation. Blue boxes mark required fields, while grey boxes mark optional fields.

## 4.2 Metadata Properties

For the metadata, in the root level, the following piece metadata properties are stored:

1. `version`: This indicates the version of the format used for storing the piece, to allow for future modifications of the format while maintaining support for previously released versions. The documentation in this doctoral thesis is for version 2.0.
2. `notation_type`: This field is used to store the type of Chinese notation used in the file. Since here the representation for *suzipu* is discussed, the content of this field must be "Suzipu".
3. `composer`: Here, the composer's name is stored.
4. `mode_properties`: This object contains the properties uniquely defining the mode, i.e., the *lǜ* which is corresponding to *Gong* (stored in `gong_lvlv` with the representation from Table 4.1, and the mode's final degree (stored in `final_note` with the representation from 4.2). Note that this is independent of the segmentation boxes annotated with the mode, since some pieces in *Baishidaoren Gequ* do not contain explicit mode information, and in some instances the mode name used is an alternative name. To provide flexibility with respect to choosing any of the possible 84 modes, the mode is represented by these two properties.
5. `images` (optional): This field contains an ordered list of relative paths to the image files belonging to the piece.
6. `content`: This is an ordered list of all the individual boxes in the piece, i.e., title characters, mode information characters, preface characters, and musical boxes consisting of notational units with lyrics. The order inside this list corresponds to the order of the boxes in the piece. Each of the items contained must have a certain set of properties.

## 4.3 Content Properties

Concerning the `content`, each box may contain textual or notational information, including the possibility to store segmentation box coordinates. The fields are as follows:

1. `box_type`: This contains the type label of the box according to Table 4.4. E.g., a segmentation box containing a musical character has the type `MUSIC`.
2. `is_excluded_from_dataset` (optional): Since one of the purposes of this corpus is to provide the basis for OMR algorithms being able to detect the *suzipu* notation given an image of it, the baseline dataset must be free of instances which are detrimental to model performance. This might be the case if the image data is heavily distorted (e.g., by ink splatters) or the annotation and the image data do not coincide due to a misprint. In case the box is excluded from the image dataset, the value of this field is `true`.

<i>Lülü</i>	Romanization	ASCII Representation
黄钟	<i>Huangzhong</i>	HUANGZHONG
大吕	<i>Dalü</i>	DALV
太簇	<i>Taicu</i>	TAICU
夹钟	<i>Jiazhong</i>	JIAZHONG
姑洗	<i>Guxian</i>	GUXIAN
仲吕	<i>Zhonglü</i>	ZHONGLV
蕤宾	<i>Ruibin</i>	RUIBIN
林钟	<i>Linzhong</i>	LINZHONG
夷则	<i>Yize</i>	YIZE
南吕	<i>Nanlü</i>	NANLV
无射	<i>Wuyi</i>	WUYI
应钟	<i>Yingzhong</i>	YINGZHONG
黄钟清	<i>Huangzhong Qing</i>	HUANGZHONG_QING
大吕清	<i>Dalü Qing</i>	DALV_QING
太簇清	<i>Taicu Qing</i>	TAICU_QING
夹钟清	<i>Jiazhong Qing</i>	JIAZHONG_QING
折字	<i>Zhezi</i>	ZHE_ZI

Table 4.1: The ASCII character representation for the *lülüpu* notation

3. `is_line_break` (optional): In order to reflect the given edition's typographical layout as closely as possible, line breaks (respectively column breaks in traditional reading order) are reflected in the JSON representation as well. This field is `true` if directly after the segmentation box a line break occurs.
4. `text_coordinates` (optional): This contains the coordinates of the upper left and lower right boxes belonging to textual data, describing the segmentation box boundary.
5. `text_content` (optional): The textual annotation of the box is stored in this string. For boxes of type `Music`, the lyrics character is stored in this field.
6. `notation_coordinates` (optional): Similar to `text_coordinates`, the coordinates for the contents pertaining to musical notations of the box are stored here.
7. `notation_content` (optional): This field consists of two subfields `pitch` for the pitch and `secondary` for the secondary *suzipu* symbol. The correspondences between the notation and the field representation are given in Table 4.3.

Scale Degree	Romanization	ASCII Representation
宫	<i>Gong</i>	GONG
商	<i>Shang</i>	SHANG
角	<i>Jue</i>	JUE
變	<i>Bian</i>	BIAN
徵	<i>Zhi</i>	ZHI
羽	<i>Yu</i>	YU
闰	<i>Run</i>	RUN

Table 4.2: The ASCII character representation for the relative scale degrees.

<i>Suzipu</i> (Pitch)	Name	ASCII Representation
ム	合	"HE"
マ	四	"SI"
一	一	"YI"
么	上	"SHANG"
ㄥ	勾	"GOU"
ハ	尺	"CHE"
フ	工	"GONG"
リ	凡	"FAN"
ス	六	"LIU"
ウ	五	"WU"
ㄨ	高五	"GAO_WU"
<i>Suzipu</i> (Secondary)	Name	ASCII Representation
ㄨ	大顿	"DA_DUN"
リ	小住	"XIAO_ZHU"
フ	丁住	"DING_ZHU"
カ	大住	"DA_ZHU"
ㄣ	折	"ZHE"
ㄩ	拽	"YE"

Table 4.3: The ASCII character representation of each of the 11 pitch symbols and 6 secondary symbols of *suzipu* notation. The representation is the capitalized pinyin transcription of the symbol's name without tone marks.

Type label	Description
Title	Marks the title characters of the piece.
Mode	Marks the mode characters of the piece.
Preface	Marks the characters belonging to the preface of the piece.
Music	Marks the piece’s musical notation symbols, consisting components for both musical notation and lyrics information.
Unmarked	Whenever a new box is created (either manually via the <code>Create</code> button or the segmentation algorithm), its label is set to be unmarked by default. The final representation should not contain any unmarked box.

Table 4.4: The type labels used for marking the type of each box. Except for the `Unmarked` boxes, the types correspond to the parts listed in Section 3.4, except that `Music` contains information about both the musical notation and the lyrics.

## 4.4 Representation of Other Notations

By generalizing some properties, e.g., the `mode_properties` and `content` fields, we can use similar formats for other kinds of notation, such as *lülüpu* or *jianzipu*. We do so by deriving separate schemata for each musical notation from a general JSON schema.<sup>2</sup>

### 4.4.1 Representation of *Lülüpu* Pieces

The *lülü* are not only important as the basis for the absolute pitch system (introduced in Section 3.1.1), but they are also used to notate pitches directly. This notation is referred to as *lülüpu*. It is especially common in the contexts of ritual music such as the sacrificial *jisiyue* 祭祀樂 (Chiu, 2009, p. 275). In addition, the *lülüpu* notation shares roots with other East Asian musical notations, such as the *jeongganbo* notation used in Korean court music (D. Han et al., 2024).

In *lülüpu*, the absolute pitches are written down directly, so it is not strictly necessary to include information about the mode. Therefore, the field `mode_properties` is optional, as opposed to *suzipu*, where the mode information is crucial for determining the absolute pitch of the pitch component. In addition, Jiang Kui used an additional *zhezi* 折字 symbol indicating intonation instructions for some instruments, as translated by Thompson (n.d.-c).

Regarding the ordered list `content`, each item has the same properties as the *suzipu* representation. Of course, the property `notation_content` is different. The optional field contains the property `pitch`. Here, the ASCII string for one of the 16 extended *lülü* or *zhezi* as described in Table 4.1 is contained. The implementation details are found online.<sup>3</sup>

<sup>2</sup>[https://github.com/SuziAI/gui-tools/blob/v2.1/json\\_schema.json](https://github.com/SuziAI/gui-tools/blob/v2.1/json_schema.json)

<sup>3</sup>[https://github.com/SuziAI/gui-tools/blob/v2.1/json\\_schema\\_lvlpvu.json](https://github.com/SuziAI/gui-tools/blob/v2.1/json_schema_lvlpvu.json)

### 4.4.2 Representation of *Jianzipu* Pieces

*Jianzipu* (literal meaning: reduced character notation) is a tablature for the instrument *guqin*, in which special characters convey the instruction of how to play the instrument. In fact, *jianzipu* developed from the older *wenzipu* 文字譜 tablature, featuring playing instructions in full sentences. The earliest *guqin* piece still preserved today (*Youlan* 幽蘭 from a scroll of the 7<sup>th</sup> century) uses this kind of notation. Jiang Kui's *Guyuan* from the 12<sup>th</sup> century is the second earliest *guqin* piece with notation currently known, and the very first that is preserved using *jianzipu* notation. The kind of *jianzipu* used in *Baishidaoren Gequ* is already very close to the one still in use today (Y. Yang, 2014, pp. 31–32, 48).

The digital representation of *jianzipu* pieces is challenging in many regards. On the one hand, the number of individual playing techniques that are represented with a special component is very high, so that at least 150-200 of such components appear in handbooks explaining the notation (van Gulik, 1940, p. 127). In addition, the distribution of these is highly imbalanced, ranging from basic techniques such as *tiao* 挑 that occur frequently to special techniques that are used seldom.

On the other hand, the structure of individual *jianzipu* symbols is complex. As opposed to the *suzipu* and *lülipu* notations, which consist of two respectively one components, a single *jianzipu* symbol can be of multiple classes, where each class has a certain structure. Some of these classes can again be made up of multiple components. This information is stored in the field `notation_content`, and accordingly, for this notation it is a nested object, where the highest level consists of the two properties `type` and `content`.

For this digital representation, a distinction into three classes is made:

- String number annotations. Here, the property `type` is "STRING\_NUMBER". The property `content` is a string containing either "1", "2", "3", "4", "5", "6", or "7", which refers to one of the seven strings of the *guqin*.
- Left hand annotations. Here, `type` is set to "LEFT\_HAND", and `content` contains an ordered list of the left hand symbols used in the notation symbol.
- Full *jianzipu* annotations. This class has `type` set to "FULL\_JIANZIPU", and the `content` field represents the tree structure of the *jianzipu* symbol. Each node has the string attribute `content`, and if it is an internal node, it also has the attribute `children` which is an ordered list consisting of all of the child nodes. For leaf nodes, the `content` is the annotation of the *jianzipu* element, while for internal nodes, it is a composition descriptor as listed in Table 4.5, which determines the number and the order of the child nodes.

An example of a full *jianzipu* character with its annotation can be seen in figure 4.2. This *jianzipu* notation can be read as *wumingzhi shihui da wuxian* 無名指十徽打五弦, which indicates that the left hand's ring finger is stopping the string at the 10<sup>th</sup> position, while the right hand's ring finger plucks the fifth string in direction of the player.








Descriptor	Meaning	Order of Child Nodes
	Left-right composition	[LEFT, RIGHT]
	Top-bottom composition	[TOP, BOTTOM]
	Left-middle-right composition	[LEFT, MIDDLE, RIGHT]
	Top-middle-bottom composition	[TOP, MIDDLE, BOTTOM]
	Surround-from-upper-left composition	[OUTER, INNER]
	Surround-from-lower-left composition	[OUTER, INNER]
	Surround-from-upper-right composition	[OUTER, INNER]

Table 4.5: A table of the composition descriptors used in the full *jianzipu* internal nodes.

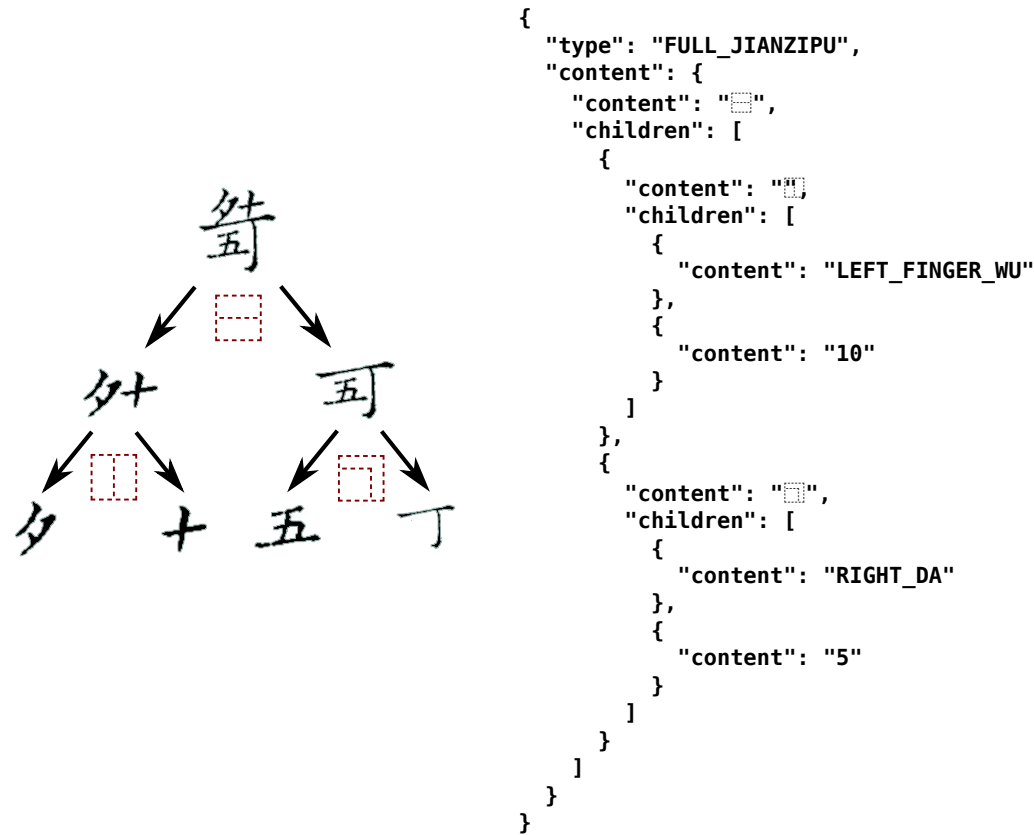


Figure 4.2: Left: A full *jianzipu* character with its tree decomposition. Right: Its digital representation.



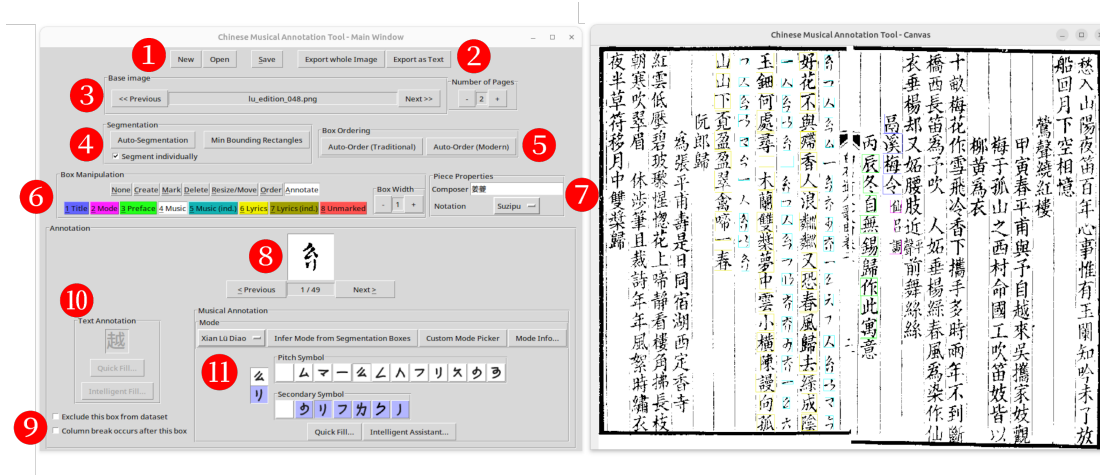


Figure 4.3: The Main Window and Canvas of the annotation tool.

Since the number of *jianzipu* basic components is difficult to determine,<sup>4</sup> custom components can be added by providing an entry in a special file and including the corresponding image.<sup>5</sup>

The implementation details are online.<sup>6</sup>

## 4.5 Digital Toolbox

In order to facilitate the digitization of historical Chinese musical notations, two tools are created: The *Chinese Musical Notation Annotation Tool* for the annotation of photographic reproductions of Chinese music, and the *Chinese Musical Notation Editor* for the creation of purely symbolic representations of music.

During the creation of the KuiSCIMA dataset, both tools were repeatedly adapted and refined to make digitization as convenient as possible. In this section, a description of the two tools is given.

### 4.5.1 The *Chinese Musical Notation Annotation Tool*

Supplementary to the description in this section is the step-by-step tutorial in Appendix C.

Figure 4.3 shows the Main Window and the Canvas. In the first window, the widgets for selecting actions are placed, while in the canvas window, the selected pages can be annotated. The widgets in the Main Window are marked with red numbers and described below:

<sup>4</sup>E.g., even when restricting ourselves to the simplest *anyin* 按音 plucks, i.e., a single right hand finger plucks a string while it is stopped with a finger of the left hand at an integer position, we get  $8 \cdot 7 \cdot 4 \cdot 14 = 3136$  different combinations.

<sup>5</sup>[https://github.com/SuziAI/gui-tools/blob/v2.1/src/plugins/jianzipu/gui\\_config.json](https://github.com/SuziAI/gui-tools/blob/v2.1/src/plugins/jianzipu/gui_config.json)

<sup>6</sup>[https://github.com/SuziAI/gui-tools/blob/v2.1/json\\_schema\\_jianzipu.json](https://github.com/SuziAI/gui-tools/blob/v2.1/json_schema_jianzipu.json)

Pitch ID	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	c	c#	d	d#
Lülu	黄	大	太	夹	姑	仲	蕤	林	夷	南	无	应	黄清	太清	太清	夹清
Gongche	合	下四	四	下一	一	上	勾	尺	下工	工	下凡	凡	六	下五	五	高五
Scale Degree	角		变	徵		羽		闰	宫		商		角		变	徵
Suzipu	厶		マ	一		么	厶	人	フ		リ		久		ウ	ヲ

(Final is marked in cyan)

Figure 4.4: The intelligent Mode Information widget is visualizing the properties of a mode and the implied relationship between notations and pitches.

1. This is the widget for creating a new file, opening an existing file or saving a file to the hard disk. The button names correspond to the performed actions. When creating a new file, the user is asked to choose the directory containing the images to be annotated. For opening and saving files, the JSON encoding described in Section 4.1 is used.
2. The export widgets contain functions for exporting the canvas image (button `Export whole Image`) or to a human-readable textual representation of the annotated contents (button `Export as Text`).
3. Using the buttons `<<Previous` and `Next>>`, the image selection widgets allows the user to navigate to the page that marks the beginning of a piece. In case a piece contains multiple pages, the number of pages can be increased with the `+` and `-` buttons, resulting in the next image (with respect to alphanumeric ordering) of the image directory to appear in the canvas.
4. The segmentation widget lets the user access the Chinese character segmentation algorithm for the automatical creation of the bounding boxes using the button named `Auto-Segmentation`. The status of the checkbox `Segment individually` indicates whether the individual images of the workspace are segmented each separately or treated as a large image. With the button `Min Bounding Rectangles`, the bounding box is shrunk so that excess whitespace is removed while keeping all non-white pixels inside.
5. The box ordering widgets provide function for automatically ordering the bounding boxes according to the reading order that is used. For Chinese texts that are read column-wise from right to left, the button `Auto-Order (Traditional)` is used, while for modern reading order, i.e., row-wise from top to bottom, the `Auto-Order (Modern)` button is used.
6. The box manipulation widgets are for the manual manipulation of bounding boxes and their contents. The top row determines the actions, while the bottom row contains

the available box types that are tied to the content classes in the piece. The button `Create` allows the user to draw a segmentation box of the selected content class in the Canvas window by pressing down the right mouse key and releasing it, these two points are then registered as a new bounding box. The `Mark` button lets the annotator assign a bounding box to another content class by right clicking inside it; if the `Ctrl` key is held, also all boxes that follow in the reading order are marked, and if `Alt` is pressed, all following boxes until a column break are marked. The `Delete` button works analogously, but deletes the selected boxes, while with the `Resize/Move` action the box can be moved by making a right click in the center and placing it elsewhere, or be resized by a right click on a border of the box. With `Order`, a manual ordering of the boxes can be made in case there are deviations from the automatically determined reading order; here, the boxes that should be reordered are selected and with pressing the `Enter` key, the selection is ordered in the sequence of selection. After the bounding boxes are all assigned, the `Annotate` button allows the use of the widgets (8)-(11) for annotation of the bounding boxes.

7. In the piece properties widget, the composer name and the notation type can be selected. Depending on the notation type, the musical annotation widget (10) adapts accordingly.
8. This widget is the selection and display of the currently modified bounding box, and with the `<Previous` and `Next>` buttons, the available boxes of the currently selected content class can be traversed in the assigned order. In the middle, the image content of the bounding box is displayed, and below its current index and total number of boxes in the selected content class.
9. In this widget, bounding boxes can be excluded from being exported into the OMR dataset. This is necessary in case of writing errors that cannot be assigned a valid annotation. Also, the column breaks can manually be adjusted in case they do not agree with the automatically assigned reading order.
10. The text annotation widget is available for all text based content classes, i.e., `Title`, `Mode`, `Preface`, `Lyrics`, `Lyrics (ind.)`, and `Unmarked`. Using the `Quick Fill...` button, all boxes of the currently select content type can be annotated at the same time by providing a string, while `Intelligent Fill...` uses a *Tesseract* OCR system for predicting and automatically filling in the Chinese characters (Kay, 2007).
11. The musical annotation widget is chosen depending on the selected notation type in widget (7). Here, the *suzipu* annotation widget is shown, where in the top row, the piece's mode can be selected either by choosing its name in the first drop-down list, by trying to infer the mode given the contents of the boxes marked with the `Mode` content type, or by custom selection of the mode's properties. The button `Mode Info...` opens a window which contains intelligent and interactive display of the mode's relationship to pitch, *lülü* notation, Chinese *gongchepu* notation, the scale degrees inside the mode, and their relationship to the *suzipu* notation symbols. The widget can be seen in Figure 4.4, and its intelligent functionality is explained by the

conversion between different notation sets in Section 7.1. The *suzipu* notation consists of two components, the pitch and secondary components, that are selected individually. Using `Quick Fill...`, all music boxes of the currently selected content class can be filled by providing a JSON string. The button `Intelligent Assistant...` starts the `Suzipu Intelligent Assistant`. A description of this widget is found below.

In addition, for each of the notation types, an intelligent assistant is provided, which can display a selection of samples from the KuiSCIMA dataset (see Chapter 5) with the same annotation as the currently selected sample. For the *lülüpu* and *suzipu* notations, there is also a built-in OMR functionality for computer-aided annotation processes, whose algorithms are thoroughly explained in Chapter 6. In this section, the intelligent assistant for the *suzipu* notation is presented in detail.

The `Suzipu Intelligent Assistant` is an intelligent widget that provides automatic labeling of notation instances, similarity analysis and display of all KuiSCIMA instances with the same annotation. It is shown in Figure 4.5, and its components are as follows:

1. By clicking the `Predict` button, all music boxes of the currently selected content class are automatically predicted using an OMR algorithm. The algorithms used are described in Section 6.3 (for *suzipu*) and Section 6.4 (for *lülüpu*). Its results are stored in the widget and can be visualized in widgets (3) and (4). The annotations that are in the `Main Window` are not changed in this step.
2. Using the `Overwrite Annotations with Predictions` button, the annotations inside the `Main Window` are replaced by the model predictions. This step is deliberately introduced for users that want to check single annotations but prefer to annotate by themselves.
3. After prediction, the three most probable model suggestions are visualized in descending order; separately for the pitch and secondary components. Below the suggestion, the model's confidence score is shown. These scores are calibrated, which means that if the confidence level is, e.g., 95%, the user can expect that the model's annotation is correct in around 19 out of 20 cases. The calibration method is described in Section 6.5.
4. The similarity widget displays the three most similar notation instances in the KuiSCIMA dataset with respect to the optical properties of the images, for both the pitch and secondary features. Below, their annotation, the edition in which the instance occurs, and a similarity score is displayed. The algorithms employed here are described in Section 6.6.
5. In this widget, a wide variety of notation instances from the KuiSCIMA dataset with the same annotation are displayed. This widget is inspired by the glyph listing by Tabin et al. (2023), and the algorithms used are found in Section 6.1.

The *Notation Display* window is shown in Figure 4.6, and a more detailed description of its inner workings is found in Section 4.5.3. Its widgets are explained below:

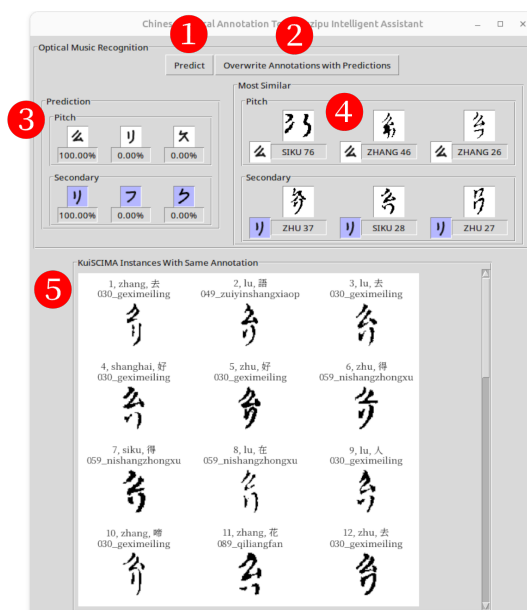


Figure 4.5: The Suzipu Intelligent Assistant widget allows for visualization and automatic labeling of notation instances.

1. The display notation selection allows for a selection of notation types in which the musical contents are displayed. The N button means that a modern typeface of the historical notation is displayed, while the button with the number refers to Chinese number notation *jianpu*, and the button with the five lines enables the staff notation display. The algorithms that are used for the conversion are described in Section 7.1.
2. This drop-down list can be used for transposing the music to adapt to user preferences regarding readability and playability. The transpositions are named according to their *jianpu* conventions, such that the lowest pitch ♭ is mapped to either 1, 2, 3, 4, 5, 6, or 7.
3. For the historical notation display, traditional reading order can be enabled, so that the displayed image is as close to the original layout as possible.
4. The export buttons allow for saving the rendered image or a transnotation in the *MusicXML* format.
5. In this widget, the rendered image is displayed. The rendering is in near real time, so that it immediately responds to user inputs.



Figure 4.6: The Notation Display window provides a variety of notations in which the musical contents can be displayed.

#### 4.5.2 The *Chinese Musical Notation Editor*

As opposed to the *Chinese Musical Notation Annotation Tool*, which serves the purpose of creating annotated images containing Chinese musical notations, the *Chinese Musical Notation Editor* is designed for digitally encoding pieces involving these notations in a purely symbolic manner.

As such, the two applications share not only functionalities and widgets, such as the notation selection widget or the mode information widget, but also the majority of the source code and involved algorithms. In Figure 4.6, the Main Window and the Notation Window are shown. Again, the widgets are marked with red numbers, and they are briefly described as follows:

1. The buttons in this widget are used to create a new file, open an already existing file, and save the currently active file to the hard disk. Note that not only symbolic JSON files created by the notation editor can be opened, but also annotated image JSON files as created by the annotation tool. During this process, only the purely symbolic data is extracted, so the information linking the symbolic contents with the images are lost.
2. The button `Edit Metadata...` opens a separate window, in which the piece can be endowed with information about the composer, the title, mode information, preface, and also information which does not fit into these categories.
3. Here, the notation display type, the transposition, and the reading order can be changed. The functionality is the same compared with the corresponding widget of the annotation tool. It affects the rendered notation in the Notation Window in real time.

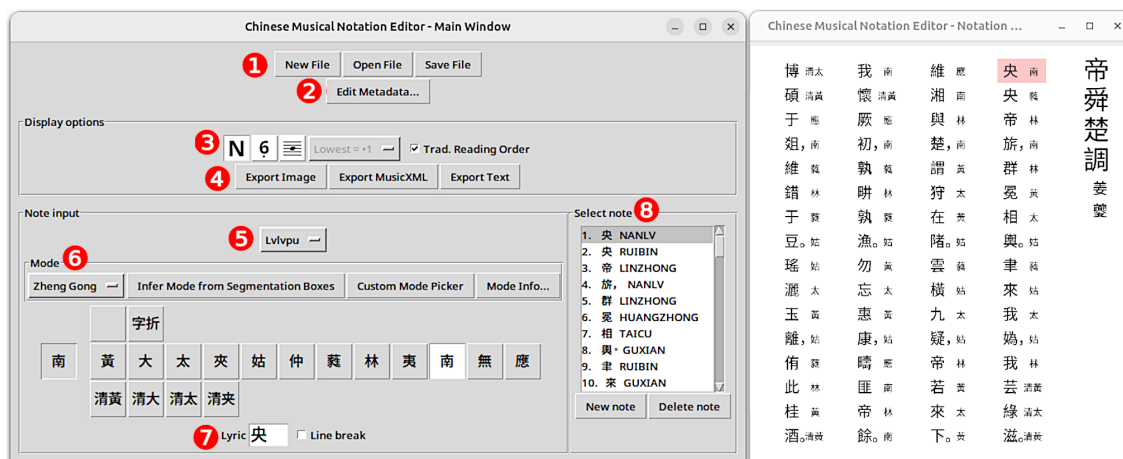


Figure 4.7: The Main Window and Notation Window of the notation editor. Here, the functionality is demonstrated for the *lülüpu* notation.

4. With the export buttons, the currently rendered image in the Notation Window, the MusicXML transnotation, and the contents of the piece in human-readable text form can be exported.
5. The notation selection widget allows to select between plain text and a variety of different Chinese historical notations, including *suzipu*, *lülüpu*, *jianzipu*.
6. This widget changes its appearance depending on the selected notation in the previous widget. Here, the piece's mode and the musical notation symbol for the currently selected syllable (or *note*) can be chosen. Note that this widget functions analogously to the one for the *Chinese Musical Notation Annotation Tool*.
7. In this widget, the current syllable's lyrics character is entered, and whether a line break occurs after the syllable.
8. Using the buttons *New note* and *Delete note*, individual syllables are created (and then edited later) or removed. It displays all notes in the piece, including their index, the lyric character, and the notation symbol's representation. The currently selected syllable is emphasized in the Notation Window using a red shading.

For further information how this notation editor can be used, Appendix D contains a step-by-step tutorial.

### 4.5.3 Interactive Notation Display

An integral feature of the graphical user interface (GUI)s presented in this dissertation is the rendering of the symbolic musical notations as an image. This process is characterized by special features to ensure good usability while at the same time being tailored to the properties of the involved Chinese notations.

For both the original notation display and the transnotation display, a custom engine is used to ensure near real-time rendering of the musical notations, this includes the historical Chinese notations and also contemporary notations such as *jianpu* notation or staff notation. Not only does this guarantee a uniform look of all renderings, but also results in a good responsiveness, since the displayed notation is adapted whenever the annotations are changed. Since the musical notations supported by the tool are all monophonic, the rendering is inspired by type printing techniques, where each instance is mapped to a type, the types are then arranged in a grid and form the final image in the end. The display of original notations is supported for all three notations, while at the current time, the transnotations can be displayed for *lülüpu* and *suzipu*.

The interactive transnotation display of *lülüpu* notation is straightforward by displaying the encoded pitch. For the *suzipu* notation, a conversion procedure (Algorithm 3) is used to determine the pitch depending on the piece’s mode. After this, the pitches are stored internally using the *music21* package (Cuthbert & Ariza, 2010) for Python, which provides all functions necessary for the efficient manipulation of pitches, such as instant transposition. This is also used in the transposition selection, with which the pitches can be moved up or down by an offset for increased readability or playability.

The *music21* package is also used for the export of the notation into the *MusicXML* format. Files of this type can easily be opened with music software, allowing for playback of the audio or the further processing of the pieces created with the GUI. It is important to note that this feature, like the transnotation display, is currently supported for the *suzipu* and *lülüpu* notations. For *suzipu*, the transnotation schema by Wu (2013) is used, while the *lülüpu* transnotation is based on each syllable rendered as a quarter note and the *zhezi* corresponding to the same pitch as the pitch before with a tenuto mark above.

## 4.6 Summary

In this chapter, we have established a digital representation of *suzipu* notation, and generalized it to the notation systems *lülüpu* and *jianzipu*, both of which have been described briefly. The digital representations faithfully reflect the features encoded in the historical notations.

In addition, the two GUIs for efficient creation of both purely symbolic representations and optical annotations of photographic reproductions of physical editions were described in detail. Both GUIs feature intelligent displays of the piece’s modal properties, the algorithms of which are explained later in Section 7.1. The *Intelligent Assistant* window of the optical annotation GUI contains intelligent features for a more efficient semi-automatic annotation workflow. These are thoroughly described in Chapter 6. Included are the clustering of annotated instances in Section 6.1, the *suzipu* and *lülüpu* classifiers in Sections 6.3 and 6.4 with calibrated output scores (Section 6.5), and feature-based similarity displays powered by UMAP clustering in Section 6.6.

In practice, the creation of the GUI tools in this Chapter, the KuiSCIMA dataset in Chapter 5, and the OMR methods in Chapter 6 were enhanced iteratively. This means that a first version of the GUI tool with limited intelligent features was developed and used to annotate a fraction of the whole dataset. With this data, preliminary OMR algorithms were created and then



included in the GUI tool, with which another fraction of the dataset could be annotated more efficiently, etc. pp.

Now that we have the adequate tools in our hands, we can proceed to Chapter 5 and investigate Research Question (RQ2). Here, we are going to understand how the KuiSCIMA dataset is designed, and how it is efficiently created using the previously introduced GUI tools.