VISUAL OVERVIEWS FOR SHEET MUSIC STRUCTURE

Frank Heyen[1] **Quynh Quang Ngo**[1] **Michael Sedlmair**[1]

[1] VISUS, University of Stuttgart, Germany

frank.heyen@visus.uni-stuttgart.de, michael.sedlmair@visus.uni-stuttgart.de

ABSTRACT

This document contains supplemental material for our ISMIR 2023 paper "Visual Overviews for Sheet Music Structure". For this publication and other material, see github.com/visvar/sheetmusic-overviews.

1. INTRODUCTION

2. DESIGN

Figure 2 summarizes the data processing and visualization pipeline of our design. We parse a sheet music file into notes, which we group into bars and sections. Notes that start at the same time are further grouped into harmonies. Through a distance metric (Section 2.2), we obtain a matrix with the distances between all possible pairs of the above groups of the same type. Based on this distance matrix, we use different methods to map distances to color (Section 2.3).

2.1 Data

We use sheet music encoded as *MusicXML* [1] files, as MusicXML allows representing a wide range of musical notations in full detail. Most common notation software allows exporting to MusicXML, such as *MuseScore* (musescore.org) and *Guitar Pro* (guitar-pro.com), for which there are abundant files online. Through a custom parser, we extract information about notes, bars, and textual annotations. This information provides us with different levels of detail: **Sections** represent the overall structure of a piece and are read from rehearsal annotations that are present in some files and easy to add to existing sheet music. Since the format is XML-based, anyone with a text editor can edit MusicXML files and add these annotations. **Bars** represent the finer structure of a piece and sometimes repeat within a section. **Harmonies** and chords – notes that are played at the same time – make up an even more detailed level and can also repeat within bars or sections. The highest level of detail distinguishes the individual notes. Figure 1e shows an example of these levels and how we can visualize them.

2.2 Similarity / Distance Metrics

Our approach works with any metric that takes the notes of a section or bar and returns a scalar similarity or distance value. In our related work section, we discussed existing similarity metrics for symbolic music. Some of these metrics do not support polyphony, while others require complete scores or additional annotations (such as chords) or assumptions on musical meaning. Metrics that are based on western tonal harmony [2, 3] would also not generalize to various cultures and genres.

We, therefore, designed the following simple but robust algorithm: First, we sort the notes of a part by their start time, where notes with equal start time will be sorted by ascending. Then, we map each note to its pitch as *MIDI* [4] number, resulting in one sequence of integers for each part. We then compute the *Levenshtein distance* [5] between all possible pairs of these sequences, optionally normalized by dividing through the longer sequence's length. The above steps result in a matrix of distances.

In one of our views, we also color harmonies by similarity, for which we need another distance matrix. For harmonies, the order of notes does not matter. We use the *Jaccard index* [6] to compute similarities between these sets of notes, which equals the ratio of intersection over union. For this metric, we only use the pitch class and ignore the note's octave.

2.3 Color Mappings

Once we have a distance matrix, we can use it to create a color mapping that respects these distances. We explored three alternative methods that use either a one-to-many comparison, dimensionality reduction, or hierarchical clustering. Figure 3 shows a comparison.

Comparison to a selected part. The first method colors bars by their distance to a selected bar. This selection is either made by the user or automatically when playing the piece, as the player selects the currently playing bar. To obtain colors, we linearly map the distances to a color scale, from 0 to the maximum distance (Figure 3a). Another mode only colors bars that are identical according to the metric, allowing to quickly spot how often and where a bar repeats.

Dimensionality reduction. Our second method uses *multi-dimensional scaling* (MDS) [7, 8], which tries to preserve distances and, crucially, accepts a distance matrix. The latter is important as we only have distances and no

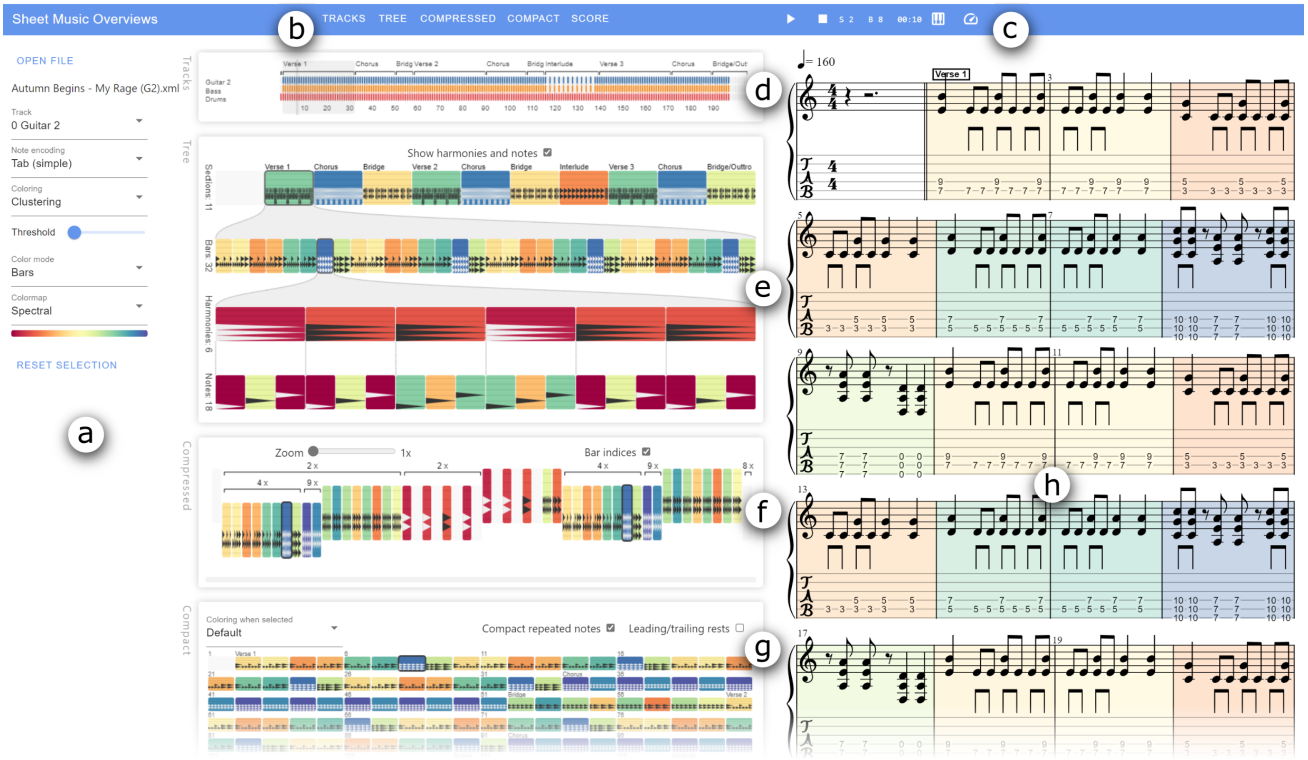


Figure 1: Screenshot of our design with all views (cut): a) data and visualization options, b) view selection, c) player, d) instrument/track overview, e) structure hierarchy, f) compressed repetitions, g) compact sheet, and h) complete score.

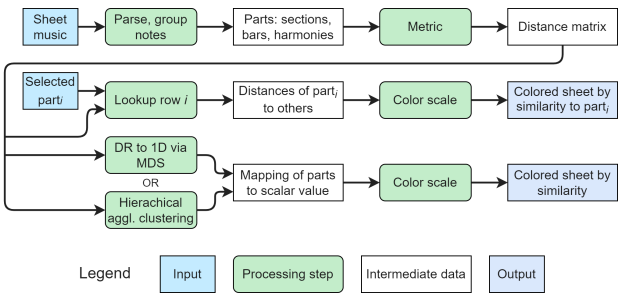
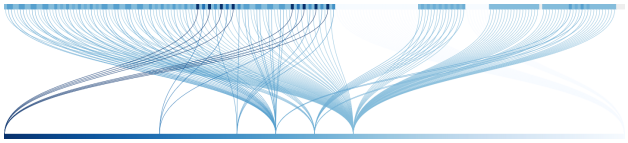


Figure 2: This pipeline of our approach shows how we compute similarity-based colors from input data. A *part* can be any sequence of notes in the piece, for example, a bar or a pre-defined section. The *selected part* is chosen by the user to compare it to all others.

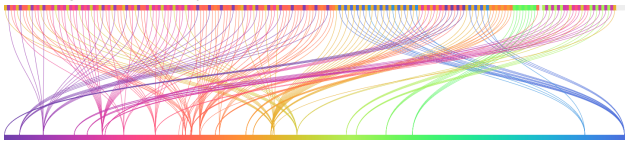
actual positions in a high-dimensional space, which some other dimensionality reduction algorithms require. We use MDS to project onto a one-dimensional space, that we can then linearly map to a color scale. This method leads to a coloring optimized for distances (Figure 3b).

Hierarchical clustering. As an alternative to the above approach with MDS, we designed a method that clusters similar parts together to then give each cluster the same color and similar clusters more similar colors. The method works as follows: Using our distance matrix, we compute *hierarchical agglomerative clustering* [9] with complete linkage, which gives us a binary tree. We then sort the leaves of the tree sorted from left to right, since leaves that are closer together are more similar. Linearly mapping each leaf's index in this sorted list to a color results in colors with equal distances within the color scale (see Figure 3d). Compared to the above method, the colors are therefore less accurate in representing distances, but easier to distinguish. The clustering can also be adapted by choosing a threshold distance for where to cut the hierarchy. This allows to steer how many clusters, and therefore colors, there will be (Figure 3e).

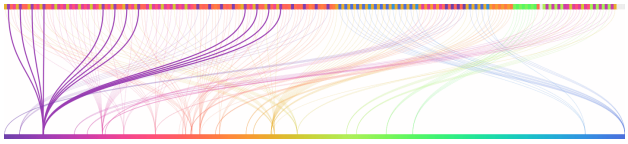
Color scales. Research on perception and visualization proposed a range of color scales specifically designed for visualization. Since there are different irreconcilable goals, no scale is appropriate for all tasks. We want to have many discernible colors, making rainbow color maps an obvious choice. While rainbow color maps have been criticized [10] as being "confusing, obscuring, and mislead-



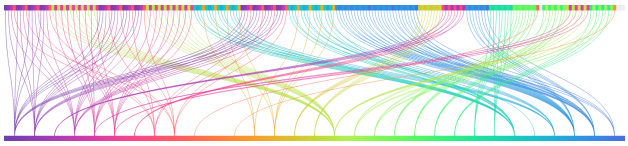
(a) Coloring based on similarity to a selected bar (darker blue indicates higher similarity).



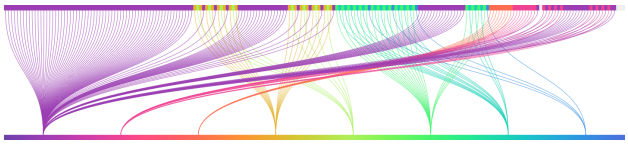
(b) Mapping from bars to colors with DR. Colors are distributed according to similarity, similar bars will have more similar colors.



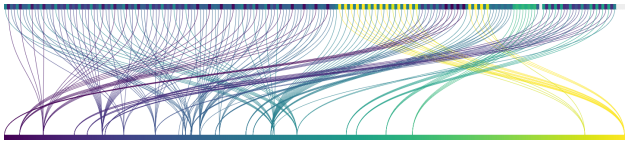
(c) In this image, highlighting a specific color and all measures with this color shows where the exact same bar repeats within the piece.



(d) Mapping from bars to colors with clustering. Colors are more evenly distributed, but similarities are represented less accurately. Repeating patterns are easier to see with this mapping.



(e) Setting the cut-off threshold to a value > 0 reduces the number of clusters and therefore colors – and reveals coarser patterns more clearly.



(f) With a scale that is more appropriate for color vision deficiency, some patterns are less clearly visible when compared to Figure 3b.

Figure 3: Comparison of our different color mapping methods. The row at the top of each image shows the bars of an example piece as small rectangles sorted by time, the color scale is shown on the bottom, and curves connect each bar to its color.

ing", they have been shown to work well for some circumstances [11–13]. Some rainbow color maps are cyclic, meaning the lowest and highest values are assigned similar colors.

For users with a color vision deficiency, other multi-hue scales with fewer hues and therefore less discernible colors can be used, such as cividis [14]. When color is used to compare different values or intervals, a color scale needs to accurately represent distances between values. For this task, single-hue scales or interpolations between two hues are appropriate, but further reduce the number of discernible colors. Although the number of distinguishable colors is limited, there are enough for our use case, as the number of different parts in a piece is limited. Furthermore, since colors are distributed by similarity, indistinguishable colors should only be assigned to very similar parts.

We chose a subset of the color scales provided by D3's [15] d3-scale-chromatic (Figure 4). To avoid the above issues, we use the *spectral* scale as default but provide others, including rainbow and accessible ones, as well as a single-hue scale and the option not to show any coloring at all (*white*). For exact comparisons (see Section 2.3), we use the single-hue *blues* scale only.

Figure 4: We include different color scales. The choice depends on the current task and the user's individual color vision limitations.

Further Considerations. We considered using dimensionality reduction to a two-dimensional or three-dimensional space to map values to an HSL or RGB color space directly. For HSL, we could choose from the hue-saturation or hue-lightness planes; for RGB, we would use the entire space. Both approaches would allow for more degrees of freedom for the dimensionality reduction algorithm, leading to less distortion. A major drawback, however, is that colors would not necessarily be equally distributed perceptually. We also considered creating our own color scale by using a straight line through the CIE LAB space but discarded this idea, as results were not perceptually uniform at the lower and upper values, and we felt more confident using existing, reviewed scales [14].

2.4 Implementation

We implemented our prototype as a web app using Svelte, so users can open their own sheet music in the form of MusicXML [1] files to analyze them in their web browser without any further downloads or setup. We use the DRUID$_{JS}$ [16] library to compute dimensionality reduction and clustering. Visual mappings and color scales are taken from D3 [15] to make sure they are appropriate for visualization. We use OpenSheetMusicDisplay for sheet music rendering, which directly renders MusicXML.

2.5 Further Ideas

Instead of only sheet music, our coloring could be used to augment more abstract music visualizations as well. Figure 5 shows how arc diagrams can be extended by coloring the arcs that connect repeating sections with the average color of the sections, making the connections even clearer. We also experimented with similarity-based layouts using MDS [7, 8], where we position bars of a piece closer together when they are more similar and vice versa (Figure 6). To be able to display small visualizations of the bars' notes, we gridified the positions using Hagrid [17]. We additionally connect the bars with links to retain the temporal ordering. The links are bowed to encode their direction [18].

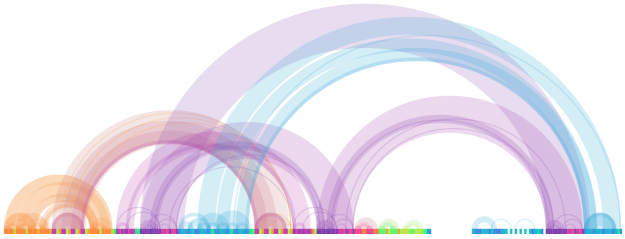


Figure 5: Arc diagrams [19] augmented through the average color of repeated bar sequences. White gaps show empty bars.

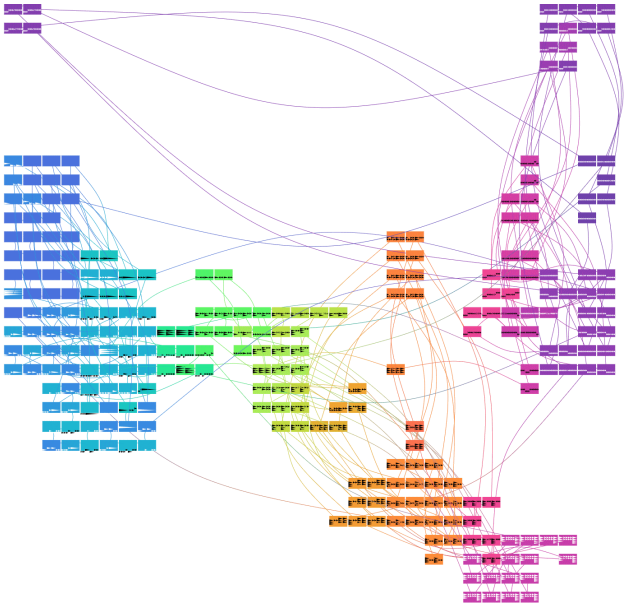


Figure 6: Gridified dimensionality reduction layout with each bar colored by similarity and displayed with our simplified encoding. Bowed links show the ordering.

3. ACKNOWLEDGMENTS

This work was funded by the Cyber Valley Research Fund and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – SFB TRR 161, project A08.

4. REFERENCES

[1] M. Good *et al.*, "MusicXML: An internet-friendly format for sheet music," in *XML Conf. and expo*. Citeseer, 2001, pp. 03–04.

[2] W. B. De Haas, M. Rohrmeier, R. C. Veltkamp *et al.*, "Modeling harmonic similarity using a generative grammar of tonal harmony," in *Proc. of the Tenth Int. Conf. on Music Information Retrieval (ISMIR)*, 2009.

[3] W. B. De Haas, J. Rodrigues Magalhães, R. C. Veltkamp *et al.*, "HarmTrace: Improving harmonic similarity estimation using functional harmony analysis," in *Proc. of the 12th Int. Conf. on Music Information Retrieval (ISMIR)*, 2011.

[4] R. A. Moog, "MIDI: Musical instrument digital interface," *Journal of the Audio Engineering Society*, vol. 34, no. 5, pp. 394–404, 1986.

[5] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[6] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[7] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[8] ——, "Nonmetric multidimensional scaling: A numerical method," *Psychometrika*, vol. 29, no. 2, pp. 115–129, 1964.

[9] A. Lukasova, "Hierarchical agglomerative clustering procedure," *Pattern Recognition*, vol. 11, no. 5, pp. 365–381, 1979.

[10] D. Borland and R. M. Taylor Ii, "Rainbow color map (still) considered harmful," *IEEE Computer Graphics and Applications (CG&A)*, vol. 27, no. 2, pp. 14–17, 2007.

[11] K. Reda and D. A. Szafir, "Rainbows revisited: Modeling effective colormap design for graphical inference," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 27, no. 2, pp. 1032–1042, 2021.

[12] K. Reda, A. A. Salvi, J. Gray *et al.*, "Color nameability predicts inference accuracy in spatial visualizations," *Computer Graphics Forum (CGF)*, vol. 40, no. 3, pp. 49–60, 2021.

[13] M. Wattenberg, F. B. Viégas, and K. Hollenbach, "Visualizing activity on Wikipedia with chromograms," in *Human-Computer Interaction – INTERACT 2007*. Springer, 2007, pp. 272–287.

[14] J. R. Nuñez, C. R. Anderton, and R. S. Renslow, "Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data," *PLOS ONE*, vol. 13, no. 7, pp. 1–14, 2018.

[15] M. Bostock, V. Ogievetsky, and J. Heer, "D^3: Data-driven documents," *IEEE Trans. on Visualization and Computer Graphics (TVCG)*, vol. 17, no. 12, pp. 2301–2309, 2011.

[16] R. Cutura, C. Kralj, and M. Sedlmair, "Druid$_{JS}$ — a javascript library for dimensionality reduction," in *IEEE Visualization Conf. (VIS)*. IEEE, 2020, pp. 111–115.

[17] R. Cutura, C. Morariu, Z. Cheng *et al.*, "Hagrid — Gridify Scatterplots with Hilbert and Gosper Curves," in *Int. Symp. Visual Information Communication and Interaction (VINCI)*, 2021.

[18] D. Holten and J. J. van Wijk, "A user study on visualizing directed edges in graphs," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, ser. CHI '09. ACM, 2009, p. 2299–2308.

[19] M. Wattenberg, "Arc Diagrams: Visualizing structure in strings," in *IEEE Symp. Information Visualization (INFOVIS)*. IEEE, 2002, pp. 110–116.