PYDACTYL: A PYTHON FRAMEWORK FOR PIANO FINGERING

David A. Randolph[1] **Justin Badgerow**[2] **Christopher Raphael**[3] **Barbara Di Eugenio**[1]
[1] University of Illinois at Chicago [2] Elizabethtown College [3] Indiana University Bloomington
drando2@uic.edu

ABSTRACT

A Python framework, "Pydactyl," supporting the rapid development of piano fingering models, is described and demonstrated. Leveraging the popular music21 toolkit, this object-oriented framework standardizes problem decomposition, data formats, fingering segment combination, and evaluation methods, while supporting automated and manual phrase segmentation. Reference implementations of three published models are included.

1. OVERVIEW

While pianists often value the fingering advice in editorial scores, the majority of piano music includes no such guidance. Even when present, the static advice in paper scores frequently proves ill-suited to the individual pianist. Thus, especially in light of the volume of freely available repertoire available online, the time is ripe for computer models that automatically generate piano fingering advice.

Toward this end, we introduce Pydactyl, an object-oriented Python 3 framework to accelerate development of computational piano fingering models and to encourage the consistent evaluation thereof. The framework includes ready-to-use implementations to address the basic needs of developers working in the domain. Common tasks like standardized data formatting, input processing, database connectivity, phrase segmentation, model evaluation, and system baselining are addressed, enabling researchers to focus on the specifics of their models.

Hosted at `https://github.com/dvdrndlph/pydactyl`, Pydactyl is open-source. To install, type this:

```
pip3 install pydactyl
```

2. FOCUS ON SEGMENTATION

Radicioni et al. [7] make a convincing case that segmented (localized) fingering solutions lead to improved global solutions. Citing [3, 5, 10], they argue "expressive aspects of performance descend from the performer's analysis of" structure, especially phrases (in the colloquial sense of segmenting a piece into musical ideas), and that fingering is one such aspect. We therefore support phrase segmentation in our base "Dactyler" class, from which all models implemented in the framework derive. To enable derived classes to focus solely on generating advice for individual phrases, the base class also provides methods for combining segmented advice into a ranked list of global solutions.

Since we leverage the music21 toolkit [2] to build our framework, Pydactyl will ultimately accept all input formats supported by music21. But current testing has focused on abc notation [11] and specifically on our own abcD [8], now expanded to support phrase segmentation (via commas, semicolons, and periods) in its abcDF fingering specifications.

Including phrase segmentation as a component of our fingering specification is not done without reservations. On the one hand, phrasing is clearly an input into the fingering problem and does not constitute an essential part of an output fingering solution. Indeed, varying notions of how music should be phrased likely contribute to variability in fingering decisions, and this contributes to the difficulty in solving the problem in a generalized way.

But as a practical matter, phrasing suggestions are exceedingly rare, even in editorial scores. Worse, in virtually all piano notation, the same symbol is used ambiguously to represent a phrase or a slur. We therefore elect to include phrasing annotation in our fingering language to make it easy to capture such data, especially from abcDE [8] users.

Clearly, phrasing is a matter of interpretation, and fingering decisions are made in support of executing phrases. So coupling fingering and phrasing data together is in an important sense justified. In addition, when MIDI and other input file formats are fully supported in our tool chain, we can use accompanying abcD header files to convey both detailed fingering and detailed segmentation information. We note that an annotation with only segmentation data still constitutes valid abcDF. Because the 2.1 abc standard [11] provides unique notations for three levels of phrase segmentation, we offer the same here.

As it turns out, combining fingered segments reduces to a k-shortest path search problem, where the arc costs are simply the costs associated with each suggested segment fingering. How these segment fingering costs are calculated affects the rankings of the complete end-to-end suggestions. The simplest method is perhaps the truest: The cost of the entire sequence is simply the sum of the individual segment costs. However, this will make the most diffi-

cult and longest segment fingerings the most consistent in any returned solutions. Moreover, these are likely to be the very segments for which a user would appreciate a variety suggestions. These would also likely be the segments that would have the least agreement among pianists. Given this, it might be preferred in some applications to normalize the cost by the number of notes in the segment or even to focus on the ordinal rank of segment suggestions, to ensure we see similar variability for each segment. These three global costing approaches are supported in Pydactyl.

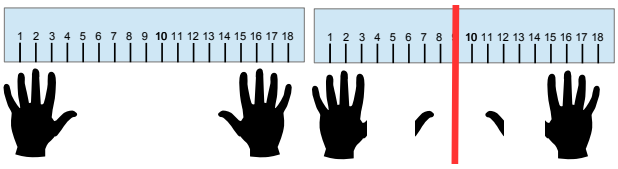
3. EVALUATION METHODS

The framework also supports a set of evaluation metrics to compare model output to gold-standard corpora. These include three edit-distance metrics: a standard Hamming distance; a more "natural" weighted measure that penalizes each individual finger deviation by the absolute difference in the expected and actual fingering numbers; and another measure that infers hand re-positioning from deviations involving the thumb and penalizes these differences more severely. We also provide a straightforward pivot-alignment measure along with one final method that requires multiple requests for advice to assess.

This last "re-entry cost" method first checks the alignment of two fingering sequences. At the first deviation detected, it imposes an edit-distance cost and re-generates advice on the subsequence of notes from the point of deviation, *constraining the fingering on the first note* to match the gold standard. This process repeats, with costs aggregating, until all notes are processed. To allow such operations, the framework requires all models to support constraining the first and last fingerings for any segment.

```
model = Parncutt(segmenter=ManualDSegmenter(),
segment_combiner="cost")
d_corpus = DCorpus(paths=["/tmp/prelude02.abcd"])
model.load_corpus(d_corpus=d_corpus)
advice = model.advise()
# Gold-standard embedded in input file.
hamming_dists = model.evaluate_strike_distance()
```

Listing 1: Pydactyl usage example.



(a) "Natural." **(b)** Re-position penalizing.

Figure 1: Alternative edit-distance measures provided by Pydactyl. The distance between any two fingers on a ruler is assessed each time a deviation is detected when comparing one suggested fingering with another. *Credit: Hand by Baranovskiy [1].*

4. REFERENCE IMPLEMENTATIONS

We provide slightly enhanced implementations of the models by Parncutt et al. [6], Sayegh [9], and Hart et al. [4] for baseline reference and to demonstrate Pydactyl usage.

5. ACKNOWLEDGMENTS

This work has been supported by a Provost's Award from UIC and a Faculty Grant from Elizabethtown College.

6. REFERENCES

[1] Dmitry Baranovskiy. Hand. `https://thenounproject.com/search/?q=creator=9767&i=5011`. Copyright information: CC-BY 3.0 license. Accessed: 2017-04-25.

[2] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 637–642, Utrecht, Netherlands, 2010.

[3] Carolyn Drake and Caroline Palmer. Skill acquisition in music performance: Relations between planning and temporal control. *Cognition*, 74(1):1–32, 2000.

[4] Melanie Hart, Robert Bosch, and Elbert Tsai. Finding Optimal Piano Fingerings. *The UMAP Journal*, 21(2):167–177, 2000.

[5] Caroline Palmer. Music performance. *Annual Review of Psychology*, 48:115–138, 1997.

[6] Richard Parncutt, John A. Sloboda, Eric F. Clarke, Matti Raekallio, and Peter Desain. An ergonomic model of keyboard fingering for melodic fragments. *Music Perception*, 14(4):341–382, 1997.

[7] Daniele Radicioni, Luca Anselma, and Vincenzo Lombardo. A segmentation-based prototype to compute string instruments fingering. In *Proceedings of the 1st Conference on Interdisciplinary Musicology*, Graz, Austria, 2004.

[8] David A. Randolph and Barbara Di Eugenio. Easy as abcDE: Piano fingering transcription online. In *Extended Abstracts for the Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conference*, 2016.

[9] Samir I. Sayegh. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal*, 13(3):76–84, 1989.

[10] John Sloboda. *The Musical Mind: The Cognitive Psychology of Music*. Oxford University Press, Oxford, UK, 1985.

[11] Chris Walshaw. The abc Music Standard 2.1. `http://abcnotation.com/wiki/abc:standard:v2.1`, 2011. Accessed: 2016-06-28.