Towards an Integration of OMR Models in Pattern Improvisation

Francesco Ardan Dal Rì[1, 2] and Nicola Conci[1]

[1] *Department of Information Engineering and Computer Science - DISI, University of Trento, Italy*
[2] *Conservatory F.A. Bonporti, Trento, Italy*

Abstract

In this paper, we present a prototype for an interactive musical system, which allows the user to manage musical patterns in real-time using images of printed music in traditional Western notation. Monophonic scores are interpreted by a pre-trained Optical Music Recognition model, converted into numeric sequences, and sent to a pattern generator/synthesizer. This prototype constitutes a novel proposal, integrating deep learning and computer vision techniques into music making and promoting interactivity across disciplines.

1 Introduction

Musical composition through pattern manipulation has a long tradition, ranging from the earliest experiments in algorithmic music to the latest computer music practices [Magnusson & McLean, 2018]. In the last two decades, the emergence of live coding as a performative practice has positioned the use of musical patterns at a prominent role in musical creation. In this context, a great variety of libraries and dedicated environments have been presented, each characterized by a specific form of notation capable of controlling the evolution over time of these patterns (e.g [Kirkbride, 2016, Roberts & Kuchera-Morin, 2012, McLean & Wiggins, 2010]). Regarding the notation of melodic-rhythmic structures, however, the traditional notation - here referred to as Common Western Music Notation (CWMN) - is still particularly effective.

On the other hand, the recent advances in deep learning have allowed for the development of models able to automatically interpret the content of printed music, which have proven to be particularly effective in the case of monophonic scores (e.g [Liu et al., 2021, Castellanos et al., 2020]).

Our proposed prototype combines these two trends into a single system that uses an Optical Music Recognition (OMR) model to convert score images into patterns that can be played by a digital synthesizer. This paper is structured as follows: in Section 2 we present the relevant related work in the fields of pattern music, notations, and OMR; in Section 3 we briefly describe the implementations of the prototype; we then discuss the usage and limitations, and eventually conclude outlining possible future developments.

2 Background

In this section, we briefly introduce several relevant works and concepts from which we started to design our system.

2.1 Pattern music

In the musical context, the term *pattern* generally refers to a well-recognizable melodic-rhythmic cell, which repeats itself over time. In the last century, examples of compositions through the creation, superimposition, and permutation of patterns can be found both in the field of instrumental music (significant examples are *In C* by Terry Riley or *Clapping Music* by Steve Reich), and in that of computer music (for example in the works of Grossi [Mori, 2015] or Spiegel [Spiegel, 1981]). Although the use of musical patterns has been explored since the first experiences of algorithmic music, in the last two decades their use has become particularly prominent in the context of live coding, a performative practice in which the performer improvises music on the computer by compiling instructions in real-time in the form of code [Magnusson & McLean, 2018]. Most of the dedicated libraries and environments (e.g. TidalCycles [McLean & Wiggins, 2010], Gibber [Roberts & Kuchera-Morin, 2012], FoxDot

[Kirkbride, 2016], ect) allow the user to generate patterns that are synchronized and repeated cyclically. While these improvisations are generally performed from scratch, another popular approach involves specifying a series of patterns in advance and simply playing them in a given order during the performance. Magnusson defines this approach as "weak coding" [Magnusson, 2014a].

At this stage, our system allows for a similar compositional approach, in which the user can pre-define a palette of patterns and play them to his or her liking.

2.2 Notations

In Section 2.1, we have seen that there are several environments that allow users to manage musical patterns in real time. Each of them requires the use of a specific form of notation. Various ways of encoding music have been proposed: some systems allow the user to specify a melodic-rhythmic pattern through pairs of numerical values, respectively pitch (note or frequency) and duration (absolute or relative), as in SuperCollider [McCartney, 2002]; others implement specific *mini-notations* (e.g. TidalCycles [McLean & Wiggins, 2010] or ixi lang [Magnusson, 2011]), or make use of graphical objects or visual feedback (e.g. Betablocker or Scheme Bricks [McLean et al., 2010]). These notations are often designed to be also applicable to control parameters, effects, functions, etc. However, they can result quite difficult to interpret, especially in the case of sequences of a certain length or density [Blackwell & Collins, 2005]. Indeed, writing instructions requires a certain abstraction, that is, to formalize musical thought into a symbolic-syntactic notation capable of being understood by the software interpreter. As for the notation of twelve-tone equal tempered pitch and metrical rhythm and duration musical cells, CWMN can prove to be more effective [Magnusson, 2014b], but its use is currently overlooked.

Therefore, we have decided to integrate it into our work.

2.3 Optical Music Recognition

OMR has been defined as "a field of research that investigates how to computationally read music notation in documents" [Calvo-Zaragoza et al., 2020]. Although it has been explored for over 50 years, research in this field has advanced considerably in the last decade thanks to the adoption of deep learning techniques (e.g. [Liu et al., 2021, Baró et al., 2019, Castellanos et al., 2020]). Nowadays, various commercial softwares exist that integrate OMR for score scanning (e.g. SharpEye [1], PhotoScore [2], SmartScore [3]). Nevertheless, OMR still involves a variety of non-trivial tasks to be addressed. In the first place, the set of semantic relationships between the various musical symbols - as in the case of polyphonic music, articulation marks, or irregular rhythms - can be particularly complex. Furthermore, the general lack of large labeled datasets with balanced classes makes it difficult to deal with the great variety of symbols in CWMN [Novotnỳ & Pokornỳ, 2015]. As shown by Shatri and Fazekas, most works in the OMR field are focused on the encoding of monophonic scores, in which neural networks are able to achieve excellent performance [Shatri & Fazekas, 2020].

Since pattern music is generally composed of a series of monophonic and quite simple musical structures, we have decided to include one of these neural networks in our system.

3 Implementation

The system proposed consists of three main modules: an OMR model, which converts the images into semantic notation, a translator/OSC parser module, which converts the semantic notation into messages suitable for the pattern generator, eventually sending them to the server via OSC, and a synthesizer module, which plays the notes as they are sequenced (Figure 1).

In this section, we briefly describe each module.

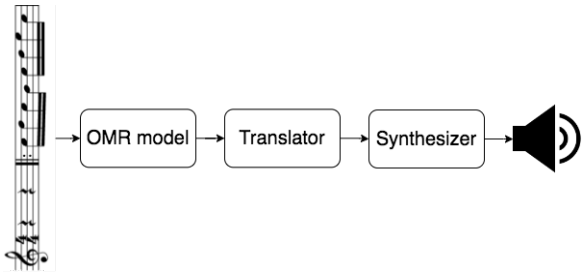


Figure 1: *Overall system pipeline*

3.1 OMR model

For this prototype, we decided to use a pre-trained, end-to-end, and open-source solution [Calvo-Zaragoza et al., 2017]. The model is trained on a series of monophonic musical incipits, namely the PrIMuS dataset [Calvo-Zaragoza & Rizo, 2018], and therefore proves to be sufficiently accurate for our needs. The model consists of a feed-forward Convolutional Recurrent Neural Network (CRNN), which extracts the individual features and converts them into discrete sequences of musical symbols. Further implementation details can be found in the original paper [Calvo-Zaragoza et al., 2017].

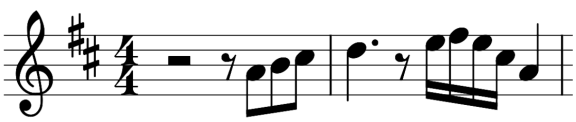
[1] http://www.visiv.co.uk/

[2] https://www.neuratron.com/photoscore.htm

[3] https://www.musitek.com/smartscore-pro.html

The model receives as input an image of a monophonic score on a single staff, and is sensitive to both clef and key signature. Consistently with the dataset used in the training phase by the authors of the original work, we consider inputs of a maximum of 4 bars in length. As output, we obtain a semantic encoding, which contains a list of all the musical symbols detected by the network in an extended format (Figure 2).

(a)



(b)

```
clef-G2, keySignature-DM, timeSignature-4/4,
rest-half, rest-eighth, note-A4_eighth,
note-B4_eighth, note-C#5_eighth, barline,
note-D5_quarter., rest-eighth, note-E5_sixteenth,
note-F#5_sixteenth, note-E5_sixteenth,
note-C#5_sixteenth, note-A4_quarter, barline
```

Figure 2: *An example of an input image (a) and the respective output in semantic notation (b)*

3.2 Translator

The translation module is responsible for converting the OMR model output into a format that the pattern generator for the synthesizer can understand, and for sending it through the Open Sound Control (OSC) protocol. The translation is performed by directly mapping a fixed vocabulary of durations and notes. The durations are considered to be relative to the quarter (e.g. quarter = 1, eighth = 0.5, etc.), while the notes are mapped to MIDI notes (e.g. C4 = 60, Db4 = 61, etc.). Rests are represented by the symbol "\" (Figure 3). We preferred to insert this translation module rather than retrain the model end-to-end for two main reasons: first, the retraining would have required the entire dataset to be completely reprocessed. Even if end-to-end retraining would be necessary for a possible definitive system, as specified in 5, at this stage we found semantic notation flexible enough to be easily translated into different types of notation.

3.3 Synthesizers

The synthesis engine, implemented in the popular open-source software SuperCollider (SC)[4], is composed of seven instruments: two additive synthesizers (characterized by a different type of envelope), two pitched single-oscillator (one octave up and one octave down, respectively), one subtractive, one FM and one producing pink noise (ideally simulating a simple drum). These instruments are recalled by the OSC messages sent by the translation module, while the related melodic-rhythmic patterns are recorded and sequenced on a quantized rhythmic grid. By default, a maximum of eight patterns can be played at the same time. In addition, the patterns are indexed: in this way, the user can activate/deactivate/overwrite specific patterns. In this first version, we have decided to keep the timbre characteristics of each instrument fixed: the user can therefore decide which instrument to play the input pattern.

[4]https://supercollider.github.io/

(b)

```
clef-G2, keySignature-DM, timeSignature-4/4,
rest-half, rest-eighth, note-A4_eighth,
note-B4_eighth, note-C#5_eighth, barline,
note-D5_quarter., rest-eighth, note-E5_sixteenth,
note-F#5_sixteenth, note-E5_sixteenth,
note-C#5_sixteenth, note-A4_quarter, barline
```

(c)

```
['\', '\', 57, 59, 61, 62, '\', 64, 66, 64, 61, 57]
[2, 0.5, 0.5, 0.5, 0.5, 1.5, 0.5, 0.25, 0.25, 0.25, 0.25, 1]
```

Figure 3: *An example of a semantic notation (b) translated into value arrays suitable for SuperCollider (c)*

4 Usage and limitations

As for the practical use of the system, it is first necessary to load a series of N images containing short musical sequences, which will be recalled with their index. These images constitute the palette of different musical sequences that can be converted into patterns. The user can input instructions consisting of four numbers, specifically representing: 1) image [1 - N]; 2) pattern index [0 - 7]; 3) instrument number [0 - 7]; 4) amplitude [0. - 1.]. The script is constantly waiting for instructions from the user, who at any time can decide to allocate a new pattern or overwrite an existing one. Note that the interpretation/translation system, on an 8GB commercial CPU, takes ∼200ms to send the message to the SC server. Since the pattern generator is by default quantized to the quarter note and set to a tempo of 60 b.p.m., this latency has to be taken into account.

As for the limitations, the model used is currently not very robust in the presence of numerous ledger lines and particularly high musical densities. However, being sensitive to the keys, it is already possible to obtain a sufficient pitch range: in case the user needs more extended sounds, we have inserted two transposing instruments (see 3.3). As for the rhythmic density, we have decided not to consider durations lower than thirty-second. While we recognize this as a limitation, shorter durations are rarely used in pattern music. Finally, the available prototype does

not allow direct manipulation of musical patterns or control over the parameters of the instruments.

5 Conclusions and future work

In this paper, we have presented a working prototype of a pattern music improvisation system starting from musical fragments in CWMN. We believe that the system proposed in this paper represents a contribution to developing further systems that can integrate Western musical notation in the context of interactive music production, constituting a possible creative stimulus, especially for more traditionally trained musicians, or a possible tool for music education. In addition, the prototype lends itself to various expansion possibilities, such as integration into existing systems/environments, both software and hardware, and the addition of additional interfaces/control possibilities. As a future work, the main objective will be to work on a new OMR model trained on fragments of handwritten scores in an end-to-end fashion, which also allows taking into account more complex musical symbols and relationships, in order to create an interactive system with more expressive possibilities with which the user can interact directly by writing music in real-time.

References

[Baró et al., 2019] Baró, A., Riba, P., Calvo-Zaragoza, J., & Fornés, A. (2019). From optical music recognition to handwritten music recognition: A baseline. *Pattern Recognition Letters*, 123, 1–8.

[Blackwell & Collins, 2005] Blackwell, A. F. & Collins, N. (2005). The programming language as a musical instrument. In *PPIG* (pp.1̃1).

[Calvo-Zaragoza & Rizo, 2018] Calvo-Zaragoza & Rizo, D. (2018). Camera-primus: Neural end-to-end optical music recognition on realistic monophonic scores. In *Ismir2018* (pp. 248–255).

[Calvo-Zaragoza et al., 2020] Calvo-Zaragoza, J., Jr, J. H., & Pacha, A. (2020). Understanding optical music recognition. *ACM Computing Surveys (CSUR)*, 53(4), 1–35.

[Calvo-Zaragoza et al., 2017] Calvo-Zaragoza, J., Valero-Mas, J. J., & Pertusa, A. (2017). End-to-end optical music recognition using neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR* (pp. 23–27).

[Castellanos et al., 2020] Castellanos, F. J., Calvo-Zaragoza, J., & Inesta, J. M. (2020). A neural approach for full-page optical music recognition of mensural documents. In *ISMIR* (pp. 558–565).

[Kirkbride, 2016] Kirkbride, R. (2016). Foxdot: Live coding with python and supercollider. In *Proceedings of the International Conference on Live Interfaces* (pp. 194–198).

[Liu et al., 2021] Liu, A., Zhang, L., Mei, Y., Han, B., Cai, Z., Zhu, Z., & Xiao, J. (2021). Residual recurrent crnn for end-to-end optical music recognition on monophonic scores. In *Proceedings of the 2021 Workshop on Multi-Modal Pre-Training for Multimedia Understanding* (pp. 23–27).

[Magnusson, 2011] Magnusson, T. (2011). The ixi lang: A supercollider parasite for live coding. In *ICMC*.

[Magnusson, 2014a] Magnusson, T. (2014a). Herding cats: Observing live coding in the wild. *Computer Music Journal*, 38(1), 8–16.

[Magnusson, 2014b] Magnusson, T. (2014b). Scoring with code: Composing with algorithmic notation. *Organised Sound*, 19(3), 268–275.

[Magnusson & McLean, 2018] Magnusson, T. & McLean, A. (2018). Performing with patterns of time. In R. T. Dean (Ed.), *The Oxford Handbook of Algorithmic Music* chapter 14, (pp. 245–266). Oxford University Press.

[McCartney, 2002] McCartney, J. (2002). Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4), 61–68.

[McLean et al., 2010] McLean, A., Griffiths, D., Collins, N., & Wiggins, G. (2010). Visualisation of live code. *Electronic Visualisation and the Arts (EVA 2010)*, (pp. 26–30).

[McLean & Wiggins, 2010] McLean, A. & Wiggins, G. (2010). Tidal–pattern language for the live coding of music. In *Proceedings of the 7th sound and music computing conference* (pp. 331–334).

[Mori, 2015] Mori, G. (2015). Pietro grossi's live coding. an early case of computer music performance. *ICLC2015 Proceedings*, (pp. 125–132).

[Novotnỳ & Pokornỳ, 2015] Novotnỳ, J. & Pokornỳ, J. (2015). Introduction to optical music recognition: Overview and practical challenges. In *DATESO* (pp. 65–76).

[Roberts & Kuchera-Morin, 2012] Roberts, C. & Kuchera-Morin, J. (2012). Gibber: Live coding audio in the browser. In *ICMC*, volume 11 (pp.6̃).

[Shatri & Fazekas, 2020] Shatri, E. & Fazekas, G. (2020). Optical music recognition: State of the art and major challenges. *arXiv preprint arXiv:2006.07885*.

[Spiegel, 1981] Spiegel, L. (1981). Manipulations of musical patterns. *Proceedings of the Symposium on Small Computers and the Arts*, (pp. 19–22).