

An archive-exploration system for the Hunting Songs of the Lakeland Fell Packs

Mary Emmett*
mh687@york.ac.uk
Department of Music
University of York
UK

Zongyu Yin
zy728@york.ac.uk
Department of Computer Science
University of York
UK

Tom Collins
tom@musicintelligence.co
Department of Music,
University of York, UK;
Music Artificial Intelligence
Algorithms, Inc., USA

ABSTRACT

From everyday users to expert archivists, there is a keen interest in being able to search digital library catalogues via domain-specific languages and queries. This paper presents a proof-of-concept system that enables library users to search musically for songs or pieces that match to sung, hummed, or typed note queries. We chose to work with a corpus of 80 Lakeland Hunting Song tunes (Items) that have been collected recently, and for which the archivist wanted to make a resource that was easily searchable in terms of its musical content. Existing systems such as that at folktunefinder.com are robust to rhythmic changes but not to missing or interpolated notes, so we employed a geometric hashing approach as the basis for determining matches and as a potential improvement on existing work. An evaluation indicates that the system works quickly and effectively. We discuss some of the examples arising from its use, and indicate how future work building on this proof-of-concept system may make it possible to search other collections of songs or pieces in a more natural, musical way. Our archive-exploration system is available for trying out at <https://hse.glitch.me>.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; • **Information systems** → *Content analysis and feature selection*; **Query representation**; **Search interfaces**; **Retrieval efficiency**; • **Computer systems organization** → *Real-time system architecture*.

KEYWORDS

domain-specific queries, music search, geometric hashing, ethnomusicology, folk music, Node.js

ACM Reference Format:

Mary Emmett, Zongyu Yin, and Tom Collins. 2021. An archive-exploration system for the Hunting Songs of the Lakeland Fell Packs. In *8th International Conference on Digital Libraries for Musicology (DLfM2021)*, July 28–30, 2021.

*All authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DLfM2021, July 28–30, 2021, Virtual Conference, GA, USA

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8429-2/21/07...\$15.00
<https://doi.org/10.1145/3469013.3469014>

Virtual Conference, GA, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3469013.3469014>

1 INTRODUCTION

Computational approaches to folk music research are well established, and have been used to shed light on stability and variation in folk song transmission (e.g., [16]). Recent research [13] has focused on the Hunting Songs of the English Lakeland Fell Packs – a folk song tradition that up until now has not been the sole focus of any academic study.¹ As indicated and defined in Table 1, this research involved collecting 313 songs, which correspond with 147 tunes. When attempting to catalogue the collection for musical analysis it became clear that there was no existing system that could be used to upload, organise, and store the tunes so that they might be a) searched and b) explored for musically interesting correspondences. Currently, this collection of songs is detailed over various spreadsheets, lyrics in word documents, and tunes in the music notation software Sibelius. To date, it has not been possible to create a convenient way of systematically discovering new knowledge such as which songs share a particular tune, and this is a cause of frustration to the current authors, but also to other ethnomusicologists and archivists seeking a more coherent and powerful archive-exploration approach.

Beyond the collection phase of this archiving project, we are looking to develop a system that would solve this problem for future music researchers and collectors, part of which is to create an application which could be integrated into existing library catalogues so that tunes might be made musically searchable. The current paper aims to establish the feasibility of such a programme, and describes the prototype system we have built. In the following sections, we introduce Lakeland Hunting Songs in more detail and by example, we review the literature on music search, and then describe the archive-exploration system we have built and its quantitative and holistic evaluation.

2 LAKELAND HUNTING SONGS

Following its inclusion in The National Song Book [24], “John Peel” – the most famous Lakeland Hunting Song – was once sung in homes and schools across England, and yet this singing tradition has not been the subject of much academic enquiry. Emmett [13] sought to address this gap in folk music and ethnomusicological scholarship through a study of the history, texts, and tunes of Lakeland Hunting

¹Other work does cover Lakeland Hunting Songs in some detail, but they are not the sole focus [1]. Other publications that include some information about the tradition include [22], [21] and [20].

Table 1: Definitions and numbers of instances of Song, Tune, and Item in the corpus. The terms are consistent with [13].

Label	Number	Definition
Song	313	A text which is conceived with the intention of being sung; most often this refers to a combination of a text and a tune.
Tune	147	The music used for a particular song, which may or may not have a known tune name.
Item	80	Tunes from the corpus used in the current archive-exploration project.

Songs. The singing tradition remains very much alive and, over a period of six years, 313 songs were collected from a variety of sources, many from live recordings made at so-called “sing-songs”.

Since the 2005 Hunting With Dogs Act, fox hunting is now illegal, but Lakeland Hunting is very different from common perceptions of the “sport” and is often described as “hunting the hard way”. The Cumbrian terrain is mountainous and craggy, so the hunts are not mounted (i.e. followed on horseback) – those following the hunt do so on foot. Lakeland Hunting is also not generally considered to be an elitist activity and emerged as a form of pest control as a service for local farmers [23].

Songs have always been an integral part of the Lakeland Hunting tradition. These poorly-funded hunts [4] were, and continue to be, reliant on the fundraising activities of the social side of the hunt, of which singing makes up a significant part.

The songs themselves are rather eclectic in nature and, it would seem, almost any song could be termed a “Lakeland Hunting Song” if it were performed at a hunt sing-song. However, the majority of the songs are concerned with “Lakeland” and/or “Hunting” in some way, or they contain some sort of countryside connection [13]. Though the songs were once often accompanied (by piano, melodeon, or any other instruments that happened to be at hand), this is no longer the case and performances are now generally unaccompanied. Furthermore, the tunes used by these songs are not unique to this tradition, most often deriving from national hunting songs or popular songs of the 20th century.

2.1 Music Examples

Of the 147 tunes in this entire collection of Lakeland Hunting Songs, 140 are in major keys with the remaining seven (for eight songs) in modal keys. We used 80 items (Table 1) in the present work because they were readily available in MusicXML format. The transcriptions tend to represent the most common version heard of the many “borrowed” tunes used by this tradition, though common variants are sometimes included on ossia staves, as can be seen in the tune “Hark Forrad” (Figure 1). This is the most commonly used tune within the tradition, with 34 songs likely employing it. Part of what makes this tune so popular and useful is it can be used in a recitative style, as rhythms can be freely altered to fit any given underlay.

A “typical” version of this tune is given in Figure 2. “Hark Forrad” is arguably the most complicated example of a tune in this collection as it can be rendered in so many different ways. It was therefore important for the search system (see Section 3) to be able

**Figure 1: General structure of “Hark Forrad”. (This and all other original transcription by Mary Emmett.)**

to return “Hark Forrad” as a result if queries contained common variants. Some examples of such common variant queries are given in Figure 3.

**Figure 2: Typical version of “Hark Forrad” – without ossia staff variations.****Figure 3: (A) Example query using option (a) from the ossia staff seen in Figure 1; (B) Example query using option (c) from the ossia staff seen in Figure 1; (C) Example query of an altered rhythm based on actual underlay from a song. This corresponds with bars 3-4 of Figures 1 and 2.**

As mentioned previously, most of the tunes used in this tradition are borrowed from popular culture. A good example of this is “Forty Shades of Green” (Figure 4), perhaps best known as a song by Johnny Cash. This tune was intended by the lyricist to be used for the song “Our Johnny”, but a local composer then wrote an original tune as an alternative setting for the song. The resulting, most commonly used tune is a mixture of the two. With regards to the search system, it will be interesting to see if entering a query

from “Forty Shades of Green” brings up the tune which is now used when performing “Our Johnny” (Figure 5).



Figure 4: Transcription of “Forty Shades of Green” as performed at a Lakeland Hunting sing-song in 2015.



Figure 5: Transcription of the tune for “Our Johnny”.

Finally, there are some tunes with slightly awkward rhythms, or where the time signature changes partway through. An example of this is “First of May”, which would, in the original transcription, appear as in Figure 6A. The rhythm in this tune should be swung, so the tune actually sounds as in Figure 6B with all the triplet rhythms written out. Neither music notation software nor a music search/exploration system would “know” to swing the rhythm without being programmed to interpret such textual instructions, but it was our hope that the system we created would be robust enough to deal with varied rhythmic inputs of this tune, whether swung or not.



Figure 6: (A) Extract from the original transcription of “First of May”; (B) “First of May” with swung rhythm fully notated.

3 MUSIC SEARCH

Music search has been a topic of sustained and diverse work over the decades, one of the reasons being that there are many different expectations users have of a music search system, and also many different ways a system might capture and represent queries, as well as mechanisms for retrieving and then representing results [2, 3, 14, 26–28, 30].

Even in our single archive-exploration system, as we aim to support a range of expertise from everyday users to professional archivists, we see potential for many different forms of query capture. For instance, an everyday user might prefer to hum a query and not want to “tidy it up” before hitting search, whereas an archivist might be exploring relationships between songs and want to be sure their queries are entered accurately.

One of the earliest, pre-digital computing examples we know of music search is Barlow and Morgenstern’s *A Dictionary of Musical Themes* [3]. As well as containing 10,000 tune excerpts from composers in alphabetic order, it finishes with an index via which tunes can be queried based on pitch-class sequences. In terms of the lookup mechanism, it is similar to modern-day music search systems such as FolkTuneFinder [28] and Musipedia [25].

The pitch or pitch class-based approach to lookup is powerful because it is robust to rhythmic variations (essentially, it ignores rhythm), and in both [3, 28] there are methods to ensure transposition invariance. But a drawback is that it is not robust to variations in the form of interpolated, missing, or varied notes. For instance, if we used the opening six notes of the general form of “Hark Forrad” (D4, G4, A4, G4, A4, B4, Fig. 1) to query a system that contained the typical version of the tune (beginning D4, D4, G4, G4, G4, . . . , Fig. 2), a pitch-sequential approach to lookup would be unlikely to identify the query as “Hark Forrad”, because the sequences are not very similar to one another due to the interpolated notes in the typical version compared to the general version. Even Parson’s coding of directions between adjacent notes, as used in several modern-day music search systems (e.g., [17]), is vulnerable to interpolated, missing, or varied notes. This is a drawback because from music-theoretic and music-perceptual points of view, the two excerpts of music from the openings of Figures 1 and 2 are similar to one another in terms of their use of pitches at corresponding rhythmic locations. One might say their “musical skeletons” are the same.

Computational methods that are capable of representing such similarity include derived viewpoints [10], geometric representations [8, 15, 26], and geometric hashing in particular [2, 6, 27]. Geometric hashing entails analysing and storing pitch and time comparisons between pairs or triples of notes that are appropriately local to one another (local in pitch and time). The advantage of this approach is that when the musical skeleton of a query is highly similar to something in the dataset being queried, the match can still be identified and returned, irrespective of a certain amount of interpolation, removal, or variation of notes between the two excerpts. In what follows, we explore the use of geometric hashing to build our archive-exploration system, because: (1) it is novel in this context, and to our knowledge is the first instance of geometric hashing being used for music-archival search/exploration; (2) it has the potential to support most of the use cases stated or inferred from Figures 1-6 – a claim that we return to assess in the evaluation.

4 DESIGN AND IMPLEMENTATION OF OUR ARCHIVE-EXPLORATION SYSTEM

An important property of any search system is that it returns relevant results quickly. Similar to previous research, we use a customised hash table to store geometric comparisons between pairs of

notes that are appropriately local to one another in pitch and time [2, 6, 27]. A hash table is a data structure that provides constant-time insertion and finding of an entry [29]. Remarkably, the larger one’s collection of entries becomes, the time required to insert or find an entry in that collection remains fixed and very short. We use JavaScript/Node.js to implement our system [7], where the regular JavaScript object is designed to support insert and find in this way.² The method relies on creating (often-inscrutable) *hashes* of entries, which are large numbers that should uniquely identify an entry’s location in the hash table. Our customisation relates to the information stored when the same entry is encountered twice or more, which occurs when the same pitch and time difference are encountered multiple times across an input music dataset. As such, we store the piece id and the time in the piece where the pitch and time difference was encountered.

4.1 Hash Construction

The concept of making pitch and time difference comparisons between pairs and triples of notes goes back to [18]. Its being done in a tractable way (as the potential comparisons grow exponentially with the number of notes in a piece) and on an industrial scale for audio input are due to [27]. The idea of moving back to a symbolic representation instead of peaks in a spectrogram before making the comparisons is due to [2], where improvements are also achieved in relation to inexact matching (e.g., transposition and tempo invariance) that hark back to [18]. The current work remains with or maps down to a symbolic representation in order to construct hashes, incorporating transposition invariance but leaving tempo invariance to one side for now, although it would be straightforward to extend to tempo invariance in future.

As well as constructing hashes for any input query on-the-fly, hashes are constructed for each piece in a dataset in advance. The MusicXML representation of each of our 80 songs is converted to a set of (ontime, MIDI note number)-pairs.³ Suppose the sets A and B each contain two points:

$$A = \{(x_1^A, y_1^A), (x_2^A, y_2^A)\} \quad (1)$$

$$B = \{(x_1^B, y_1^B), (x_2^B, y_2^B)\} \quad (2)$$

where points are ordered increasing by ontime, and by MIDI note number (MNN) if they have the same ontime, so x_1^A is the ontime of the first note in A and y_1^A the MNN. We aim to quantify the similarity between the two sets mathematically and perceptually. The similarity is calculated considering the ontime and pitch dimensions separately. Starting with the pitch dimension, we can obtain the pitch difference, denoted as pd for each set:

$$pd^A = x_2^A - x_1^A \quad (3)$$

$$pd^B = x_2^B - x_1^B \quad (4)$$

where closer values of pd^A and pd^B indicate higher similarity between A and B in the pitch dimension. We also calculate the interval

between ontime values, denoted:

$$td^A = y_2^A - y_1^A \quad (5)$$

$$td^B = y_2^B - y_1^B \quad (6)$$

where, again, closer values of td^A and td^B indicate higher similarity between A and B in the ontime dimension. We then encode pd and td into strings for valid hash entries found in a set. This string consists of two parts:

- (1) a + or – sign followed by two digits, indicating the pitch difference pd ;
- (2) a float number rounded to one decimal place, indicating the time difference td .

For example, the string for the last two notes of Figure 6A would be “-020.5”, because the interval from E4 to D4 is –2 MNNs, and the time difference between the E4 and D4 is 0.5 crotchet beats.

4.2 Storage and Retrieval

Here we describe how to restrict attention to pairs of notes (or points) that are local to one another in pitch and time. With the collected MusicXML files, we process them individually into a set of (ontime, pitch)-pairs. Then we apply the following process:

```

pts ← the set of points
npts ← pts.length
tMin ← 0.1; tMax ← 10; pMin ← 1; pMax ← 12
entries ← []
for (i = 0; i < npts - 1; i++) do
  v0 ← pts[i]
  j ← i + 1
  while (j < npts) do
    v1 ← pts[j]
    apd ← abs(v1.pitch - v0.pitch)
    td ← v1.ontime - v0.ontime
    if ((td > tMin) ∧ (td < tMax) ∧ (apd ≥ pMin) ∧ (apd ≤ pMax)) then
      entry ← create_entry(v0, v1)
      entries.insert(entry)
    end if
    if (td ≥ tMax) then
      j ← npts - 1
    end if
    j++
  end while
end for

```

So the process consists of a nested loop to confirm a valid set of two points, which is then encoded as the string of a hash entry with the aforementioned format. Starting with the outer loop where we have the first point v_0 , the inner loop checks through the points following v_0 , during which the second point v_1 can be confirmed once a point satisfies the conditions: pd is in the range between $pMin$ (minimum pitch difference, e.g., 1) and $pMax$ (maximum pitch difference, e.g., 12); td is in the range (not equal to) between $tMin$ (minimum time difference, e.g., 0.1) and $tMax$ (maximum time difference, e.g., 10). If the conditions fail, particularly when time difference exceeds the maximum, the loop breaks. The encoded hash entry string then

²<https://v8.dev/>

³Ontime is the start time of a note counting in crotchet beats, with 0 for bar 1 beat 1 [5].

Table 2: Success rate and execution times of the archive-exploration system for opening and chorus queries. Execution times are reported for a MacBook Pro running Node.js on OS X 10.14 with a 2.4 GHz processor and 16 GB RAM.

Query Type	Top 10	Top 3	Mean (std) time (ms)
Opening	86% = 69/80	53% = 42/80	10.9 (4.3)
Chorus	90% = 9/10	40% = 4/10	16.9 (4.3)

becomes the key of a (key, value) pair in a JavaScript object. The value part of this pair consists of a pair of parallel arrays, storing: (1) where in terms of ontime the pair of notes A was encountered (uses x_1^A) in the first array; (2) the song name in the corresponding location of a second array. This leads to a JavaScript object, call it lookup, which we store as a JSON file on the server.⁴

With an incoming query, we take the point-set representation and apply the same method as above to obtain valid hashes and construct (key, value) pairs. Each query key (formed from B and its analysis in Equations 1-6, say) is used to probe lookup, and may match to some key (formed from A and its analysis in Equations 1-6), returning a point set $C = \{(a_1, b_1), (a_2, b_2), \dots, (a_M, b_M)\}$ of (dataset ontime, query ontime)-pairs where there are matching pitch and time differences. In a scatterplot of such data, true positives appear as (approximately) diagonal lines. In order to analyse this data to obtain the top m matches, therefore, it is necessary to apply a transformation (e.g., $(a_i, b_i) \rightarrow (a_i, a_i - b_i)$) and calculate a histogram over this transformed data. If the query itself led to the formation of k valid hashes, then we can calculate the relevance of the match by dividing the observed frequency h in a histogram bin by k . That said, sometimes due to choosing a large bin size or having polyphonic data where there can be multiple matches simultaneously (not an issue for our melodies here), h exceeds k . To address this, we map any relevance values in excess of 1 back down to 1 before returning search results to the user, but this can create a “ceiling effect” with relevance scores, which requires more attention in future.

5 EVALUATION

As is standard when assessing music search systems [e.g., 2], we begin the evaluation of our system with some statistics on its success rate for finding certain types of query and the execution taken to do so, contained in Table 2.

The “Opening” query type row of the table shows the relative success or failure (and execution time) of queries consisting of the first six notes of each of the 80 items (“opening query” hereafter). If we consider success to be when the item from which the opening query is taken appears among the top ten returned results, then our system has a success rate of 86% (= 69 successes / 80 items); if we consider success to be limited to the top three returned results only, then the system has success rate 53 percent (= 42 successes / 80 items).

In terms of execution time, results for the 80 opening queries were returned in an average 10.9 ms (standard deviation 4.3 ms),

⁴Approximately 2 MB in size for 80 songs, but this could be made considerably smaller by utilising shorter IDs for song names.


substantiating our claim that this is a quick method. In a client-server setup, the main bottleneck is not the querying itself but the retrieving and excerpting of score snippets to return to the client.

The “Chorus” query type row of the table shows corresponding information for queries consisting of the first six notes of the chorus of each of ten items (“chorus query” hereafter). The first author annotated these locations for ten of the most prominent Hunting Songs, because (1) choruses are as, if not more, important than openings when considering the identity of these songs and how they are discussed among the hunting community, and (2) so that we could give some indication of the generalisability of the results reported for opening queries.⁵ If we consider success to be when the item from which the chorus query is taken appears among the top ten returned results, then our system has a success rate of 90% (= 9 successes / 10 items); if we limit to the top three returned results, then the system has success rate 40 percent (= 4 successes / 10 items). Execution times are similarly quick – there should not and does not appear to be a substantial difference in execution time when a query occurs partway through rather than at the start of a dataset song.

We will comment on the success rates and cases of failure further in the discussion, but for now suffice it to say that we are satisfied with this level of performance of our archival-exploration system for short, “exact” queries. This analysis says nothing, however, of the system’s ability to return relevant results when queried with tune variants, so we go on to report on case-by-case results for such scenarios.

One of the main things we wanted this system to be able to cope with was common tune variants in both pitch and rhythm, and we can report that this has been successful in many cases. When we take the “Hark Forrad” tune as an example, Figures 7, 8 and 9 show the query-result pairs for the examples laid out in Figures 3A, B, and C, respectively. In the first two examples it is the pitch which is altered and yet “Hark Forrad” comes back as the first return in both instances. In the third example, when there are rhythmic alterations, the correct tune is returned second, which the first author suggests is still an acceptable degree of accuracy.

Note query



#1 | Begins bar = 4 | relevance = 1

Hark Forrad - General Version





Figure 7: Results when testing the system with the example query from Figure 3A.

When attempting to test the system with a query taken from “Forty Shades of Green” (see bars 12-14 of Figure 4) we encountered

⁵We would probably extend these annotations to the entire collection in future.

Note query



#1 | Begins bar = 6 | relevance = 1

Hark Forrad - General Version






Figure 8: Results when testing the system with the example query from Figure 3B.

Note query



#1 | Begins bar = 19 | relevance = 1

Lass doon on the quay



#2 | Begins bar = 3 | relevance = 1

Hark Forrad - General Version




Figure 9: Results when testing the system with the example query from Figure 3C.

a few problems. We used this query because it is the most similar to “Our Johnny” (see bars 6-7 of Figure 5). When inputting the query, it was not possible to dot the second C \sharp (see Figure 10), but when pressing search we were pleased to see that “Our Johnny” came up as the fifth returned result. “Forty Shades of Green” did not appear in the top ten results, however, despite the only difference from the original transcription being a missing dot.

Note query



Figure 10: Initial example query from towards ‘Forty Shades of Green’.

The problem with inputting the desired query is due to the over-simplicity of our system’s note input method, which we will

return to in Limitations and Future Work (Section 6.1): dotting the second C \sharp in Figure 10 would cause an overflow of the (default and currently uneditable) 4-beat bar with no upbeat, and so the stave notation rendering issues an error message. The system is not yet sophisticated enough to use tied notes to deal with seemingly “too many” beats being added to a bar.

A temporary workaround is to insert a 3-beat rest at the start of the query; then dotting the second C \sharp does not overflow the bar (Figure 11). Then “Forty Shades of Green” is the first result (again, see Figure 11), but now “Our Johnny” does not appear in the top ten. This is perhaps not that surprising as while to a human listener the two tunes are related, there are likely enough pitch and rhythmic differences between the two for “Our Johnny” to not be returned within the top ten results.

Note query



#1 | Begins bar = 3 | relevance = 1

Forty Shades of Green



Figure 11: Example query from “Forty Shades of Green” with the anacrusis included.

Inputting the straight rhythm for “First of May” (see Figure 12) returned the intended tune as the first result, but finding a way to input the swung rhythm without being able to sing it was a challenge. Some further development work by the third author to extend the query syntax made it possible to input triplets to a degree, but more work is required to be able to mix crotchet and quaver triplets within the same beat. Nonetheless, we were able to get the start times and pitches in the correct locations (Figure 13), and even though the durations are not as they would be in a performance, note durations themselves are not taken into account by the search system anyway. The triplet query did not return “First of May” within the top ten results. We will discuss this further in Section 6.1 on Limitations and Future Work.

6 DISCUSSION

Music search is a topic that has a long and varied history in terms of potential use cases, attracting the attention of and having ramifications for (ethno)musicologists, archivists, music informaticians, computer scientists, music psychologists and neuroscientists, musicians, as well as “everyday users” who want to hum a tune that is in their head and be shown “relevant answers”. In the current paper, we have focused on a subcategory of music search – the development of an archive-exploration system, whose range of potential users is still wide: initially song and music collectors might use such a system, but also archivists working within music, as this system would make it possible to search musically for items

Note query



#1 | Begins bar = 0 | relevance = 1

First of May



Figure 12: Results when testing the system with the first 12 notes of “First of May”.

Note query




Figure 13: Triplet query example for “First of May”.

within a library catalogue, should the sheet music be available in the appropriate digital format (e.g., MusicXML or similar).

Recent work [12, 14] indicates a trend towards search in the absence of “intermediary representations” such as MusicXML. For instance, a deep learning algorithm can accept an audio query and relate it directly to the appropriate section of a graphics (PDF) file of the score, without the need for a digital sheet music format. That said, there are more features one can offer to users with regards the display, navigation, and manipulation of search results if such intermediary representations are available. Rather than just being effective for a discrete song collection such as the Lakeland Hunting Songs used here, it is envisaged that eventually our work could be expanded to search for any musical item held within a library catalogue. The archive-exploration system detailed in this paper could be a stepping stone towards library users being able to simply ask a library catalogue for any item within its holdings via a note-based or audio query.

In this paper, we have outlined the current issues faced by ethnomusicologists when it comes to cataloguing their collections, as well as setting out the use cases we hoped our archive-exploration system would be able to address for the specific collection of Lakeland Hunting Songs. We then outlined the design and implementation of our archive-exploration system before detailing the ways in which it operates at the current point in time, highlighting positives and areas in which we aim to be able to improve it in the future. The archive-exploration system we have created within this project works well for its intended purpose. It has an 86% success rate for opening queries and a 90% success rate for chorus queries. Furthermore, results can be obtained quickly (10-20 ms) using the hash table representation.

In terms of music representations, we discussed how existing systems such as FolkTuneFinder [28] are robust to rhythmic changes

but not to missing or interpolated notes, so we employed a geometric hashing approach as the basis for determining matches and as a potential improvement on existing work. We found this approach to be advantageous in all but one use case, discussed below.

The archive-exploration system developed in this pilot has brought several benefits to this project. Primarily, for the ethnomusicologist and archivist first author, this prototype demonstrates that it is possible to search musically within a database of tunes, and that these items can be returned quickly and, to a point, accurately.

An unexpected benefit was in highlighting which tunes were deemed to be similar in a computational approach, compared with those that might be grouped together by a human listener. This is an interesting development which the first author will take forward into her future work, but this may also be useful in some way to those working specifically on computational and/or cognitive approaches to music similarity.

Finally, the search system has brought about greater interest in the project from external parties (both archivists and folk music specialists), as there is now a physical interface with which the ideas in this paper can be demonstrated and explored.

6.1 Limitations and Future Work

The “First of May” triplet query example (see Figure 13) failed to return this song (Figure 6(A)) in the top ten results, because in terms of rhythmic spacing, the straight-to-triplet alterations disrupt what we described as the “musical skeleton” of the tune in Section 3 (p. 3). A solution to this limitation would be to reduce the level of temporal similarity required for a match. The reduction could correct this result in future, but this same change would likely lead to an increase in false-positive results in other scenarios. It is a clear example of where the pitch-sequential approach of systems such as [3, 28] is superior to the geometric hashing approach explored here, because the pitches remain the same but the rhythms are quite apart. That said, this is the only use case identified in Section 2.1 where geometric hashing fails, and so overall for this application of music search to an archive-exploration system, we would suggest geometric hashing is at least equivalent if not superior to the pitch-sequential approach. Further work and more rigorous evaluation is required to settle this matter definitively, and probably a hybrid system that utilises both approaches would be preferable overall.

There were some deficiencies in the note input and subsequent stave notation rendering that require addressing. First, when a user causes the number of beats in a bar to be exceeded by the durations of notes and rests therein, the system ought to break notes with ties across barlines automatically, but this is not the case at present. Second, while we developed a syntax to represent triplets, this assumes that those triplets are all of the same value (e.g., three quaver-note triplets) rather than mixed (e.g., one crotchet-note triplet followed by one quaver-note triplet). Third, the stave that displays the user’s query assumes a 4-4 time signature and C-major key signature. Syntactical means of specifying these will be introduced. Deficiencies in these three stave notation rendering components will be addressed in future work.

Distinctiveness [6, 9] is a topic that has come up multiple times in discussion of the above results. We are still investigating cases of failure for the results given in Table 2, and some may be due to

an anacrusis bug, but in other cases, not retrieving the intended song within the top three or ten results comes down to the distinctiveness (or lack thereof) of the query and/or material in each song. As the first author will go on to explore in future work, there is much overlap between melodic motifs in these songs, so it is to be expected that for a given opening query, there might be three or even ten moments from other songs that match equally well, and so the target result may be further down the list. A wealth of literature from music psychology, especially with regards music cognition memory [e.g., 11, 19], could be brought to bear here, acknowledging and deciding if/how to model within our future research phenomena such as humans being biased in various ways in terms of music memory formation (e.g., towards the beginnings of songs or choruses), whereas a computational search and exploration system such as that developed here is “more democratic”.

ACKNOWLEDGMENTS

This work was supported by a Jane Moody Scholarship from the Humanities Research Centre, University of York to the first author.

REFERENCES

- [1] Sue Allan. 2017. *Folk Song in Cumbria: A Distinctive Regional Repertoire?* Ph.D. Dissertation. Lancaster University, Lancaster, UK.
- [2] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. 2012. Fast Identification of Piece and Score Position via Symbolic Fingerprinting. In *ISMR*. 433–438.
- [3] Harold Barlow and Sam Morgenstern. 1948. *A dictionary of musical themes*. Crown.
- [4] Ron Black and Wendy Fraser. 2013. *Hunting Songs Volume Two: Lakeland Songs*. Self-published.
- [5] Tom Collins. 2011. *Improved methods for pattern discovery in music, with applications in automated stylistic composition*. Ph.D. Dissertation. The Open University.
- [6] Tom Collins, Andreas Arzt, Harald Frostel, and Gerhard Widmer. 2016. Using geometric symbolic fingerprinting to discover distinctive patterns in polyphonic music corpora. In *Computational Music Analysis*. Springer, 445–474.
- [7] Tom Collins and Christian Coulon. 2019. MAIA Util: An NPM Package for Bridging Web Audio with Music-theoretic Concepts. In *Proceedings of the Web Audio Conference*. Trondheim, Norway, 47–52.
- [8] Tom Collins, Jeremy Thurlow, Robin Laney, Alistair Willis, and Paul Garthwaite. 2010. A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works. In *Proceedings of the International Society for Music Information Retrieval Conference*. Utrecht, The Netherlands, 3–8.
- [9] Darrell Conklin. 2010. Discovery of distinctive patterns in music. *Intelligent Data Analysis* 14, 5 (2010), 547–554.
- [10] Darrell Conklin and Ian H Witten. 1995. Multiple viewpoint systems for music prediction. *Journal of New Music Research* 24, 1 (1995), 51–73.
- [11] Diana Deutsch. 1980. The processing of structured and unstructured tonal sequences. *Perception & psychophysics* 28, 5 (1980), 381–389.
- [12] Matthias Dorfer, Jan Hajič Jr, Andreas Arzt, Harald Frostel, and Gerhard Widmer. 2018. Learning audio-sheet music correspondences for cross-modal retrieval and piece identification. *Transactions of the International Society for Music Information Retrieval* 1, 1 (2018).
- [13] Mary Emmett. 2021. *The Hunting Songs and Singing Tradition of the Cumbrian Lakeland Fell Packs*. Ph.D. Dissertation. University of York, York, UK.
- [14] Christian Frank. 2020. The Machine Learning Behind Hum to Search. Retrieved April 11, 2021 from <https://ai.googleblog.com/2020/11/the-machine-learning-behind-hum-to.html>
- [15] David Garfinkle, Claire Arthur, Peter Schubert, Julie Cumming, and Ichiro Fujinaga. 2017. PatternFinder: Content-Based Music Retrieval with Music21. In *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*. 5–8.
- [16] Berit Dorle Janssen. 2018. Retained or lost in transmission?: Analyzing and predicting stability in Dutch folk songs. (2018).
- [17] Andreas Kornstädt. 1998. Themefinder: A web-based melodic search tool. *Computing in musicology: a directory of research* 11 (1998), 231–236.
- [18] David Lewin. 1987. *Generalized interval systems and transformations*. Yale University Press.
- [19] Daniel Müllensiefen and Andrea R Halpern. 2012. The role of features and context in recognition of novel melodies. *Music Perception: An Interdisciplinary Journal* 31, 5 (2012), 418–435.
- [20] Lyn Murfin. 1990. *Popular Leisure in the Lake Counties*. Manchester University Press.
- [21] Steve Roud. 2017. *Folk song in England*. Faber & Faber.
- [22] Ian Russell. 2002. The hunt’s up? Rural community, song, and politics. *Acta Ethnographica Hungarica* 47, 1-2 (2002), 127–141.
- [23] Ian Russell. 2003. The singer’s the thing: The individual and group identity in a Pennine singing tradition. *Folk Music Journal* (2003), 266–281.
- [24] Charles Villiers Stanford (Ed.). 1906. *The National Song Book: A Complete Collection of the Folk-songs, Carols, and Rounds suggested by the Board of Education (1905)*. Boosey & Co., London, UK.
- [25] Rainer Typke. 2007. *Music retrieval based on melodic similarity*. Ph.D. Dissertation. Utrecht University.
- [26] Esko Ukkonen, Kjell Lemström, and Veli Mäkinen. 2003. Geometric algorithms for transposition invariant content-based music retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference*. 193–199.
- [27] Avery Li-Chun Wang and Julius O. Smith III. 2012. System and methods for recognizing sound and music signals in high noise and distortion. Patent US 8,190,435 B2. Continuation of provisional application from 2000.
- [28] Joe Wass. 2008. FolkTuneFinder. Retrieved April 11, 2021 from <https://www.folktunefinder.com/>
- [29] Mark Allen Weiss. 2010. *Data structures and problem solving using Java* (4 ed.). Pearson Education, Inc.
- [30] Zongyu Yin, Federico Reuben, Susan Stepney, and Tom Collins. 2021. “A Good Algorithm Does Not Steal—It Imitates”: The Originality Report as a Means of Measuring When a Music Generation Algorithm Copies Too Much. In *Artificial Intelligence in Music, Sound, Art and Design: EvoMUSART*. Springer International Publishing, 360–375.