Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

The Intervalgram: An Audio Feature for Large-scale Cover-song Recognition

Thomas C. Walters, David A. Ross, and Richard F. Lyon

Google, Brandschenkestrasse 110, 8002 Zurich, Switzerland
tomwalters@google.com

Abstract. We present a system for representing the musical content of short pieces of audio using a novel chroma-based representation known as the 'intervalgram', which is a summary of the local pattern of musical intervals in a segment of music. The intervalgram is based on a chroma representation derived from the temporal profile of the stabilized auditory image [10] and is made locally pitch invariant by means of a 'soft' pitch transposition to a local reference. Intervalgrams are generated for a piece of music using multiple overlapping windows. These sets of intervalgrams are used as the basis of a system for detection of identical melodic and harmonic progressions in a database of music. Using a dynamic-programming approach for comparisons between a reference and the song database, performance is evaluated on the 'covers80' dataset [4]. A first test of an intervalgram-based system on this dataset yields a precision at top-1 of 53.8%, with an ROC curve that shows very high precision up to moderate recall, suggesting that the intervalgram is adept at identifying the easier-to-match cover songs in the dataset with high robustness. The intervalgram is designed to support locality-sensitive hashing, such that an index lookup from each single intervalgram feature has a moderate probability of retrieving a match, with few false matches. With this indexing approach, a large reference database can be quickly pruned before more detailed matching, as in previous content-identification systems.

Keywords: Cover Song Recognition, Auditory Image Model, Machine Hearing

1 Introduction

We are interested in solving the problem of cover song detection at very large scale. In particular, given a piece of audio, we wish to identify another piece of audio representing the same underlying composition, from a potentially very large reference set. Though our approach aims at the large-scale problem, the representation developed is compared in this paper on a small-scale problem for which other results are available.

There can be many differences between performances with identical melodies. The performer may sing or play at a different speed, in a different key or on a different instrument. However, these changes in performance do not, in general, prevent a human from identifying the same melody, or pattern of notes. Thus,

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

given a performance of a piece of music, we wish to find a representation that is to the largest extent possible invariant to such changes in instrumentation, key, and tempo.

Serra [12] gives a thorough overview of the existing work in the field of melody identification, and breaks down the problem of creating a system for identifying versions of a musical composition into a number of discrete steps. To go from audio signals for pieces of music to a similarity measure, the proposed process is:

– Feature extraction
– Key invariance (invariance to transposition)
– Tempo invariance (invariance to a faster or slower performance)
– Structure invariance (invariance to changes in long-term structure of a piece of music)
– Similarity computation

In this study, we concentrate on the first three of these steps: the extraction of an audio feature for a signal, the problem of invariance to pitch shift (both locally and globally) and the problem of invariance to changes in tempo between performances of a piece of music. For the first stage, we present a system for generating a pitch representation from an audio signal, using the stabilized auditory image (SAI) [10] as an alternative to standard spectrogram-based approaches. Key invariance is achieved locally (per feature), rather than globally (per song). Individual intervalgrams are key normalized relative to a reference chroma vector, but no guarantees are made that the reference chroma vector will be identical across consecutive features. This local pitch invariance allows for a feature that can track poor-quality performances in which, for example, a singer changes key gradually over the course of a song. It also allows the feature to be calculated in a streaming fashion, without having to wait to process all the audio for a song before making a decision on transposition. Other approaches to this problem have included shift-invariant transforms [9], the use of all possible transpositions [5] or finding the best transposition as a function of time in a symbolic system [13]. Finally, tempo invariance is achieved by the use of variable-length time bins to summarize both local and longer-term structure. This approach is in contrast to other systems [5, 9] which use explicit beat tracking to achieve tempo invariance.

While the features are designed for use in a large-scale retrieval system when coupled with a hashing technique [1], in this study we test the baseline performance of the features by using a Euclidean distance measure. A dynamic-programming alignment is performed to find the smallest-cost path through the map of distances between a probe song and a reference song; partial costs, averaged over good paths of reasonable duration, are used to compute a similarity score for a each probe-reference pair.

We evaluate performance of the intervalgam (using both SAI-based chroma and spectrogram-based chroma) using the 'covers80' dataset [4]. This is a set of 160 songs, in 80 pairs that share an underlying composition. There is no explicit notion of a 'cover' versus an 'original' in this set, just an 'A' version and

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

a 'B' version of a given composition, randomly selected. While it is a small corpus, several researchers have made use of this dataset for development of audio features, and report results on it. Ellis [5] reports performance in terms of absolute classification accuracy for the LabRosa 2006 and 2007 music information retrieval evaluation exchange (MIREX) competition, and these results are extended by, amongst others, Ravuri and Ellis [11], who present detection error tradeoff curves for a number of systems.

Since we are ultimately interested in the use of the intervalgram in a large-scale system, it is worth briefly considering the requirements of such a system. In order to perform completely automated detection of cover songs from a large reference collection, it is necessary to tune a system to have extremely low false hit rate on each reference. For such a system, we are interested less in high absolute recall and more in finding the best possible recall given a very low threshold for false positives. Such systems have previously been reported for nearly-exact-match content identification [1]. The intervalgram has been developed for and tested with a similar large-scale back end based on indexing, but there is no large accessible data set on which performance can be reported. It is hard to estimate recall on such undocumented data sets, but the system identifies a large number of covers even when tuned for less than 1% false matches.

2 Algorithm

2.1 The Stabilized Auditory Image

The stabilized auditory image (SAI) is a correlogram-like representation of the output of an auditory filterbank. In this implementation, a 64-channel pole-zero filter cascade [8] is used. The output of the filterbank is half-wave rectified and a process of 'strobe detection' is carried out. In this process, large peaks in the waveform in each channel are identified. The original waveform is then cross-correlated with a sparsified version of itself which is zero everywhere apart from at the identified strobe points. This process of 'strobed temporal integration' [10, 14] is very similar to performing autocorrelation in each channel, but is considerably cheaper to compute due to the sparsity of points in the strobe signal. The upper panels of Figure 1 show a waveform (upper panel) and stabilized auditory image (middle panel) for a sung note. The pitch of the voice is visible as a series of vertical ridges at lags corresponding to multiples of the repetition period of the waveform, and the formant structure is visible in the pattern of horizontal resonances following each large pulse.

2.2 Chroma From the Auditory Image

To generate a chroma representation from the SAI, the 'temporal profile' is first computed by summing over the frequency dimension; this gives a single vector of values which correspond to the strength of temporally-repeating patterns in the waveform at different lags. The temporal profile gives a representation of

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

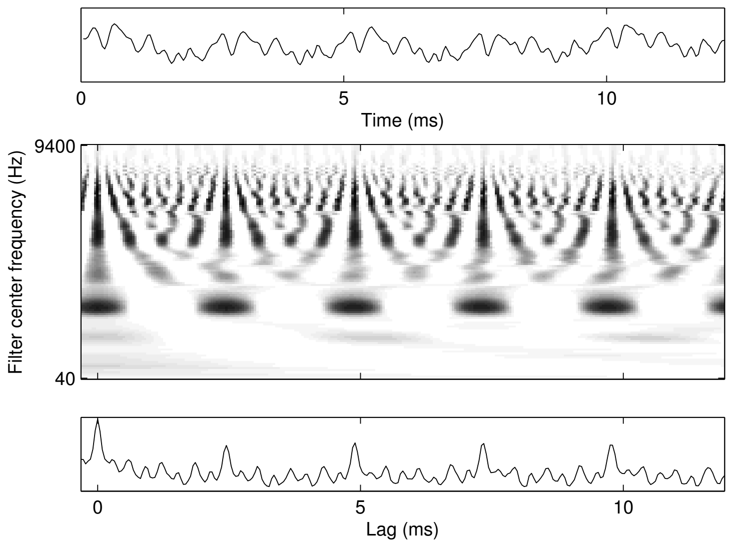

Fig. 1. Waveform (top panel), stabilized auditory image(SAI) (middle panel) and SAI temporal profile (bottom panel) for a human voice singing a note.

the time intervals associated with strong temporal repetition rates, or possible pitches, in the incoming waveform. This SAI temporal profile closely models human pitch perception [6]; for example, in the case of stimuli with a missing fundamental, there may be no energy in the spectrogram at the frequency of the pitch perceived by a human, but the temporal profile will show a peak at the time interval associated with the missing fundamental.

The lower panel of Figure 1 shows the temporal profile of the stabilized auditory image for a sung vowel. The pitch is visible as a set of strong peaks at lags corresponding to integer multiples of the pulse rate of the waveform. Figure 2 shows a series of temporal profiles stacked in time, a 'pitch-o-gram', for a piece of music with a strong singing voice in the foreground. The dark areas correspond to lags associated with strong repetition rates in the signal, and the evolving melody is visible as a sequence of horizontal stripes corresponding to notes; for example in the first second of the clip there are four strong notes, followed by a break of around 1 second during which there are some weaker note onsets.

The temporal profile is then processed to map lag values to pitch chromas in a set of discrete bins, to yield a representation as chroma vectors, also known as 'pitch class profiles' (PCPs) [12]. In our standard implementation, we use 32 pitch bins per octave. Having more bins than the standard 12 semitones in the Western scale allows the final feature to accurately track the pitch in recordings where

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

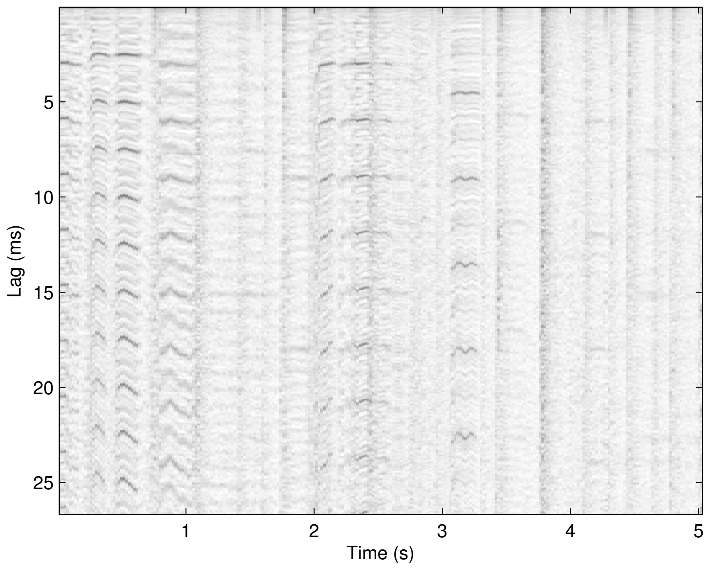

Fig. 2. A 'pitch-o-gram' created by stacking a number of SAI temporal profiles in time. The lag dimension of the auditory image is now on the vertical axis. Dark ridges are associated with strong repetition rates in the signal.

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

the performer is either mistuned or changes key gradually over the course of the performance; it also enables more accurate tracking of pitch sweeps, vibrato, and other non-quantized changes in pitch. Additionally, using an integer power of two for the dimensions of the final representation lends itself to easy use of a wavelet decomposition for hashing, which is discussed below. The chroma bin assignment is done using a weighting matrix, by which the temporal profile is multiplied to map individual samples from the lag dimension of the temporal profile into chroma bins. The weighting matrix is designed to map the linear time-interval axis to a wrapped logarithmic note pitch axis, and to provide a smooth transition between chroma bins. An example weighting matrix is shown in Figure 3. The chroma vectors for the same piece of music as in Figure 2 are shown in Figure 4.

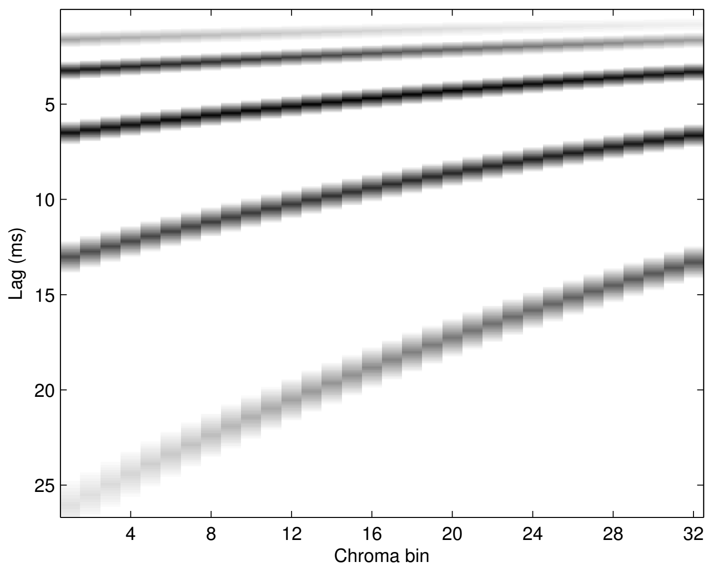

Fig. 3. Weighting matrix to map from the time-lag axis of the SAI to chroma bins.

2.3 Chroma From the Spectrogram

In addition to the SAI-based chroma representation described above, a more standard spectrogram-based chroma representation was tested as the basis for the intervalgram. In this case, chroma vectors were generated using the `chromagram_E` function distributed with the covers80 [4] dataset, with a modified step size to generate chroma vectors at the rate of 50 per second, and 32 pitch bins per

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

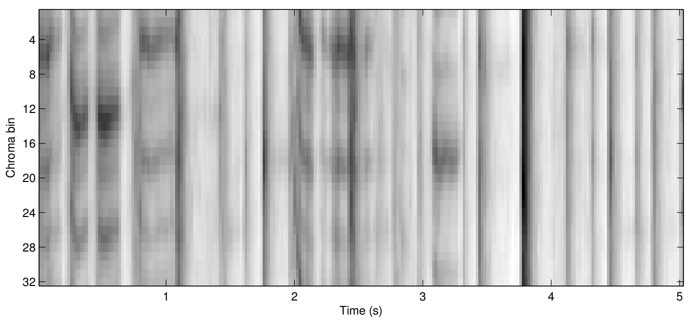

Fig. 4. Chroma vectors generated from the pitch-o-gram vectors shown in Figure 2.

octave for compatibility with the SAI-based features above. This function uses a Gaussian weighting function to map FFT bins to chroma, and weights the entire spectrum with a Gaussian weighting function to emphasize octaves in the middle of the range of musical pitches.

2.4 Intervalgram Generation

A stream of chroma vectors is generated at a rate of 50 per second. From this chromagram, a stream of 'intervalgrams' is constructed at the rate of around 4 per second. The intervalgram is a matrix with dimensions of chroma and time offset; however, depending on the exact design the time-offset axis may be nonlinear.

For each time-offset bin in the intervalgram, a sequence of individual chroma vectors are averaged together to summarize the chroma in some time window, before or after a central reference time. It takes several contiguous notes to effectively discern the structure of a melody, and for any given melody the stream of notes may be played a range of speeds. In order to take into account both short- and longer-term structure in the melody, a variable-length time-averaging process is used to provide a fine-grained view of the local structure, and simultaneously give a coarser view of longer timescales, to accommodate a moderate amount of tempo variation; that is, small absolute time offsets use narrow time bin widths, while larger absolute offsets use larger bin widths. Figure 5 shows how chroma vectors are averaged together to make the intervalgram. In the examples below, the widths of the bins increase from the center of the intervalgram, and are proportional to the sum of a forward and reverse exponential $w_b = f\left(w_f^p + w_f^{-p}\right)$, where p is an integer between 0 and 15 (for the positive bins) and between 0 and -15 (for the negative bins), f is the central bin width, and w_f is the width factor which determines the speed with which the bin width increases as a function of distance from the center of the intervalgram.

In the best-performing implementation, the temporal axis of the intervalgram is 32 bins wide and spans a total time window of around 30 seconds. The central

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

two slices along the time axis of the intervalgram are the average of 18 chroma vectors each (360ms each), moving away from the centre of the intervalgram, the outer temporal bins summarize longer time-scales before and after the central time. The number of chroma vectors averaged in each bin increases up to 99 (1.98s) in the outermost bins leading to a total temporal span of 26 seconds for each intervalgram.

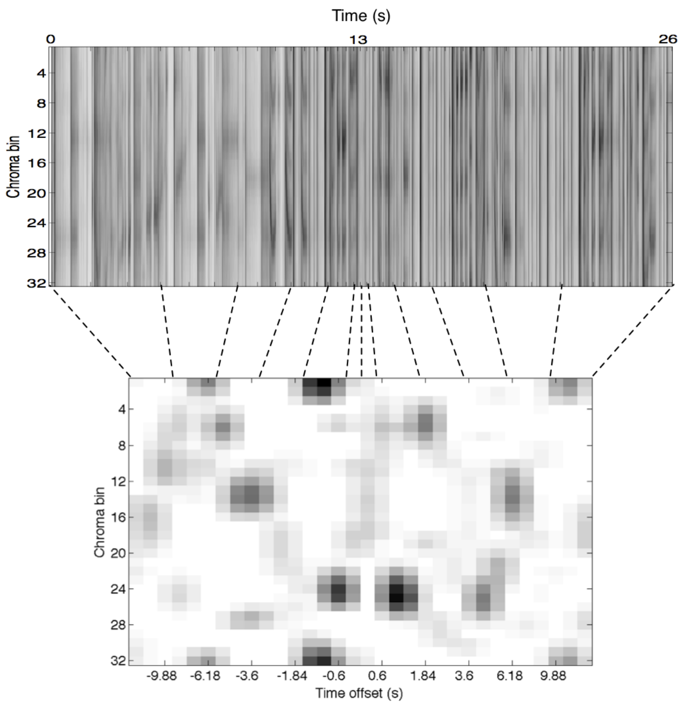

Fig. 5. The intervalgram is generated from the chromagram using variable-width time bins and cross-correlation with a reference chroma vector to normalize chroma within the individual intervalgram.

A 'reference' chroma vector is also generated from the stream of incoming chroma vectors at the same rate as the intervalgrams. The reference chroma vector is computed by averaging together nine adjacent chroma vectors using a triangular window. The temporal center of the reference chroma vector corresponds to the temporal center of the intervalgram. In order to achieve local pitch invariance, this reference vector is then circularly cross-correlated with each of the surrounding intervalgram bins. This cross-correlation process implements a 'soft' normalization of the surrounding chroma vectors to a prominent pitch or pitches in the reference chroma vector. Given a single pitch peak in the refer-

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

ence chroma vector, the process corresponds exactly to a simple transposition of all chroma vectors to be relative to the single pitch peak. In the case where there are multiple strong peaks in the reference chroma vector, the process corresponds to a simultaneous shifting to multiple reference pitches, followed by a weighted average based on the individual pitch strengths. This process leads to a blurry and more ambiguous interval representation but, crucially, never leads to a hard decision being made about the 'correct' pitch at any point. Making only 'soft' decisions at each stage means that there is less need for either heuristics or tuning of parameters in building the system. With standard parameters the intervalgram is a 32 by 32 pixel feature vector generated at the rate of one every 240ms and spanning a 26 second window. Since there are many overlapping intervalgrams generated, there are many different pitch reference slices used, some making crisp intervalgrams, and some making fuzzy intervalgrams.

2.5 Similarity Scoring

Dynamic programming is a standard approach for aligning two audio representations, and has been used for version identification by many authors (for example [16]; Serra [12] provides a representative list of example implementations). To compare sets of features from two recordings, each feature vector from the probe recording is compared to each feature vector from the reference recording, using some distance measure, for example Euclidean distance, correlation, or Hamming distance over a locality-sensitive hash of the feature. This comparison yields a distance matrix with samples from the probe on one axis and samples from the reference on the other. We then find a minimum-cost path through this matrix using a dynamic programming algorithm that is configured to allow jumping over poorly-matching pairs. Starting at the corner corresponding to the beginning of the two recordings the path can continue by jumping forward a certain number of pixels in both the horizontal and vertical dimensions. The total cost for any particular jump is a function of the similarity of the two samples to be jumped to, the cost of the jump direction and the cost of the jump distance. If two versions are exactly time-aligned, we would expect that the minimum-cost path through the distance matrix would be a straight line along the leading diagonal. Since we expect the probe and reference to be roughly aligned, the cost of a diagonal jump is set to be smaller than the cost of an off-diagonal jump.

The minimum and maximum allowed jump lengths in samples can be selected to allow the algorithm to find similar intervalgrams that are more sparsely distributed, interleaved with poorly matching ones, and to constrain the maximum and minimum deviation from the leading diagonal. Values that work well are a minimum jump of 3 and maximum of 4, with a cost factor equal to the longer of the jump dimensions (so a move of 3 steps in the reference and 4 in the probe costs as much as 4,4 even though it uses up less reference time, while jumps of 3,3 and 4,4 along the diagonal can be freely intermixed without affecting the score as long as enough good matching pairs are found to jump between). These lengths, along with the cost penalty for an off-diagonal jump and the difference

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

in cost for long jumps over short jumps, are parameters of the algorithm. Figure 6 shows a distance matrix for a probe and reference pair.

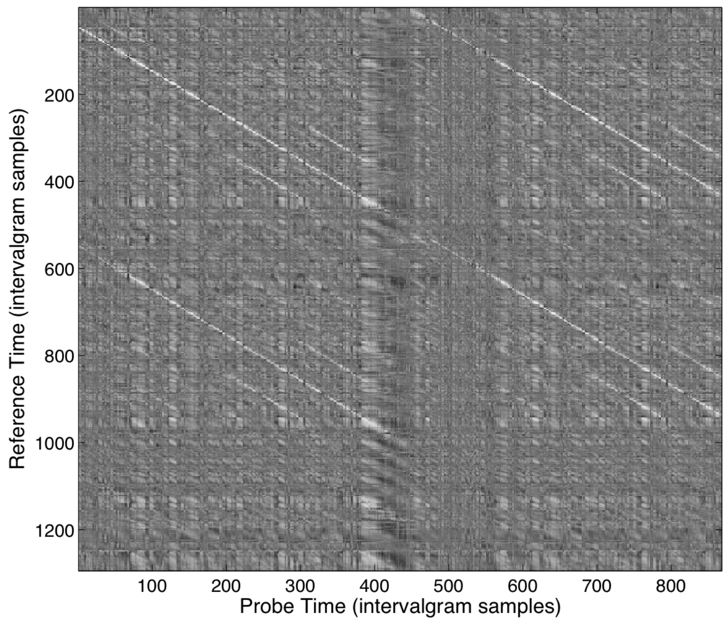

Fig. 6. Example distance matrix for a pair of songs which share an underlying composition. The lighter pixels show the regions where the intervalgrams match closely.

In the following section we test the performance of the raw intervalgrams, combined with the dynamic programming approach described above, in finding similarity between cover songs.

3 Experiments

3.1 Intervalgram Similarity

We tested performance of the similarity-scoring system based on the intervalgram, as described above, using the standard paradigm for the covers80 dataset, which is to compute a distance matrix for all query songs against all reference songs, and report the percentage of query songs for which the correct reference song has the highest similarity score.

Intervalgrams were computed from the SAI using the parameters outlined in Table 1, and scoring of probe-reference pairs was performed using the dynamic

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

programming approach described above. Figure 7 shows the matrix of scores for the comparison of each probe with all reference tracks. Darker pixels denote lower score, and lighter pixels denote higher scores. The white crosses show the highest-scoring reference for a given probe. 43 of the 80 probe tracks in the covers80 dataset were correctly matched to their associated reference track leading to a score of 53.8% on the dataset. For comparison, Ellis [5] reports a score of 42.5% for his MIREX2006 entry, and 67.5% for his MIREX2007 entry (the latter had the advantage of using covers80 as a development set, so is less directly comparable).

Parameter	Value
Chromagram step size (ms)	20
Chroma bins per octave	32
Total intervalgram width (s)	26.04
Intervalgram step size (ms)	240
Reference chroma vector width (chroma vectors)	4

Table 1. Parameters used for intervalgram computation.

In addition to the SAI-based chroma features, standard spectrogram-based chroma features were computed from all tracks in the 'covers80' dataset. These features used 32 chroma bins, and were computed at 50 frames per second, to provide a drop-in replacement for the SAI-based features. Intervalgrams were computed from these features using the parameters in Table 1.

In order to generate detection error tradeoff curves for the dataset, the scores matrix from Figure 7 was dynamically thresholded to determine the number of true and false positives for a given threshold level. The results were compared against the reference system supplied with the covers80 dataset, which is essentially the same as the system entered by LabRosa for the 2006 MIREX competition, as documented by Ellis [5]. Figure 8 shows ROC curves the Elllis MIREX'06 entry and for the intervalgram-based system, both with SAI chroma features and spectrogram chroma features. Re-plotting the ROC curve as a DET curve to compare results with Ravuri and Ellis [11], performance of the intervalgram-based features is seen to consistently lie between that of the LabRosa MIREX 2006 entry and their 2007 entry.

Of particular interest is the performance of the features at high precision. The SAI-based intervalgram can achieve 47.5% recall at 99% precision, whereas the Ellis MIREX '06 system achieves 35% recall at 99% precision. These early results suggest that the intervalgram shows good robustness to interference. The intervalgram also stands up well to testing on larger, internal, datasets in combination with hashing techniques, as discussed below.

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

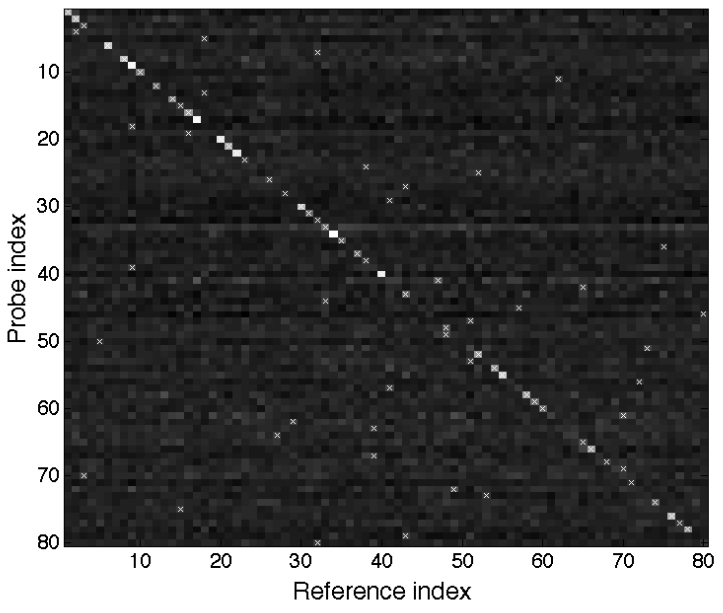

Fig. 7. Scores matrix for comparing all probes and references in the 'covers80' dataset. Lighter pixels denote higher scores, indicating a more likely match. White crosses denote the best-matching reference for each probe.

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

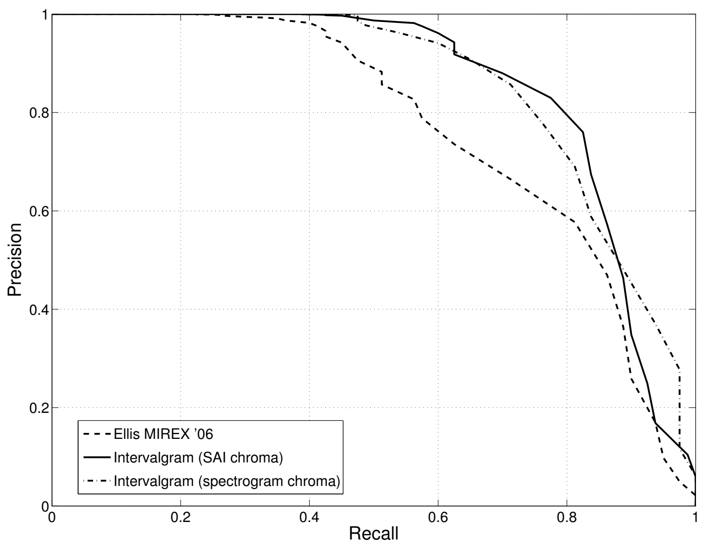

Fig. 8. ROC curves for the intervalgram-based system described in this paper and the LabROSA MIREX 2006 entry [5].

3.2 Scaling-up with Hashing

In order to perform cover version detection on a large database of content, it is necessary to find a cheaper and more efficient way of matching a probe song against many references. The brute-force approach of computing a full distance map for the probe against every possible reference scales as the product of the number of probes and the number of references; thus a system which makes it cheap to find a set of matching segments in all references for a given probe would be of great value. Bertin-Mahieux and Ellis [2] presented a system using hashed chroma landmarks as keys for a linear-time database lookup. Their system showed promise, and demonstrated a possible approach to large-scale cover-song detection but the reported performance numbers would not make for a practically-viable system. While landmark or 'interest point' detection has been extremely successful in the context of exact audio matching in noise [15] its effectiveness in such applications is largely due to the absolute invariance in the location of strong peaks in the spectrogram. For cover version identification the variability in performances, both in timing and in pitch, means that descriptors summarizing small constellations of interest points will necessarily be less discriminative than descriptors summarizing more complete features over a long time span. With this in mind, we explore some options for generating compact hashes of full intervalgrams for indexing and retrieval purposes.

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

Hashing Techniques Using the process outlined above, 32×32 pixel intervalgrams are generated from a signal at the rate of one per 240ms. To effectively find alternative performances of a piece of music in a large-scale database, it must be possible to do efficient lookup to find sequences of potentially matching intervalgrams. The use of locality-sensitive-hashing (LSH) techniques over long-timescale features for music information retrieval has previously been investigated and found to be useful for large datasets [3]. Various techniques based on locality-sensitive hashing (LSH) may be employed to generate a set of compact hashes which summarize the intervalgram, and which can be used as keys to look up likely matches in a key-value lookup system.

An effective technique for summarizing small images with a combination of wavelet analysis and Min-Hash was presented by Baluja and Covell [1] in the context of hashing spectrograms for exact audio matching. A similar system of wavelet decomposition was previously applied to image analysis [7].

Hashing of the Intervalgram In order to test the effectiveness of such techniques on intervalgrams, the system described in [1] was adapted to produce a compact locality-sensitive hash of the intervalgram features and tested at small scale using the framework and dataset above. To generate hashes, four consecutive 32×32 intervalgram frames are temporally averaged using a moving window, and the resulting summary intervalgram is decomposed into a set of wavelet coefficients using a Haar kernel. The top $t\%$ of the wavelet coefficients with the highest magnitude values are retained, and are represented by the sign of their value. In this way, a sparse bit-vector can be produced, with two bits per wavelet coefficient. The bit pattern 00 is used to represent an unused wavelet coefficient, and the patterns 10 and 01 are used to represent a retained positive and negative coefficient respectively. This sparse bit-vector is then hashed using the min-hash techniques described in [1].

A search of the parameter space over a large internal dataset led to the optimal values for the wavelet decomposition and min-hash as detailed in Table 2. In addition the choice of random permutations was optimised using the same dataset.

Parameter	Value
Top-wavelets used (%)	5
Hash bands	100
Number of permutations	255

Table 2. Optimal parameters for the wavelet decomposition and min-hash.

In this way, a 1024 element floating-point intervalgram matrix, costing 4096 bytes in storage, can be compactly summarised by a 100 byte min-hash representation. This reduction by a factor of 40 in the size of the representation

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

comes with a cost in matching ability, which can be quantified using the same framework as was used above for intervalgram matching. To compare hashes, similarity matrices were generated for each pair of songs in the covers80 dataset, as above but this time using the bytewise Hamming similarity between hashes. The dynamic programming technique described above was again employed to find the best path through the similarity matrix, and to provide a direct comparison with the raw intervalgram representation.

Figure 9 shows the overall scores matrix for the covers80 dataset computed using the hashes. Figure 10 shows the ROC curve computed from hashed intervalgrams. Performance is reduced from the full intervalgram case, and the ROC curve shows faster fall-off in precision with increasing recall, but recall at 99% precision is around 37.5%, reduced from 47.5% with full intervalgrams. Since this is the area of the curve which we wish to focus on for large-scale applications, it is gratifying to note that the massive decrease in fingerprint size does not lead to a correspondingly massive fall in achievable recall at high precision. In fact the recall at 99% precision is still higher after hashing than that of the unmodified Ellis MIREX 2006 features where recall was 35%.

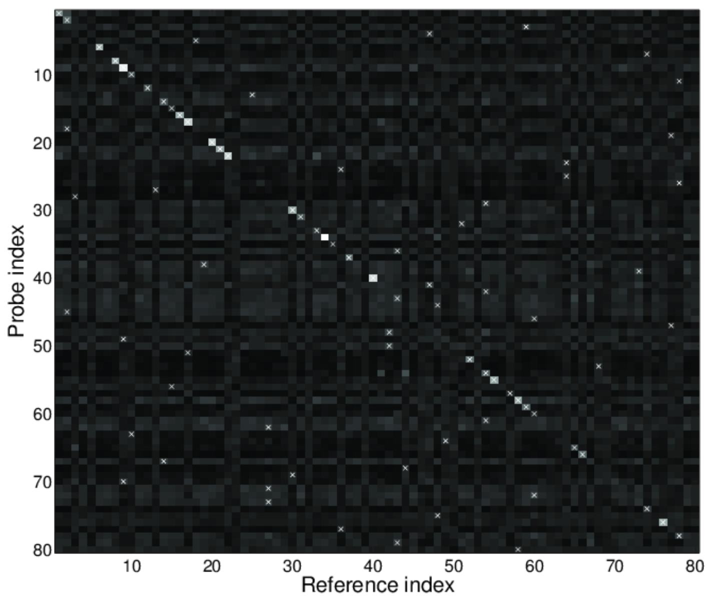

Fig. 9. Scores matrix for comparing all probes and references in the 'covers80' dataset using Hamming similarity over min-hashes. Lighter pixels denote higher scores, indicating a more likely match. White crosses denote the best-matching reference for each probe.

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

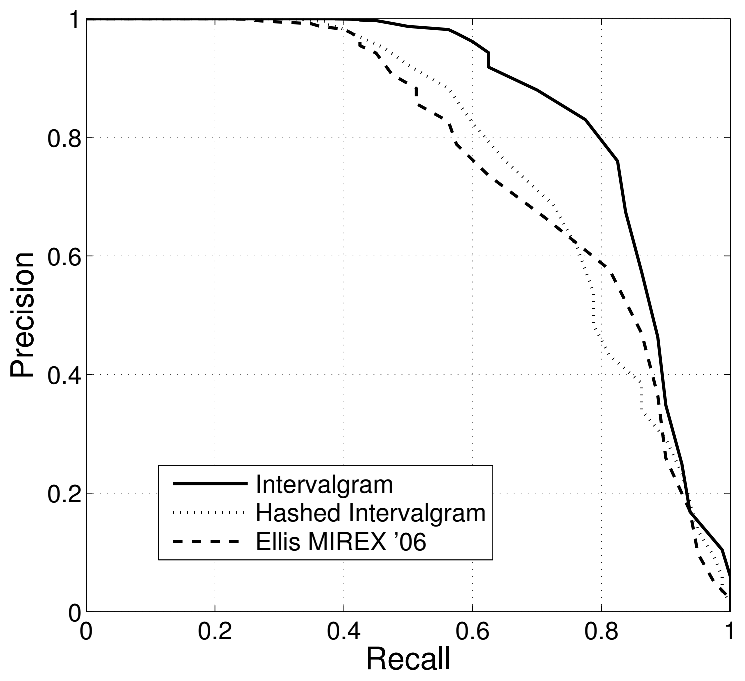

Fig. 10. ROC curves as in 8 with the addition of a curve for the hashed intervalgrams.

4 Discussion

We have introduced a new chroma-based feature for summarizing musical melodies, which does not require either beat tracking or exhaustive search for transposition invariance, and have demonstrated a good baseline performance on a standard dataset. However, we developed the intervalgram representation to be a suitable candidate for large-scale, highly robust cover-song detection. In the following sections we discuss some approaches to the application of the intervalgram in such a system.

4.1 SAI and Spectrogram-based Chroma

There was no great difference in performance between intervalgrams generated using the temporal profile of the SAI and intervalgrams generated using a spectrogram-based chroma feature. However, there are some small differences in different regions of the ROC curve. Recall at high precision is very similar for both forms of chroma features; as precision is allowed to fall, the SAI-based

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

features lead to slightly higher recall for a given precision, but the trend is reversed in the lower-precision end of the curve. This may suggest that there would be a benefit in combining both SAI-based and spectrogram-based chroma into a feature which makes use of both. There is some evidence to suggest that the temporal profile of the SAI may be robust to stimuli in which the pitch is ambiguous [6], but this result may be less relevant in the context of music.

4.2 Hashing Results

Compared to exact-match audio identification, this system is much more challenging, since the individual hash codes are noisier and less discriminative. The indexing stage necessarily has many false hits when it is tuned to get any reasonable recall, so there are still many (at least thousands out of a reference set of millions) of potential matches to score in detail before deciding whether there is a match. However, experiments with this small test set show that existing hashing techniques can be extremely effective at retaining the important detail in the full feature representation.

While the bytewise Hamming similarity is a reasonable measure for comparing fingerprints in the evaluation scheme described in this paper, it would not scale to very large libraries of reference content. In such a larger-scale system the matching could be implemented by grouping multiple bytes of the fingerprint and using these groups of bytes as keys into a lookup table storing candidate chunks of reference content which match the given key. A full discussion of such a system is beyond the scope of this paper, but this is the intended application of the hashing techniques describe here.

5 Conclusions

The intervalgram is a pitch-shift-independent feature for musical version recognition tasks. Like other features for such tasks, it is based on chroma features, but we have demonstrated that a chroma representation derived from the temporal profile of a stabilized auditory image gives comparable results to features derived from a spectrogram, and may provide complementary information. To achieve pitch-shift invariance, individual intervalgrams are shifted relative to a reference chroma vector, but no global shift invariance is used. Finally, to achieve some degree of tempo-invariance, variable-width time-offset bins are used to capture both local and longer-term features.

In this study, the performance of the intervalgram was tested by using dynamic-programming techniques to find the cheapest path through similarity matrices comparing a cover song to all references in the 'covers80' dataset. Intervalgrams, followed by dynamic-programming alignment and scoring, gave a precision at top-1 of 53.8%. This performance value, and the associated ROC curve, lies between the performance of the Ellis 2006 and Ellis 2007 MIREX entries (the latter of which was developed using the covers80 dataset).

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

The intervalgram has shown itself to be a promising feature for musical version recognition. It has good performance characteristics for high-precision matching with a low false-positive rate. Furthermore the algorithm is fairly simple and fully 'feed-forward', with no need for beat tracking or computation of global statistics. This means that it can be run in a streaming fashion, requiring only buffering of enough data to produce the first intervalgram before a stream of intervalgrams can be generated. This feature could make it suitable for applications like query-by-example in which absolute latency is an important factor.

In this study, we have also reported results which suggest that the intervalgram representation will lend itself well to large scale application when coupled with locality-sensitive hashing techniques such as wavelet-decomposition followed by minhash. The high precision at moderate recall which can be achieved with such techniques would allow for querying of a large database with a low false-positive rate, and our preliminary experiments have shown promise in this area.

References

1. Baluja, S., Covell, M.: Waveprint: Efficient wavelet-based audio fingerprinting. Pattern recognition 41(11), 3467–3480 (2008)
2. Bertin-Mahieux, T., Ellis, D.: Large-scale cover song recognition using hashed chroma landmarks. In: Proceedings of the International Symposium on Music Information Retrieval (ISMIR) (2011)
3. Casey, M., Rhodes, C., Slaney, M.: Analysis of minimum distances in high-dimensional musical spaces. IEEE Transactions on Audio, Speech, and Language Processing 16(5), 1015–1028 (2008)
4. Ellis, D.: The 'covers80' cover song data set (2007), `http://labrosa.ee.columbia.edu/projects/coversongs/covers80/`
5. Ellis, D., Cotton, C.: The 2007 LabROSA cover song detection system. MIREX 2007 Audio Cover Song Evaluation system description (2007)
6. Ives, D., Patterson, R.: Pitch strength decreases as f0 and harmonic resolution increase in complex tones composed exclusively of high harmonics. The Journal of the Acoustical Society of America 123, 2670 (2008)
7. Jacobs, C., Finkelstein, A., Salesin, D.: Fast multiresolution image querying. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. pp. 277–286. ACM (1995)
8. Lyon, R.: Cascades of two-pole-two-zero asymmetric resonators are good models of peripheral auditory function. Journal of the Acoustical Society of America 130(6), 3893 (2011)
9. Marolt, M.: A mid-level representation for melody-based retrieval in audio collections. Multimedia, IEEE Transactions on 10(8), 1617–1625 (2008)
10. Patterson, R., Robinson, K., Holdsworth, J., McKeown, D., Zhang, C., Allerhand, M.: Complex sounds and auditory images. In: Auditory physiology and perception, Proceedings of the 9th International Symposium on Hearing. pp. 429–446. Pergamon (1992)
11. Ravuri, S., Ellis, D.: Cover song detection: from high scores to general classification. In: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. pp. 65–68. IEEE (2010)

Author's self-archived copy from research.google.com.
The final publication is available at springerlink.com.
http://link.springer.com/chapter/10.1007/978-3-642-41248-6_11

12. Serra Julia, J.: Identification of versions of the same musical composition by processing audio descriptions. Ph.D. thesis, Universitat Pompeu Fabra (2011)
13. Tsai, W., Yu, H., Wang, H.: Using the similarity of main melodies to identify cover versions of popular songs for music document retrieval. Journal of Information Science and Engineering 24(6), 1669–1687 (2008)
14. Walters, T.: Auditory-based processing of communication sounds. Ph.D. thesis, University of Cambridge (2011)
15. Wang, A.: An industrial strength audio search algorithm. In: Proceedings of the International Symposium on Music Information Retrieval (ISMIR). vol. 2 (2003)
16. Yang, C.: Music database retrieval based on spectral similarity. In: Proceedings of the International Symposium on Music Information Retrieval (ISMIR) (2001)