CONTOUR-BASED MELODY REPRESENTATION: AN ANALYTICAL STUDY

Sumantra Dutta Roy, Preeti Rao and Ameya S. Galinde

Department of Electrical Engineering,
IIT Bombay, Powai, Mumbai - 400 076.
{sumantra, prao, ameya}@ee.iitb.ac.in

ABSTRACT

In this paper, we identify parameters crucial to the performance of a Query By Humming (QBH) system, and present an analytical approach to determining optimal values of such parameters. Existing systems use heuristically chosen parameters - our analytical results are in accordance with such values. We present results of experimentation with simulated data, as well as an actual melody database of a QBH system.

1. INTRODUCTION

Query by Humming (hereafter, QBH) has emerged as an important area of research in audio-based search engines, and building efficient Human-Computer Interfaces (HCI) [1], [2], [3], [4], [5], [6], [7]. Thus far, related work in the area has only considered the use of a 3-, 5- or 7-level pitch contour [8], [6], [7], [9]. - this is usually based on empirical studies. To the best of our knowledge, there has been no attempt to derive such an estimate using analytical methods. This paper presents an analytical study of QBH systems. We develop a new coefficient to evaluate the performance of a QBH system. Our results of experiments with analytical models, as well as an actual melody database give results consistent with those in existing literature. While this paper considers the special case of uniform tune lengths, our current research involves extending this to a Dynamic Programming-based framework to handle the most general case.

2. THE REPRESENTATION SCHEME

We assume all notes in a musical piece lie only on a quantized set of absolute pitch values (in Hz). The ratio of any two adjacent pitch values is $2^{1/12}$. The interval between two such adjacent values is a 'semitone', and an ensemble of 12 such pitch values or notes is an 'octave' [3]. Without loss of generality, we may assume a logarithmic scale for such notes with the least count on this scale corresponding to a semitone. (This ensures equal distances along the logarithmic scale, corresponding to any two adjacent semitones) Our representation of a user query or any melody consists of a sequence of numbers where each number signifies the 'distance' between the current note and the previous note, or more precisely, the number of semitones lying in between these two adjacent notes in the musical piece [8].

As an example, we can consider a (fairly realistic) case where a note will be no more than an octave apart from its previous note *i.e.*, it can either ascend by a maximum of 1 octave (+12 semitones) or descend by a maximum of 1 octave (-12 semitones). This is quite a reasonable assumption, since most musical pieces proceed with a steady rise or fall of notes and even a pitch change of one octave between adjacent notes is rarely encountered. We may represent this span of *relative notes* by the set $[-12, +12]$. For a casual singer's notes sung slightly off-key, we approximate the corresponding relative note to the closest one on our scale. As an illustrative example, let us consider the following sequence of 4 notes (all notes lying in the same octave) in Western music notation (all notes in the same scale, say Scale C Major): [Mi

So Fa Ti]. In our notation, this is [+3 -2 +6]. Such a representation makes a particular melodic contour invariant to pitch transpositions.

Kageyama *et al.* [1] and Ghias *et al.* [2] propose the use of static heuristic thresholds for splitting the melodic contour into the desired number of levels. Sonoda *et al.* [9] propose the use of dynamic determination of thresholds. In their method, they determine the thresholds based on the histograms for the relative pitch values, to get a 3-level contour. While this has the advantage of being optimal for a given set of songs, it is highly unlikely that the number of songs in a database will remain constant over a period of time. The thresholds, and hence the entire representation of a song may change with the addition of even a single song to the database. This is costly operation in terms of time (even if done off-line) and hence, not a desirable property. Kim *et al.* consider empirical evidence to decide on choosing the number of levels of quantization, k. However, to the best of our knowledge, no existing approach considers an analytical approach to determining the performance of a QBH system, or the number of optimum number of levels, for instance. With a view to reduce the number of parameters, we adopt a uniform quantization scheme to find out the optimal value of k, in our analysis.

2.1. Statement of the Problem

For our analysis, we consider a range of relative notes R. We further assume that the finest level of quantization (with each mark specifying a relative note) will give rise to N relative notes, lying between two limits r_0 and r_N, respectively[1]. *Given a $N-$ level quantization, we wish to divide this range into k intervals (using $k-1$ markers between r_0 and r_N) uniformly - The aim is to find an optimal value for k.* The range of possible values of k is 1 and N. Two desirable requirements governing the choice of a suitable value of k are:

- *Fidelity*: a desirable requirement is to have a close match between the hummed contour, and a close one from the database. A desirable requirement is to have a unique representation for each song in the database.
- *Robust Matching*: a strategy should ideally have adequate robustness to cater to different untrained singers, who may occasionally go off-key.

The task at hand is to find this optimal value of k, as described above. Clearly, these are contradictory requirements. A large value of k will ensure that the query melodic contour can be matched with greater precision to one in the database, thus reducing chances of a false match. However, the resultant error-resilience reduces, and vice versa.

In view of these contradictory requirements, we propose a new function, a minimum of which will result in an optimal value of k. We define the **Demerit Coefficient** $\mathcal{M}_D(k,\mu)$ for a Database of songs D, as follows:

$$\mathcal{M}_D(k,\mu) \stackrel{\triangle}{=} \mu\mathcal{F}_D(k) + (1-\mu)\mathcal{R}_D(k) \quad (1)$$

Here, $\mathcal{F}_D(k)$ and $\mathcal{R}_D(k)$ represent the **Fidelity** and **Robust-Match** functions, respectively, which we define below. μ is an arbitrary scalar coefficient which specifies the required relative percentage of the two constituent terms in the expression for $\mathcal{M}_D(k,\mu)$. The task at hand is to find $argmin_k\ \mathcal{M}_D(k,\mu)$ *i.e.*, that value of k for which $\mathcal{M}_D(k,\mu)$ achieves a minimum value.

We define l_k as the 'length' of an interval along the relative note axis, as follows: $l_k \stackrel{\triangle}{=} \lfloor (r_N - r_0)/k \rfloor$, where the $\lfloor\ \rfloor$ notation denotes the largest integer smaller than the number. In the following sections, we define $\mathcal{F}_D(k)$ and $\mathcal{R}_D(k)$.

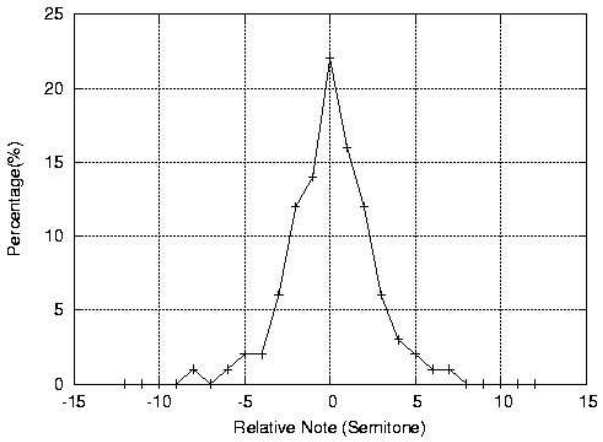
2.2. The Fidelity Term $\mathcal{F}_D(k)$

We define the Fidelity Term $\mathcal{F}_D(k)$ as follows:

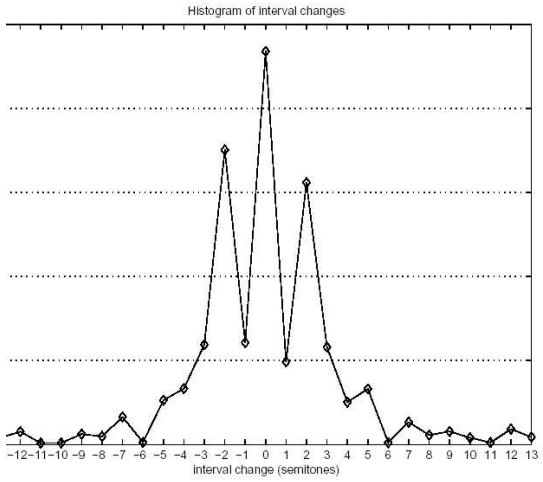
$$\mathcal{F}_D(k) \stackrel{\triangle}{=} \frac{1}{\widetilde{\mathcal{F}_D(k)}}\Big[\sum_{\forall\ x}[x - l_k(x\ div\ l_k)]^2\ p^D[x]\Big]^{1/2} \quad (2)$$

Here, the summation is over all relative notes x in the songs in the given database D, (*div* denotes

[1]The conversion between the two discrete scales $[r_0, r_N]$ and $[r'_{-N/2}, r'_{N/2}]$ is a straightforward linear and invertible function. Throughout this paper, we use the two interchangeably.



(a)



(b)

Figure 1: Distribution of relative notes: $p^D[\cdot]$ for (a) our database, and (b) an MIT database (taken from [8])

integer division *i.e.*, the quotient) and $p^D[x]$ denotes the discrete probability of a particular relative note x. This is a characteristic of the particular database, and depends its constituent songs. Figure 1 show samples of such curves from our database of songs, as well as an MIT database [8]. $\widehat{\mathcal{F}_D(k)}$ is a normalizing factor. We may take this as $l_k - 1$ for example, or simply the maximum of the terms being summed up.

Having k intervals implies that for a given relative note x, all relative notes y lying in the discrete interval $l_k(x \; div \; l_k) \leq y < l_k(x \; div \; l_k \; + \; 1)$ would be characterized by a point, which we consider (without loss of generality) as being the left limit of the interval. Thus, $\mathcal{F}_D(k)$ is a measure of the average deviation of relative notes which would get classified by the k-level system as being in it's corresponding interval. For a system to have greater fidelity of a perfectly-sung melody to one stored in the database, the $\mathcal{F}_D(k)$ term should be small. Otherwise, other melodies with each corresponding relative note lying in the same interval as the target song, would get wrongly identified as the one in question. We note that the above definition for $\mathcal{F}_D(k)$ is quite restrictive in the sense of seeking a match between two melodic contours of the same length. Section 4 looks at extensions of this approach to deal with the most general case.

2.3. The Robust-Match Term $\mathcal{R}_D(k)$

We define the Robust-match term $\mathcal{R}_D(k)$ as follows:

$$\mathcal{R}_D(k) \stackrel{\triangle}{=} \frac{1}{\widehat{\mathcal{R}_D(k)}} \sum_{\forall \; x} [\sum_{y} (y - x)^2 \; p_x^U[y]]^{1/2} \; p^D[x] \tag{3}$$

The outer summation is over all relative notes x

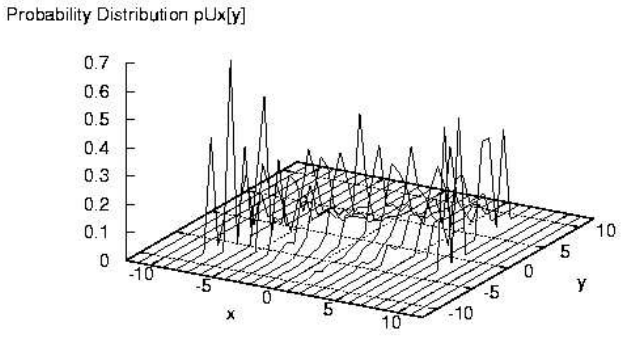
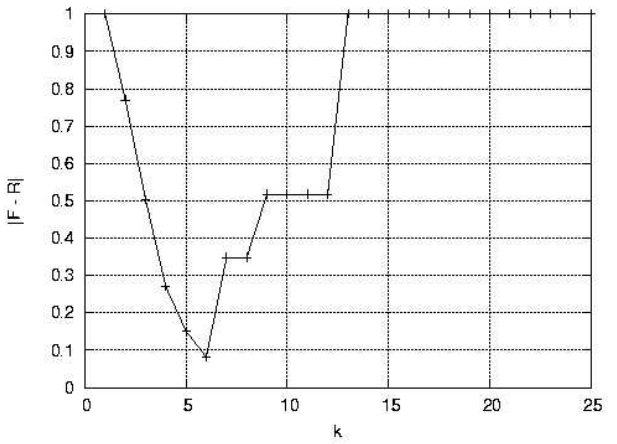
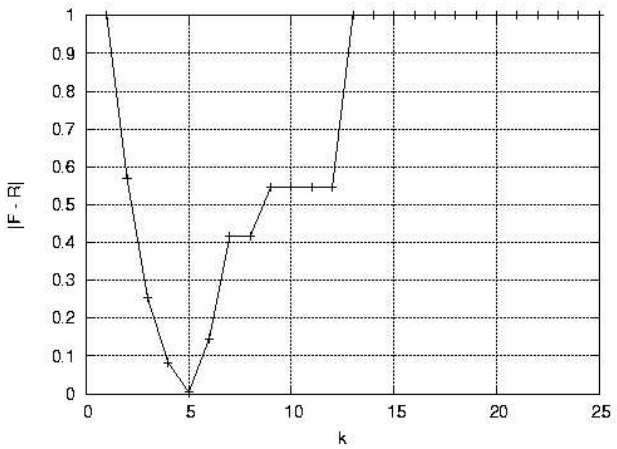


Figure 2: The probability mass function $p_x^U[y]$ for relative notes in our database (The Robust Match Function $\mathcal{R}_D(k)$, Section 2.3)

in the songs in the given database D. The summation for y is over all relative notes which are not in the same interval as the relative note x *i.e.*, $\{r_0 \leq y < l_k(x \; div \; l_k)\} \; \cup \; \{l_k(x \; div \; l_k \; + \; 1) \leq y \leq r_N\}$. Here, $\widehat{\mathcal{R}_D(k)}$ is the normalization factor. We may take this to be $R - l_k \; (= r_N - r_0 - l_k)$, or simply the maximum of the terms being summed



(a)



(b)

Figure 3: The optimal k: A plot of $|\mathcal{F}_D(k) - \mathcal{R}_D(k)|$ vs. k (a) analytical simulation: $p^D[\cdot]$ and $p_x^U[\cdot]$ as Gaussians (Section 3.1); and (b) Actual database statistics (Section 3.2)

up. Here, $p_x^U[\cdot]$ is the one-dimensional probability mass function of all user query relative notes for a given relative note x. (Figure 2 shows the probability mass function of $p_x^U[\cdot]$ for different relative notes x, in our database) It is important to note that while the Fidelity term $\mathcal{F}_D(k)$ depends only on the distribution of relative notes of songs in a given database D, $\mathcal{R}_D(k)$ depends on the statistics of the distribution of typical user queries, as well. The motivation behind taking $p_x^U[\cdot]$ into account is the fact that casual users often go off-key when the change in notes is quite drastic. The above formulation handles such an observation as statistics, and uses it to advantage in finding a suitable value of k (our ultimate objective). Clearly, a high value of $\mathcal{F}_D(k)$ would imply more robustness of the matching procedure to notes sung off-key by a casual user.

3. EXPERIMENTAL RESULTS AND DISCUSSION

We approach the problem using methods of variational calculus. We need the value of k for which the Demerit Coefficient is minimum for a given database D of songs (Equation 1). We differentiate this with respect to the two variables k and μ, and set these to zero. We can find out the value of k from the partial differentiation with respect to μ. We evaluate $|\mathcal{F}_D(k) - \mathcal{R}_D(k)|$ for varying values of k, and check for minima close to zero. Further, we can find the optimal value of μ by numerical differentiation. We evaluate $|\frac{\delta \mathcal{M}_D(k,\mu)}{\delta k}|$ for $\delta k = 1$ (the smallest possible discrete change in k), and find out the value of μ, for which $|\delta \mathcal{M}_D(k,\mu)|$ is minimum. This is the required value of μ. For our experimentation, we have considered both the normalization factors $\widehat{\mathcal{F}_D(k)}$ and $\widehat{\mathcal{R}_D(k)}$ as the maximum term in the respective summations. A system designer may like to have a control over the relative proportion of $\mathcal{F}_D(k)$ and $\mathcal{R}_D(k)$ for a particular database. In our formulation, the flexibility in choosing the normalization factors takes this desirable aspect into account.

3.1. Studies with Representative Models

An examination of Figure 1 reveals that it is reasonable to use a Gaussian to approximate $p^D[\cdot]$, the distribution of relative notes for songs in a database. Figure 2 shows the function for actual values corresponding to our song database. Section 2.3 gives the motivation for having a $p_x^U[\cdot]$ modeling. Casual users often tend to go more off-key with increasing values of relative notes, irrespective of the direction of change. Owing to these reasons, we model $p_x^U[\cdot]$ as a Gaussian with standard deviation proportional to x. Choosing $p^D[\cdot]$ to be a Gaussian close to our actual $p^D[\cdot]$ curve (Figure 1(a)), we obtain the optimal values of the

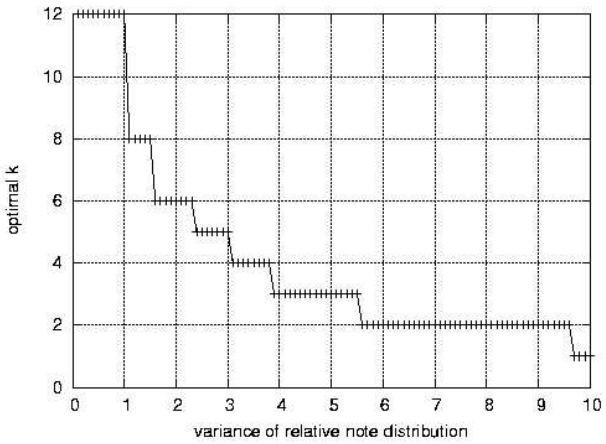


Figure 4: Variation of the optimal k for different σ values: simulation, with $p^D[\cdot]$ assumed to be a Gaussian.

parameters k and μ as 6 and 0.01. Figure 3(a) shows a plot $|\mathcal{F}_D(k) - \mathcal{R}_D(k)|$ for varying values of k. The portions of flatness in the curve are due to the nature of the floor function in l_k and hence, $|\mathcal{F}_D(k) - \mathcal{R}_D(k)|$ as well.

A set of analytical models permits one to experiment by changing various system parameters. For example, we have varied the variance of the $p^D[\cdot]$ Gaussian to test its effect on the optimal k value. Figure 4 shows this variation.

3.2. Studies with Data from an Actual Melody Retrieval System

We have experimented with a melody database [6], [7] of 69 Hindi film song phrases, with an average of about 10 notes in each, sung by one reference singer and 3 casual singers. Figure 3 (b) shows the corresponding $|\mathcal{F}_D(k) - \mathcal{R}_D(k)|$ curve for different values of k. In this case, the optimum corresponds to $k = 5$ and $\mu = 0.01$.

4. CONCLUSIONS

In this paper, we propose a new analytical approach for obtaining important parameters of a QBH system. We identify these, and present an analytical study of the performance of the QBH systems with respect to these parameters. Existing systems choose these parameters heuristically - our results are in accordance with these. We present results of experimentation with simulated data, as well as an actual melody database of a QBH system. Future work includes extending the idea of the Fidelity term $\mathcal{F}_D(k)$ to deal with varying query lengths in a Dynamic Programming-based framework.

REFERENCES

[1] T. Kageyama, K. Mochiezuki, and Y. Takashima, "Melody Retrieval with Humming," in *Proc. ICMC*, 1993, pp. 349 – 351.

[2] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query By Humming - Musical Information Retrieval in an Audio Database," in *Proc. ACM Multimedia*, 1995.

[3] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Sunningham, "Toward the Digital Music Library: Tune Retrieval from Acoustic Input," in *Proc. ACM Digital Libraries*, 1996.

[4] Midilib, `http://www-mmdb.iai.unibonn.de/forschungprojekte/midilib/english`.

[5] Tuneserver, `http://tuneserver.de`.

[6] M. Anand Raju and P. Rao, "Towards an Automatic Melody Retrieval System," in *Proc. National Conference on Communications (NCC)*, 2002.

[7] M. Anand Raju, B. Sundaram, and P. Rao, "TANSEN: A Query-By-Humming based Music Retrieval System," in *Proc. National Conference on Communications (NCC)*, 2003.

[8] Y. E. Kim, W. Chai, R. Garcia, and B. Vercoe, "Analysis of a Contour-based Representation for Melody," in *Proc. International Symposium on Music Information Retrieval (ISMIR)*, October 2000.

[9] T. Sonoda, M. Goto, and Y. Muraoka, "A WWW-based Melody Retrieval System," in *Proc. ICMC*, October 1998, pp. 349 – 352.