CS3242 assignment 2 report
Content-based music retrieval

Luong Minh Thang & Nguyen Quang Minh Tuan

1. INTRODUCTION

With the development of the Internet, searching for information has proved to be a vital and necessary part, and particularly for multimedia, the search task becomes even more challenging. When dealing with multimedia content, plain-text search engines are not enough because they are too simple and have very limited functions. In this report, we will discuss new algorithms of content-based music retrieval using slope matching. This report is a part of CS3242 - Multimedia technology - module at School of Computing, NUS. We were given source code for a basic query-by-humming music-retrieval engine and our task were to modify it to make its performance better.

2. APPROACHES

At the beginning, we identify that there are two possible areas where could modify the given source code as described in (Zhu, Xu, & Kankanhalli, 2001) to achieve better performance, which are feature extraction and matching algorithm. Feature extraction is , however, relatively hard to modify as we need to deal with audio processing. As such we decide to modify the matching algorithm. The existing matching algorithm is quite complex, so we decide that each of us will write a totally new matching algorithm. After that, the top results by each algorithm will be synthesized to give the final best results. Our two new algorithms are *Metric Distance with Gap detection* and *Dynamic Programming with Local alignment*.

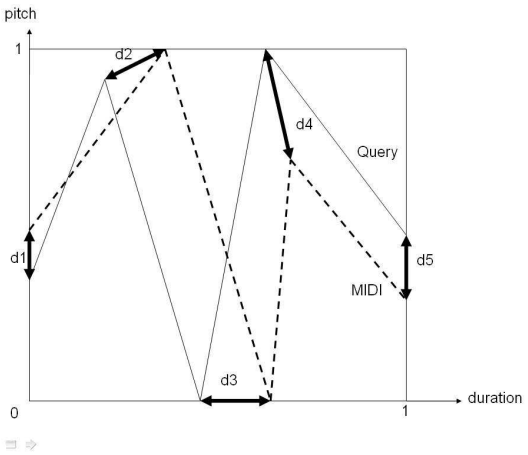
2.1. Metric distance and gap detection

In this approach, we used the concept of metric distance to calculate the difference between the humming query and a sequence of slopes in the MIDI files at the macro level. Assume that we have a humming query (H) and part of the MIDI files (M), with the same number of slopes (n). Firstly, we normalized H and M such that the pitch range of the slopes is from 0 to 1, and the total duration of H and M is 1. Therefore, we can represent H and M in the same coordinate system.

The difference between H and M can be calculate as:

$$diff(H, M) = \sum_{i=1}^{n} (\sqrt{(H_{di} - M_{di})^2 + (H_{pi} - M_{pi})^2})$$

where H_{di}, M_{di} refer to the durations of slope i; H_{pi} M_{pi} refer to the pitches.



We did a sequential search for all the MIDI files to find the one having the part with minimum diff(H,M). However, the results acquired were not satisfactory, so we decided to make an enhancement for this algorithm by adding gap detection feature. We observed that people often start the humming queries at the beginning of a song or right after a silent note (which we called a gap). Hence, it seemed to be reasonable that we concentrated more to these parts of the MIDI files. Generally, parts right after a gap of the MIDI file are more important than other parts. To detect a gap, we averaged all the notes of the MIDI files and choose a threshold for the gap (we chose 1.5 times duration of the average note). If M is right after a gap, then diff(H,M) will be reduced by 2/3 to 3/4 in our implementation. In fact, the metric distance with gap detection works reasonably well and it proved to be a good component for the original slope matching method.

2.2. Dynamic programming - local alignment

In this approach, we employ a form dynamic programming *DP* algorithm, *local alignment*, to tackle the problem. Local alignment has been well studied and extensively used in computational biology (Gusfield, 1997) to find the two best-match subsequences from the given two sequences. In the context of our query-by-humming system, we have two sequences of slopes $M[1..m]$ and $H[1..n]$ representing humming query (H) and MIDI file (M) accordingly. The idea is we want to find a subsequence in M that best matches H, and obtain a similarity score for H and M. Using DP approach, we expect the program to be more error-tolerant as several slopes could be skipped before matching next slopes; thus, the accuracy could be enhanced.

More specifically, base condition and recurrence relation in DP are described below in which $V(i, j)$ gives the best value when locally aligning $M[1..i]$ and $H[1..j]$. $s(M[i], H[j])$ gives the similarity value of MIDI slope i vs. humming slope j, while s(M[i], _) and s(_ , H[j]) are skip penalties, all of which we will provide the formula later. It is worth to mention that the **comparing value 0** in the recurrence relation is the subtle part of

local alignment, which functions as the restarting for the alignment path.

Base condition:

$$\begin{cases} V(0,j) = 0 \\ V(i,0) = 0 \end{cases} \tag{1}$$

Recurrence relation:

$$V(i,j) = max \begin{cases} V(i-1,j-1) + s(M[i],H[j]) & \text{if i} > 0, \text{j} > 0 \\ V(i-1,j) + s(M[i],\_) & \text{if i} > 0 \\ V(i,j-1) + s(\_,H[j]) & \text{if j} > 0 \\ 0 \end{cases} \tag{2}$$

Given two humming and MIDI slopes with normalized durations ($\in [0,1]$) and pitch-change values ($\in [-1,1]$), we compute the similarity based on the difference ($\in [0,1]$)as below:

$$\begin{aligned} \text{s(M[i],H[j])} &= -\text{diff} + \text{offset} && (3) \\ &= -(w * \text{duration\_diff} + (1-w) * \text{pitch\_diff}) + \text{offset} && (4) \\ &= -(w * |\text{M[i]\_p} - \text{H[j]\_p}|/2 + (1-w) * |\text{M[i]\_d} - \text{H[j]\_d}|) + \text{offset} && (5) \end{aligned}$$

Experimentally, we use w = 0.7 to give higher weight for pitch values, offset = 0.3 to have good mixture of negative and positive similarity values for local alignment to work, and skip penalty = -0.01, which give good results. The detailed experiment results are shown later in Section 3.

2.3. Combined approach

After implementing our two new algorithms, now we have three different algorithms for slope-matching part. To combine three algorithm, we proposed two approaches:

- Using gap dectection algorithm and dynamic-programming algorithm as filters for the original slope-matching algorithm. We called it the serial approach.
- Running three algorithms one by one, and choosing the top 20 results of each algorithm. We will then merged the 60 results together to get the final list. We only considered the first 15 of the final list. We called this approach the parallel approach.

After trying both approaches, the parallel approach has shown to perform better than the serial approach. One of the reason is because the relevant MIDI files is usually in the top 5 results for all the 3 algorithms. Another reason is that previous algorithm

might filter correct result before turning to the next algorithm as in the serial approach. Therefore, we pursue the parallel approach to combining the there result list.

In the parallel approach, we combine the there list of top 20 results from the three algorithms by computing a weighted average similarity for each song as follow:

$$\text{final\_sim(A)} = [1.2 * \text{O\_sim(A)} + 0.9 * \text{LA\_sim (A)} + 0.9 * GD\_sim(A)]/3$$

O_sim(A), LA_sim(A), GD_sim(A) stand for the similarity of song A in the original, LA, and GD algorithms respectively. If A is not in a top 20 result of an algorithm, its corresponding similarity value will be 0.

3. EVALUATION

Our dataset is constructed by first running the original through all available humming queries. We then pick up the three familiar songs with several queries associated, which are "Happy birthday", "Silent night", and "Qing wang". We evaluate five different implementation, i.e. the original algorithm, gap detection (GD), local alignment (LA), GD & LA, and combined algorithm, on that dataset, and obtain the detailed results as in the Table 2.

We first notice three challenging queries in which all implementations have difficulties in retrieving the results. They are queries 9 (very soft voice), and 11, 20 (humming at different tempos). In overall, GD and LA perform roughly the same, and somewhat complementary in which GD retrieve better results for "Happy birthday" queries, while LA is better at "Silent night" queries. The combination of GD and LA gives comparable performance with the original algorithm, and the combined algorithm perform best.

To better compare the above implementation, we further compute the three metrics Recall, Precision at seen-relevant documents, and F measure for each algorithm, and details are given in Table 1. Again, it is true with our observation that LA slightly outperforms GD with recalls of 56.08% vs. 50.96%, and precisions at 52.32% vs. 46.79%. The combination of GD and LA gives comparable recall and precision to the original algorithm with F-measure values of 72.15% vs. 73.5%. Finally, the combined version of the original algorithm, GD, and LA best perform at recall 91.67%, and precision 74.38%.

4. DISCUSSION

We first discuss on our new implementations in this system, which are gap detection, and local alignment. Gap detection can match slopes sequentially, so it performs well with in-tune queries. Moreover, with the mechanism of detecting gap, the algorithm could quickly match when there is clear occurrences of gaps, and rule out other false matches with no gaps. However, when dealing with out-of-rhythm or out-of-tune queries,

	Recall	Precision	F measure
Original	78.84	68.83	73.5
Gap detection	50.96	46.79	48.78
Local alignment	56.08	52.32	54.14
GD & LA	71.47	72.83	72.15
Combined	**91.67**	**74.38**	**82.12**

Table 1. Recall, precision, and F-measure statistics (in %)

	Query	Song	Original	GD	LA	GD & LA	Combined
1	0114144842	Happy Birthday	X	1st	X	1st	4th
2	0114150748	Silent Night	1, 2, 3, 4th	1,2,3,4th	2,3, 6th	2,3, 6th	1, 2, 3, 4th
3	0226104251	Silent Night	1, 2, 3rd, 6th	1,2, 3rd	1, 2, 3, 6th	1, 2, 3, 6th	1, 2, 3,6th
4	0226133350	Silent Night	1, 2, 3, 14th	1,2, 3rd	1, 2, 3, 11th	1, 2, 3, 11th	1,2, 3rd,14th
5	0226145115	Silent Night	1, 2, 3, 4th	X	1, 2, 3rd	1, 2, 3rd	1, 2, 3, 5th
6	0312102415	Happy birthday	1st	1st	1st	1st	1st
7	0328093006	Happy birthday	1st	3rd	3rd	3rd	1st
8	0328093240	Happy birthday	Corrupted	1st	2nd	1st	1st
9	0328142041	Silent Night	10, 11, 15th	X	X	X	12,13th
10	0620111910	Silent Night	1, 2, 3rd	X	1, 2, 3, 4th	1, 2, 3, 4th	1, 2, 3, 12th
11	0704164259	Happy birthday	X	X	X	X	X
12	0707102307	Happy birthday	1st	1st	6th	1st	1st
13	0707102335	Qing wang	1, 2, 3rd	X	1st	1st	1, 6, 7th
14	0707102455	Silent Night	1, 2, 3, 6th	1,2, 3rd	1, 2, 3rd	1, 2, 3rd	1,2, 3,5th
15	0710140837	Happy birthday	1st	X	1st	1st	1st
16	0710141732	Happy birthday	1st	8th	1st	1st	1st
17	0710143633	Qing wang	X	X	6th	6th	14th
18	0710150819	Qing wang	1, 2, 3rd	1st	X	1st	1, 2, 3rd
19	0710152442	Happy birthday	8th	1st	X	1st	1st
20	0710153421	Happy birthday	X	X	11th	11th	12th
21	0710154400	Happy birthday	1st	8th	X	8th	1st
22	0711095404	Happy birthday	2nd	1st	8th	1st	1st
23	0723085153	Silent nights	2,3,4,5th	X	X	X	1,4,5,6th
24	1015155425	Happy birthday	1st	X	3rd	3rd	1st
25	1015160015	Qing wang	1, 2, 3rd	X	1st	1st	1, 6, 7th
26	1015165838	Qing wang	1, 2, 9th	2,3,16,17th	1st	1st	3, 4, 5th

Table 2. Compare results across different implementations: original algorithm, gap detection (GD), local alignment (LA), GD & LA, and combined algorithm

its performance is not satisfactory, especially when the out-of-tune part happens at the beginning of the query. Dynamic programming, on the other hand, could complement for the gap detection approach in the sense that DP is more tolerant to out-of-tune queries, e.g. several slopes could be skipped until matching a slope. The limitation of DP is that the irrelevant MIDI file could happen to have the slope shape that result in a high-value alignment path. We expect that if we have two levels of retrieval, i.e. coarse grain and

fine grain, DP could be used for the latter level and results in better performance. The parameters for DP could be tuned to provide better retrieval as well.

Another important thing to note is that our current GD and LA implementations use only global features, which are slope duration and pitch changes. As compare to the use of local features including note duration and pitches of the original algorithm, we have obtained a promising precision of 72.83%, which is higher than 68.83% precision of the original, and a recall of 71.47% which is acceptable as compare to 78.84% recall of the original. We believe that we could possibly outperform the original system if local features are considered.

5. CONCLUSION

In this assignment, we have achieved the following things:

- We have discovered new ways for doing slope matching. Although comparing with the original system, our algorithms are much simpler, but still the performance has reached an acceptable level.
- This assignment has provided us a good exposure to content-based retrieval system, particularly real query-by-humming retrieval system, which is a good basis for our further study in multimedia IR.
- It might look intimidating at first to modify a published algorithm. However, as we go further into the assignment, the modification is really feasible, in which we have learned how to identify important parts of the system, and had a good starting point by identifying a set of testing queries.

In future work, we will further investigate in the use of local features. Besides, the application could be enhanced to perform online recording, as well as be able to playback matched position directly.

6. REFERENCES

[1] Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology.* New York, NY, USA: Cambridge University Press.

[2] Zhu, Y., Xu, C., & Kankanhalli, M. (2001). Melody curve processing for music retrieval. *icme*, *00*, 2001, 73.